Red Hat Enterprise Linux 8

Configuring and managing networking

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

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

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION ........................................ 12

CHAPTER 1. OVERVIEW OF NETWORKING TOPICS ............................................. 13

1.1. IP VERSUS NON-IP NETWORKS ................................................................. 13
   Categories of network communication ...................................................... 13
1.2. STATIC VERSUS DYNAMIC IP ADDRESSING ............................................. 13
1.3. CONFIGURING THE DHCP CLIENT BEHAVIOR ........................................ 14
   Configuring the DHCP timeout ............................................................... 14
   Lease renewal and expiration ............................................................... 14
   1.3.1. Making DHCPv4 persistent .......................................................... 14
1.4. SETTING THE WIRELESS REGULATORY DOMAIN ................................... 15
   Additional resources ............................................................................. 15
1.5. USING NETWORK KERNEL TUNABLES WITH SYSCTL ................................. 15

CHAPTER 2. CONSISTENT NETWORK INTERFACE DEVICE NAMING .................... 16

2.1. NETWORK INTERFACE DEVICE NAMING HIERARCHY .................................. 16
2.2. HOW THE NETWORK DEVICE RENAMING WORKS .................................... 17
   Additional resources ............................................................................. 18
2.3. PREDICTABLE NETWORK INTERFACE DEVICE NAMES ON THE X86_64 PLATFORM EXPLAINED ................................................................. 18
2.4. PREDICTABLE NETWORK INTERFACE DEVICE NAMES ON THE SYSTEM Z PLATFORM EXPLAINED ................................................................. 18
2.5. DISABLING CONSISTENT INTERFACE DEVICE NAMING DURING THE INSTALLATION ................................................................. 19
   Procedure ............................................................................................. 19
   Additional resources ............................................................................. 19
2.6. DISABLING CONSISTENT INTERFACE DEVICE NAMING ON AN INSTALLED SYSTEM ................................................................. 19
   Prerequisites ......................................................................................... 20
   Procedure ............................................................................................. 20
2.7. USING PREFIXDEVNAME FOR NAMING OF ETHERNET NETWORK INTERFACES ................................................................................................. 20
   2.7.1. Introduction to prefixdevname ....................................................... 20
   2.7.2. Setting prefixdevname .................................................................. 21
   2.7.3. Limitations of prefixdevname ....................................................... 21
   Related information ............................................................................... 21

CHAPTER 3. NETCONSOLE .................................................................................... 22

3.1. CONFIGURING NETCONSOLE ................................................................. 22
   Prerequisites ......................................................................................... 22
   Procedure ............................................................................................. 22
   Additional resources ............................................................................. 22

CHAPTER 4. GETTING STARTED WITH MANAGING NETWORKING WITH NETWORKMANAGER ................................................................. 23

4.1. OVERVIEW OF NETWORKMANAGER ...................................................... 23
   4.1.1. Benefits of using NetworkManager ................................................ 23
   Additional resources ............................................................................. 23
4.2. INSTALLING NETWORKMANAGER .......................................................... 23
   Additional resources ............................................................................. 24
4.3. CHECKING THE STATUS OF NETWORKMANAGER ................................... 24
4.4. STARTING NETWORKMANAGER .............................................................. 24
4.5. NETWORKMANAGER TOOLS .................................................................. 24
   Additional resources ............................................................................. 25
4.6. RUNNING DISPATCHER SCRIPTS .............................................................. 25
   Additional resources ............................................................................. 25
4.7. USING NETWORKMANAGER WITH SYSCONFIG FILES ......................... 25
   4.7.1. Legacy network scripts support ..................................................... 26
CHAPTER 8. CONFIGURING NETWORKING WITH GNOME GUI ........................................ 71

8.1. CONNECTING TO A NETWORK USING THE GNOME SHELL NETWORK CONNECTION ICON 71
8.2. CREATING A NETWORK CONNECTION USING CONTROL-CENTER 72
  Procedure 72
8.3. CONFIGURING A WIRED (ETHERNET) CONNECTION USING CONTROL-CENTER 73
  Procedure 73
  Basic configuration options 73
  Configuring IPv4 settings for wired with control-center 74
  Configuring IPv6 settings for wired with control center 75
  Configuring 802.1X security for wired with control-center 76
  Configuring TLS settings 77
  Configuring PWD settings 78
  Configuring FAST settings 78
  Configuring tunneled TLS settings 79
  Configuring protected EAP (PEAP) settings 79
8.4. CONFIGURING A WIRED (ETHERNET) CONNECTION USING NM-CONNECTION-EDITOR 80
  Prerequisites 80
  Procedure 80
8.5. CONNECTING TO A WI-FI NETWORK 81
  Procedure 81
  Additional resources 82
8.6. CONFIGURING A WI-FI CONNECTION USING CONTROL-CENTER 82
  Prerequisites 82
  Procedure 82
8.7. CONFIGURING A VPN CONNECTION WITH CONTROL-CENTER 85
  Prerequisites 85
  Procedure 85
  Additional resources 89
8.8. CONFIGURING A VPN CONNECTION USING NM-CONNECTION-EDITOR 89
  Prerequisites 89
  Procedure 89
  Additional resources 91
8.9. CONFIGURING A STATIC ROUTE USING CONTROL-CENTER 92
  Prerequisites 92
  Procedure 92
8.10. CONFIGURING A STATIC ROUTE USING NM-CONNECTION-EDITOR 93
  Prerequisites 93
  Procedure 93
8.11. SETTING THE DEFAULT GATEWAY ON AN EXISTING CONNECTION USING CONTROL-CENTER 94
  Prerequisites 94
  Procedure 94
  Additional resources 95
8.12. SETTING THE DEFAULT GATEWAY ON AN EXISTING CONNECTION USING NM-CONNECTION-EDITOR 95
  Prerequisites 95
  Procedure 95
  Additional resources 96
8.13. CONFIGURING NETWORK BONDING USING NM-CONNECTION-EDITOR

8.13.1. Understanding network bonding

8.13.2. Understanding the default behavior of master and slave interfaces

8.13.3. Comparison of network teaming and bonding features

8.13.4. Configuring a network bond using nm-connection-editor

Prerequisites

Procedure

8.14. CONFIGURING A NETWORK BRIDGE USING NM-CONNECTION-EDITOR

Prerequisites

Procedure

8.15. CONFIGURING A NETWORK TEAM USING NM-CONNECTION-EDITOR

8.15.1. Understanding network teaming

8.15.2. Understanding the default behavior of master and slave interfaces

8.15.3. Comparison of network teaming and bonding features

8.15.4. Understanding the teamd service, runners, and link-watchers

8.15.5. Installing the teamd service

Prerequisites

Procedure

8.15.6. Configuring a network team using nm-connection-editor

Prerequisites

Procedure

Additional resources

8.16. CONFIGURING VLAN TAGGING USING NM-CONNECTION-EDITOR

Prerequisites

Procedure

CHAPTER 9. CONFIGURING IP NETWORKING WITH IFCFG FILES

9.1. CONFIGURING AN INTERFACE WITH STATIC NETWORK SETTINGS USING IFCFG FILES

Prerequisites

Procedure

9.2. CONFIGURING AN INTERFACE WITH DYNAMIC NETWORK SETTINGS USING IFCFG FILES

Prerequisites

Procedure

9.3. MANAGING SYSTEM-WIDE AND PRIVATE CONNECTION PROFILES WITH IFCFG FILES

Prerequisites

Procedure

CHAPTER 10. CONFIGURING MACSEC

10.1. INTRODUCTION TO MACSEC

10.2. USING MACSEC WITH NMCLI TOOL

Prerequisites

Procedure

10.3. USING MACSEC WITH WPA_SUPPLICANT

Procedure

10.4. RELATED INFORMATION

CHAPTER 11. GETTING STARTED WITH IPVLAN

11.1. IPVLAN OVERVIEW

11.2. IPVLAN MODES

11.3. OVERVIEW OF MACVLAN

11.4. COMPARISON OF IPVLAN AND MACVLAN

11.5. CONFIGURING IPVLAN NETWORK

11.5.1. Creating and configuring the IPVLAN device using iproute2
CHAPTER 12. CONFIGURING VIRTUAL ROUTING AND FORWARDING (VRF) ............................. 122
12.1. TEMPORARILY REUSING THE SAME IP ADDRESS ON DIFFERENT INTERFACES 122
  Prerequisites 122
  Procedure 122
12.2. PERMANENTLY REUSING THE SAME IP ADDRESS ON DIFFERENT INTERFACES 123
  Prerequisites 124
  Procedure 124
  Additional resources 124
  Related information 125

CHAPTER 13. SETTING THE ROUTING PROTOCOLS FOR YOUR SYSTEM ........................ 126
13.1. INTRODUCTION TO FRRouting 126
13.2. SETTING UP FRRouting 127
  Prerequisites 127
  Procedure 127
13.3. MODIFYING THE CONFIGURATION OF FRR 127
  Enabling an additional daemon 128
    Prerequisites 128
    Procedure 128
  Disabling a daemon 128
    Prerequisites 128
    Procedure 128
13.4. MODIFYING A CONFIGURATION OF A PARTICULAR DAEMON 128
  Procedure 128

CHAPTER 14. PROVIDING DHCP SERVICES ................................................................. 130
14.1. THE DIFFERENCES WHEN USING DHCPD FOR DHCPV4 AND DHCPV6 130
14.2. THE LEASE DATABASE OF THE DHCPD SERVICE 130
  Additional resources 131
14.3. DYNAMIC IP ADDRESS ASSIGNMENT IN IPV6 NETWORKS 131
  14.3.1. Comparison of DHCPv6 to radvd 131
  14.3.2. Configuring the radvd service for IPv6 routers 131
    Prerequisites 131
    Procedure 131
  Additional resources 132
14.4. SETTING NETWORK INTERFACES FOR THE DHCP SERVERS 132
  Prerequisites 133
  Procedure 133
    For IPv4 networks 133
    For IPv6 networks 133
14.5. SETTING UP THE DHCP SERVICE FOR SUBNETS DIRECTLY CONNECTED TO THE DHCP SERVER 134
  Prerequisites 134
  Procedure 134
    For IPv4 networks 134
    For IPv6 networks 135
  Additional resources 136
14.6. SETTING UP THE DHCP SERVICE FOR SUBNETS THAT ARE NOT DIRECTLY CONNECTED TO THE DHCP SERVER 136
  Prerequisites 137
  Procedure 137
    For IPv4 networks 137
    For IPv6 networks 138
15.8. CONTROLLING NETWORK TRAFFIC USING FIREWALLD

15.8.1. Disabling all traffic in case of emergency using CLI
Procedure
15.8.2. Controlling traffic with predefined services using CLI
Procedure
15.8.3. Controlling traffic with predefined services using GUI

15.8.4. Adding new services
Procedure
15.8.5. Controlling ports using CLI
15.8.5.1. Opening a port
Procedure
15.8.5.2. Closing a port
Procedure
15.8.6. Opening ports using GUI

15.8.7. Controlling traffic with protocols using GUI
15.8.8. Opening source ports using GUI

15.9. WORKING WITH FIREWALLD ZONES

15.9.1. Listing zones
Procedure
15.9.2. Modifying firewalld settings for a certain zone
Procedure
15.9.3. Changing the default zone
Procedure
15.9.4. Assigning a network interface to a zone
Procedure
15.9.5. Assigning a default zone to a network connection
Procedure
15.9.6. Creating a new zone
Procedure
15.9.7. Zone configuration files
Additional resources
15.9.8. Using zone targets to set default behavior for incoming traffic
Procedure

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

15.10.1. Using zones to manage incoming traffic depending on a source

15.10.2. Adding a source
Procedure
15.10.3. Removing a source
Procedure
15.10.4. Adding a source port
Procedure
15.10.5. Removing a source port
Procedure
15.10.6. Using zones and sources to allow a service for only a specific domain
Procedure
15.10.7. Configuring traffic accepted by a zone based on a protocol
15.10.7.1. Adding a protocol to a zone
Procedure
15.10.7.2. Removing a protocol from a zone
Procedure

15.11. CONFIGURING IP ADDRESS MASQUERADING

Procedure

15.12. PORT FORWARDING
15.12.1. Adding a port to redirect
Prerequisites
Procedure

15.12.2. Redirecting TCP port 80 to port 88 on the same machine
Procedure

15.12.3. Removing a redirected port
Procedure

15.12.4. Removing TCP port 80 forwarded to port 88 on the same machine
Procedure

15.13. MANAGING ICMP REQUESTS
15.13.1.Listing and blocking ICMP requests
15.13.2. Configuring the ICMP filter using GUI

15.14. SETTING AND CONTROLLING IP SETS USING FIREWALLD

15.15. PRIORITIZING RICH RULES
15.15.1. How the priority parameter organizes rules into different chains
15.15.2. Setting the priority of a rich rule
Procedure

15.16. CONFIGURING FIREWALL LOCKDOWN
15.16.1. Configuring lockdown with using CLI
15.16.2. Configuring lockdown whitelist options using CLI
15.16.3. Configuring lockdown whitelist options using configuration files

15.17. LOG FOR DENIED PACKETS

15.18. RELATED INFORMATION
Installed documentation
Online documentation

CHAPTER 16. GETTING STARTED WITH NFTABLES
16.1. INTRODUCTION TO NFTABLES
Additional resources

16.2. WHEN TO USE FIREWALLD, NFTABLES, OR IPTABLES

16.3. CONVERTING IPTABLES RULES TO NFTABLES RULES

16.4. WRITING AND EXECUTING NFTABLES SCRIPTS
16.4.1. The required script header in nftables script
16.4.2. Supported nftables script formats
16.4.3. Running nftables scripts
Prerequisites
Procedure
Additional resources

16.4.4. Using comments in nftables scripts

16.4.5. Using variables in an nftables script
Variables with a single value
Variables that contain an anonymous set
Additional resources

16.4.6. Including files in an nftables script
Additional resources

16.4.7. Automatically loading nftables rules when the system boots
Prerequisites
Procedure
Additional resources

16.5. DISPLAYING NFTABLES RULE SETS

16.6. CREATING AN NFTABLES TABLE
16.7. CREATING AN NFTABLES CHAIN
Prerequisites
Procedure
Additional resources

16.8. ADDING A RULE TO AN NFTABLES CHAIN
Prerequisites
Procedure
Additional resources

16.9. INSERTING A RULE INTO AN NFTABLES CHAIN
Prerequisites
Procedure

16.10. USING SETS IN NFTABLES COMMANDS
16.10.1. Using an anonymous sets in nftables
Prerequisites
Procedure
16.10.2. Using named sets in nftables
Prerequisites
Procedure
Related information

16.11. USING VERDICT MAPS IN NFTABLES COMMANDS
16.11.1. Using literal maps in nftables
Procedure
16.11.2. Using mutable verdict maps in nftables
Procedure
Related information

16.12. LIMITING THE NUMBER OF CONNECTIONS USING NFTABLES
Prerequisites
Procedure

16.13. DEBUGGING NFTABLES RULES
16.13.1. Creating a rule with a counter
Prerequisites
Procedure
16.13.2. Adding a counter to an existing rule
Prerequisites
Procedure
16.13.3. Monitoring packets that match an existing rule
Prerequisites
Procedure

16.14. BACKING UP AND RESTORING NFTABLES RULE SETS
16.14.1. Backing up nftables rule sets to a file
Procedure
16.14.2. Restoring nftables rule sets from a file
Procedure

16.15. RELATED INFORMATION

CHAPTER 17. GETTING STARTED WITH DPDK
17.1. INSTALLING THE DPDK PACKAGE
Prerequisites
Procedure
Related information
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.
1.1. IP VERSUS 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 IP 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. STATIC VERSUS 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. 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 5.1, “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.

### 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. 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
```
CHAPTER 2. CONSISTENT NETWORK INTERFACE DEVICE NAMING

Red Hat Enterprise Linux 8 provides methods for consistent and predictable device naming for network interfaces. These features help locating and differentiating network interfaces.

The kernel assigns names to network interfaces by concatenating a fixed prefix and a number that increases as the kernel initialize the network devices. For instance, `eth0` would represent the first device being probed on start-up. However, these names do not necessarily correspond to labels on the chassis. Modern server platforms with multiple network adapters can encounter non-deterministic and counter-intuitive naming of these interfaces. This affects both network adapters embedded on the system board and add-in adapters.

In Red Hat Enterprise Linux 8, the `udev` device manager supports a number of different naming schemes. By default, `udev` assigns fixed names based on firmware, topology, and location information. This has the following advantages:

- Device names are fully predictable.
- Device names stay fixed even if you add or remove hardware, because no re-enumeration takes places.
- Defective hardware can be seamlessly replaced.

2.1. NETWORK INTERFACE DEVICE NAMING HIERARCHY

If consistent device naming is enabled, which is the default in Red Hat Enterprise Linux 8, the `udev` device manager generates device names based on the following schemes:

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Device names incorporate firmware or BIOS-provided index numbers for onboard devices. If this information is not available or applicable, <code>udev</code> uses scheme 2.</td>
<td>eno1</td>
</tr>
<tr>
<td>2</td>
<td>Device names incorporate firmware or BIOS-provided PCI Express (PCIe) hot plug slot index numbers. If this information is not available or applicable, <code>udev</code> uses scheme 3.</td>
<td>ens1</td>
</tr>
<tr>
<td>3</td>
<td>Device names incorporate the physical location of the connector of the hardware. If this information is not available or applicable, <code>udev</code> uses scheme 5.</td>
<td>enp2s0</td>
</tr>
<tr>
<td>4</td>
<td>Device names incorporate the MAC address. Red Hat Enterprise Linux does not use this scheme by default, but administrators can optionally use it.</td>
<td>enx525400d5e0fb</td>
</tr>
<tr>
<td>5</td>
<td>The traditional unpredictable kernel naming scheme. If <code>udev</code> cannot apply any of the other schemes, the device manager uses this scheme.</td>
<td>eth0</td>
</tr>
</tbody>
</table>

By default, Red Hat Enterprise Linux selects the device name based on the `NamePolicy` setting in the `/usr/lib/systemd/network/99-default.link` file. The order of the values in `NamePolicy` is important.
Red Hat Enterprise Linux uses the first device name that is both specified in the file and that udev generated.

If you manually configured udev rules to change the name of kernel devices, those rules take precedence.

2.2. HOW THE NETWORK DEVICE RENAMING WORKS

By default, consistent device naming is enabled in Red Hat Enterprise Linux 8. The udev device manager processes different rules to rename the devices. The following list describes the order in which udev processes these rules and what actions these rules are responsible for:

1. The /usr/lib/udev/rules.d/60-net.rules file defines that the /lib/udev/rename_device helper utility searches for the HWADDR parameter in /etc/sysconfig/network-scripts/ifcfg-* files. If the value set in the variable matches the MAC address of an interface, the helper utility renames the interface to the name set in the DEVICE parameter of the file.

2. The /usr/lib/udev/rules.d/71-biosdevname.rules file defines that the biosdevname utility renames the interface according to its naming policy, provided that it was not renamed in the previous step.

3. The /usr/lib/udev/rules.d/75-net-description.rules file defines that udev examines the network interface device and sets the properties in udev-internal variables, that will be processed in the next step. Note that some of these properties might be undefined.

4. The /usr/lib/udev/rules.d/80-net-setup-link.rules file calls the net_setup_link udev built-in which then applies the policy. The following is the default policy that is stored in the /usr/lib/systemd/network/99-default.link file:

   [Link]
   NamePolicy=kernel database onboard slot path
   MACAddressPolicy=persistent

   With this policy, if the kernel uses a persistent name, udev does not rename the interface. If the kernel does not use a persistent name, udev renames the interface to the name provided by the hardware database of udev. If this database is not available, Red Hat Enterprise Linux falls back to the mechanisms described above.

   Alternatively, set the NamePolicy parameter in this file to mac for media access control (MAC) address-based interface names.

5. The /usr/lib/udev/rules.d/80-net-setup-link.rules file defines that udev renames the interface based on the udev-internal parameters in the following order:

   a. ID_NET_NAME_ONBOARD
   b. ID_NET_NAME_SLOT
   c. ID_NET_NAME_PATH

   If one parameter is not set, udev uses the next one. If none of the parameters are set, the interface is not renamed.

Steps 3 and 4 implement the naming schemes 1 to 4 described in Section 2.1, “Network interface device naming hierarchy”.
Additional resources

- For details about setting custom prefixes for consistent naming, see Section 2.7, "Using prefixdevname for naming of Ethernet network interfaces".

- For details about the NamePolicy parameter, see the systemd.link(5) man page.

2.3. PREDICTABLE NETWORK INTERFACE DEVICE NAMES ON THE X86_64 PLATFORM EXPLAINED

When the consistent network device name feature is enabled, the udev device manager creates the names of devices based on different criteria. This section describes the naming scheme when Red Hat Enterprise Linux 8 is installed on a x86_64 platform.

The interface name starts with a two-character prefix based on the type of interface:

- en for Ethernet
- wl for wireless LAN (WLAN)
- ww for wireless wide area network (WWAN)

Additionally, one of the following is appended to one of the above-mentioned prefix based on the schema the udev device manager applies:

- o<on-board_index_number>
- s<hot_plug_slot_index_number>[f<function>][d<device_id>]
  Note that all multi-function PCI devices have the [f<function>] number in the device name, including the function 0 device.
- x<MAC_address>
- [P<domain_number>][p<bus>s<slot>][f<function>][d<device_id>]
  The [P<domain_number>] part defines the PCI geographical location. This part is only set if the domain number is not 0.
- [P<domain_number>][p<bus>s<slot>][f<function>][u<usb_port>][...][c<config>][i<interface>]
  For USB devices, the full chain of port numbers of hubs is composed. If the name is longer than the maximum (15 characters), the name is not exported. If there are multiple USB devices in the chain, udev suppresses the default values for USB configuration descriptors (c1) and USB interface descriptors (i0).

2.4. PREDICTABLE NETWORK INTERFACE DEVICE NAMES ON THE SYSTEM Z PLATFORM EXPLAINED

When the consistent network device name feature is enabled, the udev device manager on the System z platform creates the names of devices based on the bus ID. The bus ID identifies a device in the s390 channel subsystem.

For a channel command word (CCW) device, the bus ID is the device number with a leading 0.n prefix where n is the subchannel set ID.
Ethernet interfaces are named, for example, `enccw0.0.1234`. Serial Line Internet Protocol (SLIP) channel-to-channel (CTC) network devices are named, for example, `slccw0.0.1234`.

Use the `znetconf -c` or the `lscss -a` commands to display available network devices and their bus IDs.

2.5. DISABLING CONSISTENT INTERFACE DEVICE NAMING DURING THE INSTALLATION

This section describes how to disable consistent interface device naming during the installation.

**WARNING**

Red Hat recommends not to disable consistent device naming. Disabling consistent device naming can cause different kind of problems. For example, if you add another network interface card to the system, the assignment of the kernel device names, such as `eth0`, is no longer fixed. Consequently, after a reboot, the Kernel can name the device differently.

**Procedure**

1. Boot the Red Hat Enterprise Linux 8 installation media.

2. In the boot manager, select `Install Red Hat Enterprise Linux 8`, and press the Tab key to edit the entry.

3. Append the `net.ifnames=0` parameter to the kernel command line:

   ```
   vmlinuz... net.ifnames=0
   ```

4. Press Enter to start the installation.

**Additional resources**

- Is it safe to set `net.ifnames=0` in RHEL 7 and RHEL 8?
- How to perform an in-place upgrade to RHEL 8 when using kernel NIC names on RHEL 7

2.6. DISABLING CONSISTENT INTERFACE DEVICE NAMING ON AN INSTALLED SYSTEM

This section describes how to disable consistent interface device naming on a system that is already installed.
WARNING

Red Hat recommends not to disable consistent device naming. Disabling consistent device naming can cause different kind of problems. For example, if you add another network interface card to the system, the assignment of the kernel device names, such as `eth0`, is no longer fixed. Consequently, after a reboot, the Kernel can name the device differently.

Prerequisites

- The system uses consistent interface device naming, which is the default.

Procedure

1. Edit the `/etc/default/grub` file and append the `net.ifnames=0` parameter to the `GRUB_CMDLINE_LINUX` variable:

   ```
   GRUB_CMDLINE_LINUX="... net.ifnames=0
   ```

2. Rebuild the `grub.cfg` file:
   - On a system with UEFI boot mode:
     ```
     # grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
     ```
   - On a system with legacy boot mode:
     ```
     # grub2-mkconfig -o /boot/grub2/grub.cfg
     ```

3. If you use interface names in configuration files or scripts, you must manually update them.

4. Reboot the host:

   ```
   # reboot
   ```

2.7. USING PREFIXDEVNAME FOR NAMING OF ETHERNET NETWORK INTERFACES

This documentation describes how to set the prefixes for consistent naming of Ethernet network interfaces in case that you do not want to use the default naming scheme of such interfaces.

However, Red Hat recommends to use the default naming scheme, which is the same as in Red Hat Enterprise Linux 7. For more details about this scheme, see Consistent Network Device Naming.

2.7.1. Introduction to prefixdevname

The `prefixdevname` tool is a udev helper utility that enables you to define your own prefix used for naming of the Ethernet network interfaces.
2.7.2. Setting prefixdevname

The setting of the prefix with `prefixdevname` is done during system installation.

To set and activate the required prefix for your Ethernet network interfaces, add the following on the kernel command line:

```
net.ifnames.prefix=<required prefix>
```

**WARNING**

Red Hat does not support the use of `prefixdevname` on already deployed systems.

After the prefix was once set, and the operating system was rebooted, the prefix is effective every time when a new network interface appears. The new device is assigned a name in the form of `<PREFIX><INDEX>`. For example, if your selected prefix is `net`, and the interfaces with `net0` and `net1` prefixes already exist on the system, the new interface is named `net2`. The `prefixdevname` utility then generates the new `.link` file in the `/etc/systemd/network` directory that applies the name to the interface with the MAC address that just appeared. The configuration is persistent across reboots.

2.7.3. Limitations of prefixdevname

There are certain limitations for prefixes of Ethernet network interfaces.

The prefix that you choose must meet the following requirements:

- be ASCII string
- be alphanumeric string
- be shorter than 16 characters

**WARNING**

The prefix cannot conflict with any other well-known prefix used for network interface naming on Linux. Specifically, you cannot use these prefixes: `eth`, `eno`, `ens`, `em`.

Related information

- See the `udev(7)` man page for details about the `udev` device manager.
CHAPTER 3. NETCONSOLE

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

3.1. CONFIGURING NETCONSOLE

This procedure describes how you can configure netconsole in Red Hat Enterprise Linux (RHEL) 8.

Prerequisites
The netconsole-service package is installed.

```
~]$ yum install netconsole-service
```

Procedure

1. Set the SYSLOGADDR to the IP address of the syslogd server 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.

```
~]$ 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
```

Additional resources
How to configure netconsole under Red Hat Enterprise Linux 8?
CHAPTER 4. GETTING STARTED WITH MANAGING NETWORKING WITH NETWORKMANAGER

4.1. 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.

4.1.1. Benefits of using NetworkManager

The main benefits of using NetworkManager are:

- Offering an API through D-Bus which allows to query and control network configuration and state. In this way, networking can be checked and configured by multiple applications ensuring a synced and up-to-date networking status. For example, the RHEL web console, which monitors and configures servers through a web browser, uses the NetworkManager D-BUS interface to configure networking, as well as the Gnome GUI, the nmcli and the nm-connection-editor tools. Each change made in one of these tools is detected by all the others.

- 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.

- 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 4.5, "NetworkManager tools"

- For more information on installing and using the RHEL 8 web console, see Managing systems using the RHEL 8 web console.

4.2. INSTALLING NETWORKMANAGER

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

To start NetworkManager:

```
~$ systemctl start NetworkManager
```

To enable NetworkManager automatically at boot time:

```
~$ systemctl enable NetworkManager
```

### 4.5. NETWORKMANAGER TOOLS

<table>
<thead>
<tr>
<th>Application or Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nmcli</td>
<td>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.</td>
</tr>
<tr>
<td>nmtui</td>
<td>A simple curses-based text user interface (TUI) for NetworkManager.</td>
</tr>
<tr>
<td>nm-connection-editor</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.

4.6. 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

- Chapter 6, Configuring IP networking with nmtui
- Chapter 7, Configuring networking with nmcli
- Chapter 8, Configuring networking with GNOME GUI

4.7. 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.

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:
```bash
~$ nmcli connection reload
```

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

```bash
~$ 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:

```bash
~$ nmcli con up connection-name
```

4.7.1. 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:

```bash
~$ 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 5. OVERVIEW OF NETWORK CONFIGURATION METHODS

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

5.1. 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.

- To configure the network settings when the root filesystem is **not** local:
  - use the kernel command-line.

**Additional resources**

- Chapter 6, *Configuring IP networking with nmtui*
- Chapter 7, *Configuring networking with nmcli*
- Chapter 8, *Configuring networking with GNOME GUI*
CHAPTER 6. CONFIGURING IP NETWORKING WITH NMTUI

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

6.1. 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 4.3, “Checking the status of NetworkManager”

Procedure

1. Start the nmtui tool:

  ```
  ~]$ nmtui
  ```

   The text user interface appears.

   Figure 6.1. 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.

6.1.1. Editing a connection with nmtui
Prerequisites

- Section 6.1, “Getting started with nmtui”

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

6.1.2. 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 6.1, “Getting started with nmtui”

Procedure

1. Select the Activate a connection menu entry.

![Figure 6.2. Activating a connection with nmtui](image)

2. Select the modified connection. On the right, click the Deactivate button.
3. Choose the connection again and click the **Activate** button.
The following commands are also available:

\[
\sim\$ \texttt{nmtui edit }\textit{connection-name}
\]

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.

\[
\sim\$ \texttt{nmtui connect }\textit{connection-name}
\]

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 4.5, “NetworkManager tools”
CHAPTER 7. CONFIGURING NETWORKING WITH NMCLI

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

7.1. 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.

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 7.4, “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.

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 7.1. 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 7.2. 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


CHAPTER 7. CONFIGURING NETWORKING WITH NMCLI
This option causes `nmcli` to produce human-readable output. For example, values are aligned and headers are printed.

Example 7.3. 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 4.5, “NetworkManager tools”
- the `nmcli(1)` man page.
- Section 7.3, “Brief selection of `nmcli` commands”
- Section 7.5, “Creating a connection profile with `nmcli`”

7.2. 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.

<table>
<thead>
<tr>
<th>Alias</th>
<th>Example</th>
<th>Property</th>
<th>Example</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>type bond</td>
<td>connection.type</td>
<td>connection.type bond</td>
<td>type of a specific connection. Some of the connection types are: bond, bridge, ethernet, wifi, infiniband, vlan</td>
</tr>
<tr>
<td>ifname</td>
<td>ifname eth0</td>
<td>connection.interface-name</td>
<td>connection.interface-name eth0</td>
<td>name of the device to which a connection belongs to</td>
</tr>
</tbody>
</table>
7.3. 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 7.4. 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 7.5. Viewing only currently active connections**

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

**Example 7.6. 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    --
```
Example 7.7. 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 bond0
```

Example 7.8. 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 7.9. 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 7.10. Viewing general information for a device

```
~
$ nmcli device show
GENERAL.DEVICE:      ens3
GENERAL.TYPE:        ethernet
```
Example 7.11. 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 7.12. 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>
<thead>
<tr>
<th>nmcli command</th>
<th>abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>nmcli general status</td>
<td>nmcli g</td>
</tr>
<tr>
<td>nmcli general logging</td>
<td>nmcli g log</td>
</tr>
<tr>
<td>nmcli connection show</td>
<td>nmcli con show or nmcli c</td>
</tr>
<tr>
<td>nmcli connection show --active</td>
<td>nmcli con show -a or nmcli c -a</td>
</tr>
<tr>
<td>nmcli device status</td>
<td>nmcli dev or nmcli d</td>
</tr>
<tr>
<td>nmcli device show</td>
<td>nmcli dev show or nmcli d show</td>
</tr>
</tbody>
</table>

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 7.5, “Creating a connection profile with nmcli”

7.4. SETTING A DEVICE MANAGED OR UNMANAGED WITH NMCLI

Prerequisites

- Section 7.1, “Getting started with nmcli”
- Section 7.2, “Overview of nmcli property names and aliases”

Procedure

1. 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     --
```

2. 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.

### 7.5. CREATING A CONNECTION PROFILE WITH NMCLI

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

**Prerequisites**

- Section 7.1, “Getting started with nmcli”
- Section 7.2, “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
```

**Procedure**

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

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

- a. where `{COMMON_OPTIONS}` are the aliases or property names, see Aliases and Property names.
- b. where `[IP_OPTIONS]` are the IP addresses:
For IPv4 addresses: \texttt{ip4}

For IPv6 addresses: \texttt{ip6}

c. where \texttt{[NETMASK]} is the network mask width. For example, \texttt{255.255.255.0} is the network mask for the prefix \texttt{198.51.100.0/24}.

d. where \texttt{[GATEWAY]} is the gateway information:

- For IPv4 addresses: \texttt{gw4}
- For IPv6 addresses: \texttt{gw6}

\texttt{nmcli} \texttt{connection add type ethernet con-name connection-name ifname interface-name ip4 address/network mask gw4 address}

1. To create a connection profile with an IPv4 address:

\texttt{~\$ 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.

2. To activate the created connection:

\texttt{~\$ nmcli c up _"My Connection"}

3. To view the created connection:

\texttt{~\$ nmcli c show "My Connection"}

Note that the \texttt{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, \texttt{nmcli} generates a pager to allow an easy navigation on the output. In the pager, use arrows to move up and down and the \texttt{q} key to quit.

For a more compact display of the connection, use the \texttt{-o} option:

\texttt{~\$ nmcli -o c show "My Connection"}

The \texttt{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 \texttt{nm-settings(5)} man page for more information on properties and their settings.

### 7.6. USING THE NMCLI INTERACTIVE CONNECTION EDITOR

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

\texttt{~\$ nmcli con edit}
You should enter a valid **connection type** from the list displayed. Then, you are able to modify its parameters.

```sh
~$ 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 7.13. Adding a new Ethernet connection using the nmcli interactive connection editor**

```sh
~$ 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
```
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:

```bash
~]# 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
```

### 7.7. MODIFYING A CONNECTION PROFILE WITH NMCLI

You can modify the existing configuration of a connection profile.

**Prerequisites**

- Section 7.1, “Getting started with nmcli”
- Section 7.2, “Overview of nmcli property names and aliases”
- Section 7.5, “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:

```bash
~]# 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:

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

**Procedure**
1. 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
```

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

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

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

### 7.8. Configuring a Static Ethernet Connection

#### 7.8.1. Configuring a static Ethernet connection with `nmcli`

**Prerequisites**

- [Section 7.1, “Getting started with `nmcli`”](#)
- [Section 7.5, “Creating a connection profile with `nmcli`”](#)

**Procedure**

1. 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.

2. 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"
```

3. To activate the new Ethernet connection:
4. 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   --

5. 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.

### 7.8.2. 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-fa65a3dbcc916) 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.

7.9. CONFIGURING A DYNAMIC ETHERNET CONNECTION

7.9.1. Configuring a dynamic Ethernet connection with nmcli

Prerequisites
- Section 7.1, “Getting started with nmcli”
- Section 7.5, “Creating a connection profile with nmcli”

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

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

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

   ```bash
   ~]$ 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.

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

   ```bash
   ```

7.9.2. Configuring a dynamic Ethernet connection using the interactive editor

To configure a dynamic Ethernet connection using the interactive editor:

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

==| nmcli interactive connection editor ==

Adding a new '802-3-ethernet' connection
```
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.

7.10. HOW TO USE THE NMCLI COMMAND TO CONFIGURE A STATIC ROUTE

To configure a static route, use the nmcli utility with the following syntax:

$ nmcli connection modify connection_name ipv4.routes "ip[/prefix] [next_hop] [metric] [attribute=value] [attribute=value] ..."

The command supports the following route attributes:

- table=n
- src=address
- tos=n
- onlink=true|false
- window=n
- cwnd=n
If you use the `ipv4.routes` sub-command, `nmcli` overrides all current settings of this parameter. To add an additional route, use the `nmcli connection modify connection_name +ipv4.routes "..."` command. In a similar way, you can use `nmcli connection modify connection_name -ipv4.routes "..."` to remove a specific route.

### 7.11. Configuring a Static Route Using a NMCLI Command

You can add a static route to the configuration of a network connection using the `nmcli connection modify` command.

The procedure in this section describes how to add a route to the `192.0.2.0/24` network that uses the gateway running on `198.51.100.1`, which is reachable through the `example` connection.

**Prerequisites**

- The network is configured
- The gateway for the static route must be directly reachable on the interface.
- If the user is logged in on a physical console, user permissions are sufficient. Otherwise, the command requires `root` permissions.

**Procedure**

1. Add the static route to the `example` connection:

   ```bash
   $ sudo nmcli connection modify example +ipv4.routes "192.0.2.0/24 198.51.100.1"
   ```

   To set multiple routes in one step, pass the individual routes comma-separated to the command. For example, to add a route to the `192.0.2.0/24` and `203.0.113.0/24` networks, both routed through the `198.51.100.1` gateway, enter:

   ```bash
   $ sudo nmcli connection modify example +ipv4.routes "192.0.2.0/24 198.51.100.1,
   203.0.113.0/24 198.51.100.1"
   ```

2. Optionally, verify that the routes were added correctly to the configuration:

   ```bash
   $ nmcli connection show example
   ...
   ipv4.routes:   { ip = 192.0.2.1/24, nh = 198.51.100.1 }
   ...
   ```

3. Restart the network connection:

   ```bash
   $ sudo nmcli connection up example
   ```
4. Optionally, verify that the route is active:

```bash
$ ip route
... 192.0.2.0/24 via 198.51.100.1 dev example proto static metric 100
```

**Additional resources**
- For further details about **nmcli**, see the **nmcli(1)** man page.

## 7.12. Configuring a Static Route Using the NMCLI Interactive Mode

You can use the interactive mode of the **nmcli** utility to add a static route to the configuration of a network connection.

The procedure in this section describes how to add a route to the **192.0.2.0/24** network that uses the gateway running on **198.51.100.1**, which is reachable through the **example** connection.

**Prerequisites**
- The network is configured
- The gateway for the static route must be directly reachable on the interface.
- If the user is logged in on a physical console, user permissions are sufficient. Otherwise, the command requires **root** permissions.

**Procedure**

1. Open the **nmcli** interactive mode for the **example** connection:

   ```bash
   $ sudo nmcli connection edit example
   ```

2. Add the static route:

   ```bash
   nmcli> set ipv4.routes 192.0.2.0/24 198.51.100.1
   ```

3. Optionally, verify that the routes were added correctly to the configuration:

   ```bash
   nmcli> print
   ...
   ipv4.routes: { ip = 192.0.2.1/24, nh = 198.51.100.1 }
   ...
   ```

   The **ip** attribute displays the network to route and the **nh** attribute the gateway (next hop).
4. Save the configuration:

```
nmcli> save persistent
```

5. Restart the network connection:

```
nmcli> activate example
```

![WARNING]

When you restart the connection, all connections currently using this connection will be temporarily interrupted.

6. Leave the `nmcli` interactive mode:

```
nmcli> quit
```

7. Optionally, verify that the route is active:

```
$ ip route
... 192.0.2.0/24 via 198.51.100.1 dev example proto static metric 100
```

Additional resources

- For the list of commands available in the interactive mode, enter `help` in the interactive shell.

### 7.13. Setting the Default Gateway on an Existing Connection Using NMCLI

In most situations, administrators set the default gateway when they create a connection as explained in, for example, Section 7.8.1, “Configuring a static Ethernet connection with nmcli”.

This section describes how to set or update the default gateway on a previously created connection using the `nmcli` utility.

**Prerequisites**

- At least one static IP address must be configured on the connection on which the default gateway will be set.

- If the user is logged in on a physical console, user permissions are sufficient. Otherwise, user must have `root` permissions.

**Procedure**

1. Set the IP address of the default gateway.
   
   For example, to set the IPv4 address of the default gateway on the `example` connection to `192.0.2.1`:
For example, to set the IPv6 address of the default gateway on the `example` connection to `2001:db8::1`:

```
$ sudo nmcli connection modify example ipv6.gateway "2001:db8::1"
```

2. Restart the network connection for changes to take effect. For example, to restart the `example` connection using the command line:

```
$ sudo nmcli connection up example
```

3. Optionally, verify that the route is active. To display the IPv4 default gateway:

```
$ ip -4 route
default via 192.0.2.1 dev example proto static metric 100
```

To display the IPv6 default gateway:

```
$ ip -6 route
default via 2001:db8::1 dev example proto static metric 100 pref medium
```

**Additional resources**

- Section 7.8.1, “Configuring a static Ethernet connection with nmcli”

### 7.14. SETTING THE DEFAULT GATEWAY ON AN EXISTING CONNECTION USING THE NMCLI INTERACTIVE MODE

In most situations, administrators set the default gateway when they create a connection as explained in, for example, Section 7.8.2, “Configuring a static Ethernet connection using the nmcli interactive editor”.

This section describes how to set or update the default gateway on a previously created connection using the interactive mode of the `nmcli` utility.

**Prerequisites**

- At least one static IP address must be configured on the connection on which the default gateway will be set.

- If the user is logged in on a physical console, user permissions are sufficient. Otherwise, the user must have `root` permissions.
Procedure

1. Open the **nmcli** interactive mode for the required connection. For example, to open the **nmcli** interactive mode for the *example* connection:

   ```
   $ sudo nmcli connection edit example
   ```

2. Set the default gateway.
   For example, to set the IPv4 address of the default gateway on the *example* connection to 192.0.2.1:

   ```
   nmcli> set ipv4.gateway 192.0.2.1
   ```

   For example, to set the IPv6 address of the default gateway on the *example* connection to 2001:db8::1:

   ```
   nmcli> set ipv6.gateway 2001:db8::1
   ```

3. Optionally, verify that the default gateway was set correctly:

   ```
   nmcli> print
   ...
   ipv4.gateway: 192.0.2.1
   ...
   ipv6.gateway: 2001:db8::1
   ```

4. Save the configuration:

   ```
   nmcli> save persistent
   ```

5. Restart the network connection for changes to take effect:

   ```
   nmcli> activate example
   ```

   **WARNING**
   All connections currently using this network connection are temporarily interrupted during the restart.

6. Leave the **nmcli** interactive mode:

   ```
   nmcli> quit
   ```

7. Optionally, verify that the route is active.
   To display the IPv4 default gateway:
$ ip -4 route
default via 192.0.2.1 dev example proto static metric 100

To display the IPv6 default gateway:

$ ip -6 route
default via 2001:db8::1 dev example proto static metric 100 pref medium

Additional resources
- Section 7.8.2, "Configuring a static Ethernet connection using the nmcli interactive editor"

7.15. CONNECTING A WI-FI CONNECTION WITH NMCLI

This procedure describes how to connect to a wireless connection using the nmcli utility.

Prerequisites
- The nmcli utility to be installed.
- Make sure that the WiFi radio is on (default):

  ~]$ nmcli radio wifi on

Procedure

1. To refresh the available Wi-Fi connection list:

  ~]$ nmcli device wifi rescan

2. To view the available Wi-Fi access points:

  ~]$ nmcli dev wifi list

  IN-USE  SSID      MODE   CHAN  RATE        SIGNAL  BARS  SECURITY
  ...  MyCafe    Infra  3     405 Mbit/s  85▂▄▆█ WPA1 WPA2

3. To connect to a Wi-Fi connection using nmcli:

  ~]$ `nmcli dev wifi connect _SSID-Name_ password _wireless-password_`

For example:

  ~]$ nmcli dev wifi connect MyCafe password wireless-password

Note that if you want to disable the Wi-Fi state:

  ~]$ nmcli radio wifi off

7.15.1. Configuring a Wi-Fi connection using nmcli
Prerequisites

- The `nmcli` utility to be installed.
- Make sure that the WiFi radio is on (default):

```
~$ nmcli radio wifi on
```

Procedure

1. To create a Wi-Fi connection profile with static IP configuration:

```
~$ nmcli con add con-name MyCafe ifname wlan0 type wifi ssid MyCafe ` `ip4 192.168.100.101/24 gw4 192.168.100.1
```

2. To check a specific property, for example `mtu`:

```
~$ nmcli connection show id MyCafe | grep mtu
802-11-wireless.mtu: auto
```

3. To change the property of a setting:

```
~$ nmcli connection modify id MyCafe 802-11-wireless.mtu 1350
```

4. To verify the change:

```
~$ nmcli connection show id MyCafe | grep mtu
802-11-wireless.mtu: 1350
```

Additional resources

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

7.15.2. Connecting to a hidden network using nmcli

All access points have a Service Set Identifier (SSID) to identify them. However, an access point may be configured not to broadcast its SSID, in which case it is hidden, and will not show up in NetworkManager’s list of Available networks.

This procedure shows how you can connect to a hidden network using the `nmcli` tool.

Prerequisites

- The `nmcli` utility to be installed.
- To know the SSID, and password of the Wi-Fi connection.
- Make sure that the WiFi radio is on (default):

```
~$ nmcli radio wifi on
```

Procedure

Connect to the SSID that is hidden:
You can configure the 802.3 link settings of an Ethernet connection by modifying the following configuration parameters:

- **802-3-ethernet.auto-negotiate**
- **802-3-ethernet.speed**
- **802-3-ethernet.duplex**

You can configure the 802.3 link settings to the following main modes:

- Ignore link negotiation
- Enforce the auto-negotiation activation
- Manually set the `speed` and `duplex` link settings

### 7.17. CONFIGURING 802.3 LINK SETTINGS WITH NMCLI TOOL

This procedure describes how to configure 802.3 link settings using the `nmcli` tool.

**Prerequisites**

- The **NetworkManager** must be installed and running.

**Procedure**

1. To ignore link negotiation, set the following parameters:

   ```bash
   $ nmcli connection modify connection_name 802-3-ethernet.auto-negotiate no 802-3-ethernet.speed 0 802-3-ethernet.duplex ""
   ```

   Note, that the auto-negotiation parameter is not disabled even if the speed and duplex parameters are not set and the auto-negotiation parameter is set to no.

2. To enforce the auto-negotiation activation, enter the following command:

   ```bash
   $ nmcli connection modify connection_name 802-3-ethernet.auto-negotiate yes 802-3-ethernet.speed 0 802-3-ethernet.duplex ""
   ```

   That allows to negotiate all the available speed and duplex modes supported by the NIC.

You can also enable auto-negotiation while advertising and allowing only one speed/duplex mode. This can be useful if you want to enforce **1000BASE-T** and **10GBASE-T** Ethernet link configuration, as these standards mandate auto-negotiation enabled. To enforce **1000BASE-T** standard:

   ```bash
   $ nmcli connection modify connection_name 802-3-ethernet.auto-negotiate yes 802-3-ethernet.speed 1000 802-3-ethernet.duplex full
   ```
3. To manually set the speed and duplex link settings, enter the following command:

```
~]# nmcli connection modify connection_name 802-3-ethernet.auto-negotiate no 802-3-ethernet.speed [speed in Mbit/s] 802-3-ethernet.duplex [full|half]
```

### 7.18. CONFIGURING 802.1X SECURITY FOR WI-FI AND WIRED WITH NMCLI

This procedure describes how to set the network security settings in a **wireless** or a **Wired** connection using the `nmcli` utility.

**Prerequisites**

- The `nmcli` utility is installed.

**Procedure**

1. For a **wireless** connection, set the authenticated **key-mgmt** (key management) protocol. It configures the keying mechanism for a secure **wifi** connection.

2. Configure the 802-1x authentication settings.

**Table 7.2. The 802-1x authentication settings**

<table>
<thead>
<tr>
<th>802-1x authentication setting</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>802-1x.identity</td>
<td>Identity</td>
</tr>
<tr>
<td>802-1x.ca-cert</td>
<td>CA certificate</td>
</tr>
<tr>
<td>802-1x.client-cert</td>
<td>User certificate</td>
</tr>
<tr>
<td>802-1x.private-key</td>
<td>Private key</td>
</tr>
<tr>
<td>802-1x.private-key-password</td>
<td>Private key password</td>
</tr>
</tbody>
</table>

For example, to configure WPA2 Enterprise using the EAP-TLS authentication method, apply the following settings:

```
~]$ nmcli c add type wifi ifname wlan0 con-name 'My Wifi Network'
  802-11-wireless.ssid 'My Wifi'
  802-11-wireless-security.key-mgmt wpa-eap
  802-1x.eap tls
  802-1x.identity identity@example.com
  802-1x.ca-cert /etc/pki/my-wifi/ca.crt
  802-1x.client-cert /etc/pki/my-wifi/client.crt
  802-1x.private-key /etc/pki/my-wifi/client.key
  802-1x.private-key-password s3cr3t
```
NOTE

To configure a **wired** connection using the `nmcli` tool, follow the same procedure as for a **wireless** connection, except the `802-11-wireless.ssid` and `802-11-wireless-security key-mgmt` settings.

7.19. CONFIGURING NETWORK BONDING USING NMCLI

This section describes the basics of network bonding, the differences between bonding and teaming, and how to configure a network bond on Red Hat Enterprise Linux 8 using the `nmcli` utility.

7.19.1. Understanding network bonding

Network bonding is a method to combine or aggregate network interfaces to provide a logical interface with higher throughput or redundancy.

The **active-backup**, **balance-tlb**, and **balance-alb** modes do not require any specific configuration of the network switch. However, other bonding modes require configuring the switch to aggregate the links. For example, Cisco switches requires **EtherChannel** for modes 0, 2, and 3, but for mode 4, the Link Aggregation Control Protocol (LACP) and **EtherChannel** are required.

For further details, see the documentation of your switch and https://www.kernel.org/doc/Documentation/networking/bonding.txt.

**IMPORTANT**

Certain network bonding features, such as the fail-over mechanism, do not support direct cable connections without a network switch. For further details, see the Is bonding supported with direct connection using crossover cables? KCS solution.

7.19.2. Understanding the default behavior of master and slave interfaces

Consider the following default behavior of, when managing or troubleshooting team or bond port interfaces using the **NetworkManager** service:

- Starting the master interface does not automatically start the port interfaces.
- Starting a port interface always starts the master interface.
- Stopping the master interface also stops the port interface.
- A master without ports can start static IP connections.
- A master without ports waits for ports when starting DHCP connections.
- A master with a DHCP connection waiting for ports completes when you add a port with a carrier.
- A master with a DHCP connection waiting for ports continues waiting when you add a port without carrier.

7.19.3. Comparison of network teaming and bonding features

The following table compares features supported in network teams and network bonds:
<table>
<thead>
<tr>
<th>Feature</th>
<th>Network bond</th>
<th>Network team</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcast Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Round-robin Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Active-backup Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LACP (802.3ad) support</td>
<td>Yes (active only)</td>
<td>Yes</td>
</tr>
<tr>
<td>Hash-based Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User can set hash function</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Tx load-balancing support (TLB)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LACP hash port select</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Load-balancing for LACP support</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ethtool link monitoring</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ARP link monitoring</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NS/NA (IPv6) link monitoring</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ports up/down delays</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Port priorities and stickiness (&quot;primary&quot; option enhancement)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Separate per-port link monitoring setup</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Multiple link monitoring setup</td>
<td>Limited</td>
<td>Yes</td>
</tr>
<tr>
<td>Lockless Tx/Rx path</td>
<td>No (rwlock)</td>
<td>Yes (RCU)</td>
</tr>
<tr>
<td>VLAN support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User-space runtime control</td>
<td>Limited</td>
<td>Yes</td>
</tr>
<tr>
<td>Logic in user-space</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Extensibility</td>
<td>Hard</td>
<td>Easy</td>
</tr>
<tr>
<td>Modular design</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
7.19.4. Configuring a network bond using nmcli commands

This section describes how to configure a network bond using `nmcli` commands.

**Prerequisites**

- Two or more network cards are installed in the server.
- The network cards are connected to a switch.

**Procedure**

1. Create a bond interface. For example, to create a bond interface that uses the `active-backup` mode and both the interface and the connection are named `bond0`, enter:

   ```bash
   # nmcli connection add type bond con-name bond0 ifname bond0 bond.options "mode=active-backup"
   ```

   To additionally set a Media Independent Interface (MII) monitoring interval, add the `miimon=INTERVAL` option to the `bond.options` property. For example, to use the same command but, additionally, set the MII monitoring interval to 1000 milliseconds (1 second), enter:

   ```bash
   # nmcli connection add type bond con-name bond0 ifname bond0 bond.options "mode=active-backup,miimon=1000"
   ```

2. Configure the IPv4 settings. For example, to set a static IPv4 address, network mask, default gateway, and DNS server to the `bond0` connection, enter:

   ```bash
   # nmcli connection modify bond0 ipv4.addresses '192.0.2.1/24'
   # nmcli connection modify bond0 ipv4.gateway '192.0.2.254'
   # nmcli connection modify bond0 ipv4.dns '192.0.2.253'
   # nmcli connection modify bond0 ipv4.method manual
   ```

3. Configure the IPv6 settings. For example, to set a static IPv6 address, network mask, default gateway, and DNS server of the `bond0` connection, enter:

   ```bash
   # nmcli connection modify bond0 ipv6.addresses '2001:db8::1/32'
   # nmcli connection modify bond0 ipv6.gateway '2001:db8::fffe'
   # nmcli connection modify bond0 ipv6.dns '2001:db8::fff8'
   # nmcli connection modify bond0 ipv6.method manual
   ```
4. Optionally, display the network interfaces, and note names of interfaces you plan to add to the bond:

```
# nmcli device
DEVICE  TYPE      STATE         CONNECTION
enp1s0  ethernet  connected     enp1s0
enp7s0  ethernet  disconnected  --
enp8s0  ethernet  disconnected  --
lo      loopback  unmanaged     --
```

5. Assign port interfaces to the bond’s connection. For example, to add the interfaces named `enp7s0` and `enp8s0` to the `bond0` connection:

```
# nmcli connection add type ethernet slave-type bond con-name bond0-port1 ifname enp7s0 master bond0
# nmcli connection add type ethernet slave-type bond con-name bond0-port2 ifname enp8s0 master bond0
```

6. Activate the connection. For example, to activate the `bond0` connection:

```
# nmcli connection up bond0
```

7. Verify that the slave devices are connected, and the `CONNECTION` column displays the slave’s connection name:

```
# nmcli device
DEVICE   TYPE      STATE      CONNECTION
...      
enp7s0   ethernet  connected  bond0-port1
enp8s0   ethernet  connected  bond0-port2
```

Red Hat Enterprise Linux activates master and slave devices when the system boots. By activating any slave connection, the master is also activated. However, in this case, only one slave connection is activated. By default, activating the master does not automatically activate the slaves. However, you can enable this behavior by setting:

a. Enable the `connection.autoconnect-slaves` parameter of the bond’s connection:

```
# nmcli connection modify bond0 connection.autoconnect-slaves 1
```

b. Reactivate the bridge:

```
# nmcli connection up bond0
```

8. Optionally, display the status of the bond:

```
# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)
Bonding Mode: fault-tolerance (active-backup)
Primary Slave: None
Currently Active Slave: enp7s0
MII Status: up
MII Polling Interval (ms): 100
```
In the example, both ports are up.

Additional resources

- For `nmcli` examples, see the `nmcli-examples(7)` man page.
- For a list of options you can set in the `bond.options` parameter of the `nmcli` command when you create a bond, see https://www.kernel.org/doc/Documentation/networking/bonding.txt.

### 7.20. CONFIGURING A NETWORK BRIDGE USING NMCLI COMMANDS

This section explains how to configure a network bridge using the `nmcli` utility.

A network bridge is a link-layer device which forwards traffic between networks based on MAC addresses. The bridge device decides on forwarding packages based on a table of MAC addresses. The bridge builds the MAC addresses table by listening to network traffic and thereby learning what hosts are connected to each network.

For example, you can use a software bridge on a Red Hat Enterprise Linux 8 host:

- To emulate a hardware bridge
- In virtualization environments, to integrate virtual machines (VM) to the same network as the host

Due to the IEEE 802.11 standard which specifies the use of 3-address frames in Wi-Fi for the efficient use of airtime, you cannot configure a bridge over Wi-Fi networks operating in Ad-Hoc or Infrastructure modes.

**Prerequisites**

- Two or more physical or virtual network devices are installed in the server.
- You are logged in as the `root` user.

**Procedure**

1. Create a bridge interface. For example, to create the bridge interface named `bridge0`, enter:
# nmcli connection add type bridge con-name bridge0 ifname bridge0

2. Configure the IPv4 settings. For example, to set a static IPv4 address, network mask, default gateway, and DNS server of the bridge0 connection, enter:

   # nmcli connection modify bridge0 ipv4.addresses '192.0.2.1/24'
   # nmcli connection modify bridge0 ipv4.gateway '192.0.2.254'
   # nmcli connection modify bridge0 ipv4.dns '192.0.2.253'
   # nmcli connection modify bridge0 ipv4.method manual

3. Configure the IPv6 settings. For example, to set a static IPv6 address, network mask, default gateway, and DNS server of the bridge0 connection, enter:

   # nmcli connection modify bridge0 ipv6.addresses '2001:db8::1/32'
   # nmcli connection modify bridge0 ipv6.gateway '2001:db8::fffe'
   # nmcli connection modify bridge0 ipv6.dns '2001:db8::fffd'
   # nmcli connection modify bridge0 ipv6.method manual

4. Optionally, configure further properties of the bridge. For example, to set the Spanning Tree Protocol (STP) priority of bridge0 to 16384, enter:

   # nmcli connection modify bridge0 bridge.priority '16384'

   By default, STP is enabled.

5. Optionally, display the network interfaces, and note the names of the interfaces you want to add to the bridge as a slave in the next step:

   # nmcli device

   DEVICE  TYPE      STATE         CONNECTION
   enp1s0  ethernet  connected     enp1s0
   enp7s0  ethernet  disconnected  --
   enp8s0  ethernet  disconnected  --
   lo      loopback  unmanaged     --

6. Assign the port interfaces to the bridge’s connection. For example, to add the interfaces named enp7s0 and enp8s0 to the bridge0 connection, enter:

   # nmcli connection add type ethernet slave-type bridge con-name bridge0-port1 ifname enp7s0 master bridge0
   # nmcli connection add type ethernet slave-type bridge con-name bridge0-port2 ifname enp8s0 master bridge0

7. Activate the connection. For example, to activate the bridge0 connection, enter:

   # nmcli connection up bridge0

8. Verify that the slave devices are connected, and the CONNECTION column displays the slave’s connection name:

   # nmcli device

   DEVICE  TYPE      STATE      CONNECTION
   ...
Red Hat Enterprise Linux activates master and slave devices when the system boots. By activating any slave connection, the master is also activated. However, in this case, only one slave connection is activated. By default, activating the master does not automatically activate the slaves. However, you can enable this behavior by setting:

a. Enable the connection.autoconnect-slaves parameter of the bridge connection:

```
# nmcli connection modify bridge0 connection.autoconnect-slaves 1
```

b. Reactivate the bridge:

```
# nmcli connection up bridge0
```

9. Optionally, use the following command to display the status of the bridge:

```
# bridge link show bridge0
```

For details about the bridge utility, see the bridge(8) man page.

Additional resources

- For nmcli examples, see the nmcli-examples(7) man page.
- For all bridge properties you can set, see the bridge settings section in the nm-settings(5) man page.
- For all bridge port properties you can set, see the bridge-port settings section in the nm-settings(5) man page.
- For details about the bridge utility, see the bridge(8) man page.

7.21. CONFIGURING NETWORK TEAMING USING NMCLI

This section describes the basics of network teaming, the differences between bonding and teaming, and how to configure a network team on Red Hat Enterprise Linux 8 using the nmcli utility.

Prerequisites

- Red Hat Enterprise Linux 8 is installed.
- The system has an active subscription assigned.

7.21.1. Understanding network teaming

Network teaming is a feature that combines or aggregates network interfaces to provide a logical interface with higher throughput or redundancy.

Network teaming uses a kernel driver to implement fast handling of packet flows, as well as user-space libraries and services for other tasks. This way, network teaming is an easily extensible and scalable solution for load-balancing and redundancy requirements.
Note that in the context of network teaming, the term port is also known as slave. In the teamd service, the term port is preferred while in the NetworkManager service, the term slave refers to interfaces which create a team.

**IMPORTANT**

Certain network teaming features, such as the fail-over mechanism, do not support direct cable connections without a network switch. For further details, see Is bonding supported with direct connection using crossover cables?

7.21.2. Understanding the default behavior of master and slave interfaces

Consider the following default behavior of, when managing or troubleshooting team or bond port interfaces using the NetworkManager service:

- Starting the master interface does not automatically start the port interfaces.
- Starting a port interface always starts the master interface.
- Stopping the master interface also stops the port interface.
- A master without ports can start static IP connections.
- A master without ports waits for ports when starting DHCP connections.
- A master with a DHCP connection waiting for ports completes when you add a port with a carrier.
- A master with a DHCP connection waiting for ports continues waiting when you add a port without carrier.

7.21.3. Comparison of network teaming and bonding features

The following table compares features supported in network teams and network bonds:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Network bond</th>
<th>Network team</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcast Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Round-robin Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Active-backup Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LACP (802.3ad) support</td>
<td>Yes (active only)</td>
<td>Yes</td>
</tr>
<tr>
<td>Hash-based Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User can set hash function</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Tx load-balancing support (TLB)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Feature</td>
<td>Network bond</td>
<td>Network team</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>LACP hash port select</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Load-balancing for LACP support</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ethtool link monitoring</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ARP link monitoring</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NS/NA (IPv6) link monitoring</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ports up/down delays</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Port priorities and stickiness (“primary” option enhancement)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Separate per-port link monitoring setup</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Multiple link monitoring setup</td>
<td>Limited</td>
<td>Yes</td>
</tr>
<tr>
<td>Lockless Tx/Rx path</td>
<td>No (rwlock)</td>
<td>Yes (RCU)</td>
</tr>
<tr>
<td>VLAN support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User-space runtime control</td>
<td>Limited</td>
<td>Yes</td>
</tr>
<tr>
<td>Logic in user-space</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Extensibility</td>
<td>Hard</td>
<td>Easy</td>
</tr>
<tr>
<td>Modular design</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Performance overhead</td>
<td>Low</td>
<td>Very low</td>
</tr>
<tr>
<td>D-Bus interface</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Multiple device stacking</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Zero config using LLDP</td>
<td>No</td>
<td>(in planning)</td>
</tr>
<tr>
<td>NetworkManager support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

7.21.4. Understanding the teamd service, runners, and link-watchers
The team service, `teamd`, controls one instance of the team driver. This instance of the driver adds instances of a hardware device driver to form a team of network interfaces. The team driver presents a network interface, for example `team0`, to the kernel.

The `teamd` service implements the common logic to all methods of teaming. Those functions are unique to the different load sharing and backup methods, such as round-robin, and implemented by separate units of code referred to as runners. Administrators specify runners in JavaScript Object Notation (JSON) format, and the JSON code is compiled into an instance of `teamd` when the instance is created. Alternatively, when using `NetworkManager`, you can set the runner in the `team.runner` parameter, and `NetworkManager` auto-creates the corresponding JSON code.

The following runners are available:

- **broadcast**: Transmits data over all ports.
- **roundrobin**: Transmits data over all ports in turn.
- **activebackup**: Transmits data over one port while the others are kept as a backup.
- **loadbalance**: Transmits data over all ports with active Tx load balancing and Berkeley Packet Filter (BPF)-based Tx port selectors.
- **random**: Transmits data on a randomly selected port.
- **lacp**: Implements the 802.3ad Link Aggregation Control Protocol (LACP).

The `teamd` services uses a link watcher to monitor the state of subordinate devices. The following link-watchers are available:

- **ethtool**: The `libteam` library uses the `ethtool` utility to watch for link state changes. This is the default link-watcher.
- **arp_ping**: The `libteam` library uses the `arp_ping` utility to monitor the presence of a far-end hardware address using Address Resolution Protocol (ARP).
- **nsna_ping**: On IPv6 connections, the `libteam` library uses the Neighbor Advertisement and Neighbor Solicitation features from the IPv6 Neighbor Discovery protocol to monitor the presence of a neighbor’s interface.

Each runner can use any link watcher, with the exception of `lacp`. This runner can only use the `ethtool` link watcher.

### 7.21.5. Installing the teamd service

To configure a network team in `NetworkManager`, you require the `teamd` service and the team plug-in for `NetworkManager`. Both are installed on Red Hat Enterprise Linux 8 by default. This section describes how you install the required packages in case that you remove them.

**Prerequisites**

- An active Red Hat subscription is assigned to the host.

**Procedure**

1. Install the `teamd` and `NetworkManager-team` packages:

   ```bash
   # yum install teamd NetworkManager-team
   ```
7.21.6. Configuring a network team using nmcli commands

This section describes how you configure a network team using **nmcli** commands.

**Prerequisites**
- Two or more network cards are installed in the server.
- The network cards are connected to a switch.

**Procedure**

1. Create the team interface. For example, to create a team interface that uses the **activebackup** runner and both the interface and connection named **team0**, enter:

   ```
   # nmcli connection add type team con-name team0 ifname team0 team.runner activebackup
   ```

2. Optionally, set a link watcher. For example, to set the **ethtool** link watcher, modify the **team0** connection:

   ```
   # nmcli connection modify team0 team.link-watchers "name=ethtool"
   ```

   Link watchers support different parameters. To set parameters for a link watcher, specify them space-separated in the **name** property. Note that the name property must be surrounded by quotes. For example, to use the **ethtool** link watcher and set its **delay-up** parameter to 2500 milliseconds (2.5 seconds):

   ```
   # nmcli connection modify team0 team.link-watchers "name=ethtool delay-up=2500"
   ```

   To set multiple link watchers and each of them with specific parameters, the link watchers must be separated by a comma. The following example sets the **ethtool** link watcher with the **delay-up** parameter and the **arp_ping** link watcher with the **source-host** and **target-host** parameter:

   ```
   # nmcli connection modify team0 team.link-watchers "name=ethtool delay-up=2, name=arp_ping source-host=192.0.2.1 target-host=192.0.2.2"
   ```

3. Configure the IPv4 settings. For example, to set a static IPv4 address, network mask, default gateway, and DNS server of the **team0** connection, enter:

   ```
   # nmcli connection modify team0 ipv4.addresses '192.0.2.1/24'
   # nmcli connection modify team0 ipv4.gateway '192.0.2.254'
   # nmcli connection modify team0 ipv4.dns '192.0.2.253'
   # nmcli connection modify team0 ipv4.method manual
   ```

4. Configure the IPv6 settings. For example, to set a static IPv6 address, network mask, default gateway, and DNS server of the **team0** connection, enter:

   ```
   # nmcli connection modify team0 ipv6.addresses '2001:db8::1/32'
   # nmcli connection modify team0 ipv6.gateway '2001:db8::ffe' 
   # nmcli connection modify team0 ipv6.dns '2001:db8::fffd'
   # nmcli connection modify team0 ipv6.method manual
   ```

5. Optionally, display the network interfaces, and note the names of the interfaces you want to add to the team in the next step:
6. Assign the port interfaces to the team’s connection. For example, to add the interfaces named `enp7s0` and `enp8s0` to the `team0` connection:

```bash
# nmcli connection add type ethernet slave-type team con-name team0-port1 ifname enp7s0 master team0
# nmcli connection add type ethernet slave-type team con-name team0-port2 ifname enp8s0 master team0
```

7. Activate the connection. For example, to activate the `team0` connection:

```bash
# nmcli connection up team0
```

8. Optionally, display the status of the team:

```bash
# teamdctl team0 state
setup:
  runner: activebackup
ports:
  enp7s0
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
down count: 0
  enp8s0
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
down count: 0
runner:
  active port: enp7s0
```

In the example, both ports are up.

**Additional resources**

- Section 7.21.4, “Understanding the teamd service, runners, and link-watchers”.

For \texttt{nmcli} examples, see the \texttt{nmcli-examples(7)} man page.

For all team properties you can set, see the \texttt{team} section in the \texttt{nm-settings(5)} man page.

For parameters you can set in the JSON configuration, as well as JSON examples, see the \texttt{teamd.conf(5)} man page.

### 7.22. CONFIGURING VLAN TAGGING USING NMCLI COMMANDS

This section describes how to configure Virtual Local Area Network (VLAN) tagging using \texttt{nmcli} commands. A VLAN is a logical network within a physical network. The VLAN interface tags packets with the VLAN ID as they pass through the interface, and removes tags of returning packets.

You create a VLAN interface on top of another interface, such as Ethernet, bond, team, or bridge. This interface is called the \texttt{parent interface}.

#### Prerequisites

- The interface you plan to use as a parent to the virtual VLAN interface supports VLAN tags.
- If you configure the VLAN on top of a bond interface:
  - The slaves of the bond are up.
  - The bond is not configured with the \texttt{fail_over_mac=follow} option. A VLAN virtual device cannot change its MAC address to match the parent’s new MAC address. In such a case, the traffic would still be sent with the then incorrect source MAC address.
- The switch the host is connected to is configured to support VLAN tags. For details, see the documentation of your switch.

#### Procedure

1. Optionally, display the available network interfaces:

\begin{verbatim}
# ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 52:54:00:d5:e0:fb brd ff:ff:ff:ff:ff:ff
\end{verbatim}

2. Create the VLAN interface. For example, to create a VLAN interface named \texttt{vlan10} that uses \texttt{enp1s0} as its parent interface and that tags packets with VLAN ID \texttt{10}, enter:

\begin{verbatim}
# nmcli connection add type vlan con-name vlan10 ifname vlan10 vlan.parent enp1s0 vlan.id 10
\end{verbatim}

Note that the VLAN must be within the range from \texttt{0} to \texttt{4094}. 

3. By default, the VLAN connection inherits the maximum transmission unit (MTU) from the parent interface. Optionally, set a different MTU value:

```bash
# nmcli connection modify vlan10 802-3-ethernet.mtu 2000
```

4. Configure the IPv4 settings. For example, to set a static IPv4 address, network mask, default gateway, and DNS server to the `vlan10` connection, enter:

```bash
# nmcli connection modify vlan10 ipv4.addresses '192.0.2.1/24'
# nmcli connection modify vlan10 ipv4.gateway '192.0.2.254'
# nmcli connection modify vlan10 ipv4.dns '192.0.2.253'
# nmcli connection modify vlan10 ipv4.method manual
```

5. Configure the IPv6 settings. For example, to set a static IPv6 address, network mask, default gateway, and DNS server to the `vlan10` connection, enter:

```bash
# nmcli connection modify vlan10 ipv6.addresses '2001:db8::1/32'
# nmcli connection modify vlan10 ipv6.gateway '2001:db8::fffe'
# nmcli connection modify vlan10 ipv6.dns '2001:db8::fffd'
# nmcli connection modify vlan10 ipv6.method manual
```

6. Optionally, verify the settings:

```bash
# ip -d addr show vlan10
4: vlan10@enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    state UP group default qlen 1000
    link/ether 52:54:00:d5:e0:fb brd ff:ff:ff:ff:ff:ff promiscuity 0
    vlan protocol 802.1Q id 10 <REORDER_HDR> numtxqueues 1 numrxqueues 1
gso_max_size 65536 gso_max_segs 65535
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute vlan10
        valid_lft forever preferred_lft forever
    inet6 fe80::8dd7:9030:6f8e:89e6/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

**Additional resources**

- For `nmcli` examples, see the `nmcli-examples(7)` man page.

- For all vlan properties you can set, see the `vlan setting` section in the `nm-settings(5)` man page.
CHAPTER 8. 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

8.1. 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 8.1. 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.

### 8.2. 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 procedure 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:

   ![Network settings window](image)

   **Figure 8.2. Opening the network settings window**

2. Click the plus button to add a new connection:

   • For **Wired** connections, click the plus button next to **Wired** entry and configure the connection.
   
   • For **VPN** connections, click the plus button next to **VPN** entry. If you want to add an **IPsec VPN**, click on **IPsec based VPN** and configure the connection.
   
   • For **Wi-Fi** connections, click the **Wi-Fi** entry on the left-hand side in the **Settings** menu and configure the connection.
8.3. CONFIGURING A WIRED (ETHERNET) CONNECTION USING CONTROL-CENTER

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

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 8.3. 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 8.4. 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 8.5. 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::/64.

**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 8.6. Configuring 802.1X security for a wired with control-center**

![Configuring 802.1X security for a wired with control-center](image)

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 8.3, “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 8.3, “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 8.3, “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
- 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 8.3, “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
- MD5 – Message Digest 5, a cryptographic hash function.
- GTC – Generic Token Card.
8.4. CONFIGURING A WIRED (ETHERNET) CONNECTION USING NM-CONNECTION-EDITOR

Ethernet connections are the most frequently used connection types in physical or virtual servers. This section describes how to configure this connection type in Red Hat Enterprise Linux.

Prerequisites

- A physical or virtual Ethernet device exists in the server’s configuration.

Procedure

1. Open a terminal, and enter:

   ```
   $ nm-connection-editor
   ```

2. Click the + button to add a new connection.

3. Select the Ethernet connection type, and click Create.

4. On the Ethernet tab, select a device and, optionally, further Ethernet-related settings.

![Editing Ethernet connection 1](image)
5. On the IPv4 Settings tab, configure the IPv4 settings. For example, set a static IPv4 address, network mask, default gateway, and DNS server:

<table>
<thead>
<tr>
<th>Method:</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Addresses</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Address</strong></td>
<td><strong>Netmask</strong></td>
</tr>
<tr>
<td>192.0.2.1</td>
<td>24</td>
</tr>
<tr>
<td>DNS servers:</td>
<td>192.0.2.1</td>
</tr>
</tbody>
</table>

6. On the IPv6 Settings tab, configure the IPv6 settings. For example, set a static IPv6 address, network mask, default gateway, and DNS server:

<table>
<thead>
<tr>
<th>Method:</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Addresses</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Address</strong></td>
<td><strong>Prefix</strong></td>
</tr>
<tr>
<td>2001:db8::1</td>
<td>32</td>
</tr>
<tr>
<td>DNS servers:</td>
<td>2001:db8::fffd</td>
</tr>
</tbody>
</table>

7. Save the connection.


### 8.5. CONNECTING TO A WI-FI NETWORK

This procedure describes how you can connect to a wireless network to get access to the internet.

**Procedure**

1. Open the GNOME Shell network connection icon menu from the top right-hand corner of the screen.

2. Select Wi-Fi Not Connected.

3. Click the Select Network option.

4. Click the name of the network to which you want to connect, and then click Connect.
   
   Note that if you do not see the network, the network might be hidden.

5. If the network is protected by a password or encryption keys are required, enter the password and click Connect.
   
   Note that if you do not know the password, contact the administrator of the Wi-Fi network.
6. If the connection is successful, the name of the network is visible in the connection icon menu and the wireless indicator is on the top right-hand corner of the screen.

Additional resources

- Configuring a Wi-Fi connection using control center.

8.6. CONFIGURING A WI-FI CONNECTION USING CONTROL-CENTER

When you connect to a Wi-Fi, the network settings are prefilled depending on the current network connection. This means that the settings will be detected automatically when the interface connects to a network.

This procedure describes how to use control-center to manually configure the Wi-Fi settings.

Prerequisites

- Section 8.5, "Connecting to a Wi-Fi network".

Procedure

1. Press the Super key to enter the Activities Overview, type Wi-Fi and press Enter. In the left-hand-side menu entry you see the list of available networks.

2. Select the gear wheel icon to the right of the Wi-Fi connection name that you want to edit, and the editing connection dialog appears. The Details menu window shows the connection details where you can make further configuration.

   Options

   a. If you select Connect automatically, NetworkManager auto-connects to this connection whenever NetworkManager detects that it is available. If you do not want NetworkManager to connect automatically, clear the check box. Note that when the check box is clear, you have to select that connection manually in the network connection icon’s menu to cause it to connect.

   b. To make a connection available to other users, select the Make available to other users check box.

   c. You can also control the background data usage. If you leave Restrict background data usage unspecified (default), then NetworkManager tries to download data that you are actively using. Otherwise, select the check box and NetworkManager sets the connection as metered, and applies restriction on the background data usage.

      NOTE

      To delete a Wi-Fi connection, click the Forget Connection red box.

3. Select the Identity menu entry to see the basic configuration options.

   SSID — The Service Set Identifier (SSID) of the access point (AP).

   BSSID — The Basic Service Set Identifier (BSSID) is the MAC address, also known as a hardware address, of the specific wireless access point you are connecting to when in Infrastructure mode. This field is blank by default, and you are able to connect to a wireless access point by SSID without having to specify its BSSID. If the BSSID is specified, it will force the system to
associate to a specific access point only. For ad-hoc networks, the **BSSID** is generated randomly by the `mac80211` subsystem when the ad-hoc network is created. It is not displayed by NetworkManager.

**MAC address** – The **MAC address** allows you to associate a specific wireless adapter with a specific connection (or connections).

**Cloned Address** – A cloned MAC address to use in place of the real hardware address. Leave blank unless required.

4. For further IP address configuration, select the **IPv4** and **IPv6** menu entries.

By default, both **IPv4** and **IPv6** are set to automatic configuration depending on current network settings. This means that addresses such as the local IP address, DNS address, and other settings will be detected automatically when the interface connects to a network. If a DHCP server assigns the IP configuration in this network, this is sufficient, but you can also provide static configuration in the **IPv4** and **IPv6** Settings. In the **IPv4** and **IPv6** menu entries, you can see the following settings:

- **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. You can see the assigned IP address in the **Details** menu entry.
  - **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](https://tools.ietf.org/html/rfc3927) 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**
  - If **Automatic** is **ON**, and no DHCP server is available that assigns DNS servers to this connection, switch it to **OFF** to enter the IP address of a DNS server separating the IPs by comma.

- **Routes**
  - Note that 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.

Alternatively, to configure **IPv6** settings in a **Wi-Fi** connection, select the **IPv6** menu entry:
- **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.
  
  - **Disable** – IPv6 is disabled for this connection.

- The **DNS, Routes, Use this connection only for resources on its network** fields are common to IPv4 settings.

5. To configure **Security** settings in a **Wi-Fi** connection, select the **Security** menu entry. The following configuration options are available:

- **Security**
  - **None** – Do not encrypt the Wi-Fi connection.
  
  - **WEP 40/128-bit Key** – Wired Equivalent Privacy (WEP), from the IEEE 802.11 standard. Uses a single pre-shared key (PSK).
  
  - **WEP 128-bit Passphrase** – An MD5 hash of the passphrase to derive a WEP key.

  **WARNING**

  If the Wi-Fi use no encryption, **WEP**, or **WPA**, do not use the network because it is insecure and everyone can read the data you send over this network.

  - **LEAP** – Lightweight Extensible Authentication Protocol, from Cisco Systems.
  
  - **Dynamic WEP (802.1X)** – WEP keys are changed dynamically.
  
  - **WPA & WPA2 Personal** – Wi-Fi Protected Access (WPA), from the draft IEEE 802.11i standard. A replacement for WEP. Wi-Fi Protected Access II (WPA2), from the 802.11i-2004 standard. Personal mode uses a pre-shared key (WPA-PSK).
  
  - **WPA & WPA2 Enterprise** – WPA for use with a RADIUS authentication server to provide IEEE 802.1X network access control.

- **Password** – Enter the password to be used in the authentication process.

6. Once you have finished the configuration, click the **Apply** button to save it.
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.

8.7. CONFIGURING A VPN CONNECTION WITH CONTROL-CENTER

A Virtual Private Network (VPN) is a way of connecting to a local network over the internet. IPsec, provided by Libreswan, is the preferred method for creating a VPN. Libreswan is an open-source, user-space IPsec implementation for VPN. A Virtual Private Network (VPN) enables communication between your Local Area Network (LAN), and another, remote LAN. This is done by setting up a tunnel across an intermediate network such as the Internet. The VPN tunnel that is set up typically uses authentication and encryption.

This procedure describes how to configure a VPN connection using control-center.

Prerequisites

- The NetworkManager-libreswan-gnome package is installed.
- Start for adding a VPN connection as explained Section 8.2, “Creating a network connection using control-center”.

Procedure

1. Select the Identity menu entry to see the basic configuration options:
   - General
     - Gateway – The name or IP address of the remote VPN gateway.
   - Authentication
   - Type
     - IKEv2 (Certificate) - client is authenticated by certificate. It is more secure (default).
     - IKEv1 (XAUTH) - client is authenticated by username and password, or secret (PSK).

The following configuration settings are available under the Advanced section:
Figure 8.7. Advanced options of a VPN connection

![IPsec Advanced Options window](image)

- **Identification**
  - Domain: [field]

- **Security**
  - Phase 1 Algorithms: [field]
  - Phase 2 Algorithms: [field]
  - [ ] Disable PFS
  - Phase 1 Lifetime: [field]
  - Phase 2 Lifetime: [field]
  - [ ] Disable rekeying

- **Connectivity**
  - Remote Network: [field]
  - [ ] narrowing
  - Enable fragmentation: [ ] yes
  - Enable MOBIKE: [ ] no

[Apply button]
When configuring an **IPsec** based VPN connection using the *gnome-control-center* application, the *Advanced* dialog will only display the configuration, but will not allow doing any change. As a consequence, users cannot change any advanced IPsec options. Use the *nm-connection-editor* or *nmcli* tools instead to perform configuration of the advanced properties.

**Identification**

**Domain** – If required, enter the Domain Name.

**Security**

- **Phase1 Algorithms** – corresponds to the *ike* Libreswan parameter – enter the algorithms to be used to authenticate and set up an encrypted channel.

- **Phase2 Algorithms** – corresponds to the *esp* Libreswan parameter – enter the algorithms to be used for the **IPsec** negotiations.
  - Check the **Disable PFS** field to turn off Perfect Forward Secrecy (PFS) to ensure compatibility with old servers that do not support PFS.

- **Phase1 Lifetime** – corresponds to the *ikelifetime* Libreswan parameter – how long the key used to encrypt the traffic will be valid.

- **Phase2 Lifetime** – corresponds to the *salifetime* Libreswan parameter – how long a particular instance of a connection should last before expiring. Note that the encryption key should be changed from time to time for security reasons.

- **Remote network** – corresponds to the *rightsubnet* Libreswan parameter – the destination private remote network that should be reached through the VPN.
  - Check the **narrowing** field to enable narrowing. Note that it is only effective in IKEv2 negotiation.

- **Enable fragmentation** – corresponds to the *fragmentation* Libreswan parameter – whether or not to allow IKE fragmentation. Valid values are *yes* (default), or *no*.

- **Enable Mobike** – corresponds to the *mobike* Libreswan parameter – whether to allow MOBIKE (RFC 4555) to enable a connection to migrate its endpoint without needing to restart the connection from scratch. This is used on mobile devices that switch between wired, wireless or mobile data connections. The values are *no* (default) or *yes*.

2. For further configuration, select the **IPv4** menu entry:

- **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.

  ![CHAPTER 8. CONFIGURING NETWORKING WITH GNOME GUI](image-url)
- **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 it to OFF to enter the IP address of a DNS server you want to use separating the IPs by comma.

- **Routes**
  Note that 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. 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.

3. Alternatively, to configure IPv6 settings in a VPN connection, select the IPv6 menu entry:

- **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.

  - **Disable** – IPv6 is disabled for this connection.

    Note that DNS, Routes, Use this connection only for resources on its network are common to IPv4 settings.

4. Once you have finished editing the VPN connection, click the Add button to customize the configuration or the Apply button to save it for the existing one.

5. Switch the profile to ON to active the VPN 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.

Additional resources

- For more details on the supported Libreswan parameters, see the nm-settings-libreswan man page.

8.8. CONFIGURING A VPN CONNECTION USING NM-CONNECTION-EDITOR

This procedure describes how to configure a VPN connection using nm-connection-editor

Prerequisites

- The NetworkManager-libreswan-gnome package is installed.
- If you configure an Internet Key Exchange version 2 (IKEv2) connection:
  - The certificate is imported into the IPsec network security services (NSS) database.
  - The nickname of the certificate in the NSS database is known.

Procedure

1. Open a terminal, and enter:

   $ nm-connection-editor

2. Click the button to add a new connection.

3. Select the IPsec based VPN connection type, and click Create.

4. On the VPN tab:

   a. Enter the host name or IP address of the VPN gateway into the Gateway field, and select an authentication type. Based on the authentication type, you must enter different additional information:
      - IKEv2 (Certificate) authenticates the client by using a certificate, which is more secure. This setting requires:
        - The nickname of the certificate in the IPsec NSS database
      - IKEv1 (XAUTH) authenticates the user by using a user name and password (pre-shared key). This setting requires that you enter the following values:
        - User name
        - Password
        - Group name
b. If the remote server specifies a local identifier for the IKE exchange, enter the exact string in the **Remote ID** field. In the remote server runs Libreswan, this value is set in the server’s `leftid` parameter.

![Editing VPN connection 1](image)

- **General**
  - **Gateway:** `vpn.example.com`

- **Authentication**
  - **Type:** `IKEv2 (Certificate)`
  - **Certificate name:** `cert_name_in_IPSec_DB`
  - **Remote ID:**

- **Advanced...**

---

c. Optionally, configure additional settings by clicking the **Advanced** button. You can configure the following settings:

- **Identification**
  - **Domain** – If required, enter the domain name.

- **Security**
  - **Phase1 Algorithms** corresponds to the `ike` Libreswan parameter. Enter the algorithms to be used to authenticate and set up an encrypted channel.
  - **Phase2 Algorithms** corresponds to the `esp` Libreswan parameter. Enter the algorithms to be used for the IPsec negotiations.
    - Check the **Disable PFS** field to turn off Perfect Forward Secrecy (PFS) to ensure compatibility with old servers that do not support PFS.
  - **Phase1 Lifetime** corresponds to the `ikelifetime` Libreswan parameter. This parameter defines how long the key used to encrypt the traffic is valid.
○ **Phase2 Lifetime** corresponds to the `salifetime` Libreswan parameter. This parameter defines how long a security association is valid.

- **Connectivity**

  ○ **Remote network** corresponds to the `rightsubnet` Libreswan parameter and defines the destination private remote network that should be reached through the VPN.

    - Check the **narrowing** field to enable narrowing. Note that it is only effective in the IKEv2 negotiation.

  ○ **Enable fragmentation** corresponds to the `fragmentation` Libreswan parameter and defines whether or not to allow IKE fragmentation. Valid values are **yes** (default), or **no**.

  ○ **Enable Mobike** corresponds to the `mobike` Libreswan parameter. The parameter defines whether to allow Mobility and Multihoming Protocol (MOBIKE) (RFC 4555) to enable a connection to migrate its endpoint without needing to restart the connection from scratch. This is used on mobile devices that switch between wired, wireless or mobile data connections. The values are **no** (default) or **yes**.

5. On the **IPv4 Settings** tab, select the IP assignment method and, optionally, set additional static addresses, DNS servers, search domains, and routes.

![Editing VPN connection 1](image)

6. Save the connection.


**Additional resources**

- For further details on the supported IPsec parameters, see the `nm-settings-libreswan(5)` man page.
8.9. CONFIGURING A STATIC ROUTE USING CONTROL-CENTER

You can use control-center in GNOME to add a static route to the configuration of a network connection.

The procedure in this section describes how to add a route to the 192.0.2.0/24 network that uses the gateway running on 198.51.100.1.

Prerequisites

- The network is configured.
- The gateway for the static route must be directly reachable on the interface.
- The network configuration of the connection is opened in the control-center application. See Section 8.3, "Configuring a Wired (Ethernet) connection using control-center".

Procedure

1. Open the IPv4 tab.

2. Optionally, disable automatic routes by clicking the On button in the Routes section of the IPv4 tab to use only static routes. If automatic routes are enabled, Red Hat Enterprise Linux uses static routes and routes received from a DHCP server.

3. Enter the address, netmask, gateway, and optionally a metric value:

<table>
<thead>
<tr>
<th>Routes</th>
<th>Automatic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>Netmask</td>
</tr>
<tr>
<td>192.0.2.0</td>
<td>24</td>
</tr>
</tbody>
</table>

4. Click Apply.

5. Back in the Network window, disable and re-enable the connection by switching the button for the connection to Off and back to On for changes to take effect.

   ! WARNING
   
   Restarting the connection briefly disrupts connectivity on that interface.

6. Optionally, verify that the route is active:

   $ ip route
   ...
   192.0.2.0/24 via 198.51.100.1 dev example proto static metric 100
8.10. CONFIGURING A STATIC ROUTE USING NM-CONNECTION-EDITOR

You can use the `nm-connection-editor` application to add a static route to the configuration of a network connection.

The procedure in this section describes how to add a route to the 192.0.2.0/24 network that uses the gateway running on 198.51.100.1, which is reachable through the example connection.

Prerequisites

- The network is configured.
- The gateway for the static route must be directly reachable on the interface.

Procedure

1. Open a terminal and enter `nm-connection-editor`:

   ```
   $ nm-connection-editor
   ```

2. Select the example connection and click the gear wheel icon to edit the existing connection.

3. Open the IPv4 tab.

4. Click the Routes button.

5. Click the Add button and enter the address, netmask, gateway, and optionally a metric value.

6. Click OK.

7. Click Save.

8. Restart the network connection for changes to take effect. For example, to restart the example connection using the command line:

   ```
   $ sudo nmcli connection up example
   ```

9. Optionally, verify that the route is active:

   ```
   $ ip route
   ...
   192.0.2.0/24 via 198.51.100.1 dev example proto static metric 100
   ```
8.11. SETTING THE DEFAULT GATEWAY ON AN EXISTING CONNECTION USING CONTROL-CENTER

In most situations, administrators set the default gateway when they create a connection as explained in, for example, Section 8.3, “Configuring a Wired (Ethernet) connection using control-center”. This section describes how to set or update the default gateway on a previously created connection using the control-center application.

Prerequisites

- At least one static IP address must be configured on the connection on which the default gateway will be set.
- The network configuration of the connection is open in the control-center application. See Section 8.3, “Configuring a Wired (Ethernet) connection using control-center”.

Procedure

1. Set the IPv4 default gateway. For example, to set the IPv4 address of the default gateway on the connection to 192.0.2.1:
   a. Open the IPv4 tab.
   b. Enter the address in the gateway field next to the IP range the gateway’s address is within:

<table>
<thead>
<tr>
<th>Addresses</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>Netmask</td>
</tr>
<tr>
<td>192.0.2.123</td>
<td>255.255.255.0</td>
</tr>
</tbody>
</table>

2. Set the IPv6 default gateway. For example, to set the IPv6 address of the default gateway on the connection to 2001:db8::1:
   a. Open the IPv6 tab.
   b. Enter the address in the gateway field next to the IP range the gateway’s address is within:

<table>
<thead>
<tr>
<th>Addresses</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>Prefix</td>
</tr>
<tr>
<td>2001:db8::5</td>
<td>64</td>
</tr>
</tbody>
</table>

3. Click Apply.

4. Back in the Network window, disable and re-enable the connection by switching the button for the connection to Off and back to On for changes to take effect.

WARNING

All connections currently using this network connection are temporarily interrupted during the restart.
5. Optionally, verify that the route is active.
   To display the IPv4 default gateway:

   ```
   $ ip -4 route
default via 192.0.2.1 dev example proto static metric 100
   ```

   To display the IPv6 default gateway:

   ```
   $ ip -6 route
default via 2001:db8::1 dev example proto static metric 100 pref medium
   ```

Additional resources

- Section 8.3, “Configuring a Wired (Ethernet) connection using control-center”

8.12. SETTING THE DEFAULT GATEWAY ON AN EXISTING CONNECTION USING NM-CONNECTION-EDITOR

In most situations, administrators set the default gateway when they create a connection as explained in, for example, Section 8.3, “Configuring a Wired (Ethernet) connection using control-center”.

This section describes how to set or update the default gateway on a previously created connection using the `nm-connection-editor` application.

Prerequisites

- At least one static IP address must be configured on the connection on which the default gateway will be set.

Procedure

1. Open a terminal, and enter `nm-connection-editor`:

   ```
   $ nm-connection-editor
   ```

2. Select the connection to modify, and click the gear wheel icon to edit the existing connection.

3. Set the IPv4 default gateway. For example, to set the IPv4 address of the default gateway on the connection to `192.0.2.1`:

   a. Open the IPv4 Settings tab.

   b. Enter the address in the gateway field next to the IP range the gateway’s address is within:

   ```
<table>
<thead>
<tr>
<th>Address</th>
<th>Netmask</th>
<th>Gateway</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.0.2.123</td>
<td>24</td>
<td>192.0.2.1</td>
</tr>
</tbody>
</table>
   ```

4. Set the IPv6 default gateway. For example, to set the IPv6 address of the default gateway on the connection to `2001:db8::1`:

   a. Open the IPv6 tab.

   b. Enter the address in the gateway field next to the IP range the gateway’s address is within:
5. Click **OK**.

6. Click **Save**.

7. Restart the network connection for changes to take effect. For example, to restart the `example` connection using the command line:

   ```
   $ sudo nmcli connection up example
   ```

   **WARNING**

   All connections currently using this network connection are temporarily interrupted during the restart.

8. Optionally, verify that the route is active.
   To display the IPv4 default gateway:

   ```
   $ ip -4 route
default via 192.0.2.1 dev example proto static metric 100
   ```

   To display the IPv6 default gateway:

   ```
   $ ip -6 route
default via 2001:db8::1 dev example proto static metric 100 pref medium
   ```

**Additional resources**

- Section 8.3, "Configuring a Wired (Ethernet) connection using control-center"

### 8.13. CONFIGURING NETWORK BONDING USING NM-CONNECTION-EDITOR

This section describes the basics of network bonding, the differences between bonding and teaming, and how to configure a network bond on Red Hat Enterprise Linux 8 using the `nm-connection-editor` application.

#### 8.13.1. Understanding network bonding

Network bonding is a method to combine or aggregate network interfaces to provide a logical interface with higher throughput or redundancy.

The **active-backup**, **balance-tlb**, and **balance-alb** modes do not require any specific configuration of the network switch. However, other bonding modes require configuring the switch to aggregate the
links. For example, Cisco switches requires **EtherChannel** for modes 0, 2, and 3, but for mode 4, the Link Aggregation Control Protocol (LACP) and **EtherChannel** are required.

For further details, see the documentation of your switch and https://www.kernel.org/doc/Documentation/networking/bonding.txt.

**IMPORTANT**

Certain network bonding features, such as the fail-over mechanism, do not support direct cable connections without a network switch. For further details, see the *Is bonding supported with direct connection using crossover cables?* KCS solution.

### 8.13.2. Understanding the default behavior of master and slave interfaces

Consider the following default behavior of, when managing or troubleshooting team or bond port interfaces using the **NetworkManager** service:

- Starting the master interface does not automatically start the port interfaces.
- Starting a port interface always starts the master interface.
- Stopping the master interface also stops the port interface.
- A master without ports can start static IP connections.
- A master without ports waits for ports when starting DHCP connections.
- A master with a DHCP connection waiting for ports completes when you add a port with a carrier.
- A master with a DHCP connection waiting for ports continues waiting when you add a port without carrier.

### 8.13.3. Comparison of network teaming and bonding features

The following table compares features supported in network teams and network bonds:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Network bond</th>
<th>Network team</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcast Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Round-robin Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Active-backup Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LACP (802.3ad) support</td>
<td>Yes (active only)</td>
<td>Yes</td>
</tr>
<tr>
<td>Hash-based Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User can set hash function</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Tx load-balancing support (TLB)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Feature</td>
<td>Network bond</td>
<td>Network team</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>LACP hash port select</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Load-balancing for LACP support</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ethtool link monitoring</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ARP link monitoring</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NS/NA (IPv6) link monitoring</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ports up/down delays</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Port priorities and stickiness (“primary” option enhancement)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Separate per-port link monitoring setup</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Multiple link monitoring setup</td>
<td>Limited</td>
<td>Yes</td>
</tr>
<tr>
<td>Lockless Tx/Rx path</td>
<td>No (rwlock)</td>
<td>Yes (RCU)</td>
</tr>
<tr>
<td>VLAN support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User-space runtime control</td>
<td>Limited</td>
<td>Yes</td>
</tr>
<tr>
<td>Logic in user-space</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Extensibility</td>
<td>Hard</td>
<td>Easy</td>
</tr>
<tr>
<td>Modular design</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Performance overhead</td>
<td>Low</td>
<td>Very low</td>
</tr>
<tr>
<td>D-Bus interface</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Multiple device stacking</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Zero config using LLDP</td>
<td>No</td>
<td>(in planning)</td>
</tr>
<tr>
<td>NetworkManager support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### 8.13.4. Configuring a network bond using `nm-connection-editor`

This section describes how to configure a network bond using the `nm-connection-editor` application.

---

Red Hat Enterprise Linux 8 Configuring and managing networking

98
Prerequisites

- Two or more network cards are installed in the server.
- The network cards are connected to a switch.

Procedure

1. Open a terminal, and enter `nm-connection-editor`:

   ```
   $ nm-connection-editor
   ```

2. Click the + button to add a new connection.

3. Select the Bond connection type, and click Create.

4. On the Bond tab:
   a. Optionally, set the name of the bond interface in the Interface name field.
   b. Click the Add button to add a network interface as a slave to the bond.
      i. Select the connection type of the interface. For example, select Ethernet for a wired connection.
      ii. Optionally, set a connection name for the slave device.
      iii. In the Device field on the Ethernet tab, select the network interface you want to add as a slave to the bond.
   
   IMPORTANT
   You can only use network interfaces in a bond that are not configured.

   iv. Click Save.

   c. Repeat the previous step for each interface you want to add to the bond:
d. Optionally, set other options, such as the Media Independent Interface (MII) monitoring interval.

5. On the IPv4 Settings tab, configure the IPv4 settings. For example, set a static IPv4 address, network mask, default gateway, and DNS server:

![IPv4 Settings](image)

6. On the IPv6 Settings tab, configure the IPv6 settings. For example, set a static IPv6 address, network mask, default gateway, and DNS server:
7. Click **Save** to save the bond connection.

8. Close **nm-connection-editor**.

9. Optionally, display the status of the bond:

```bash
$ cat /proc/net/bonding/_bond0_
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup)
Primary Slave: None
Currently Active Slave: enp7s0
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: enp7s0
MII Status: up
Speed: Unknown
Duplex: Unknown
Link Failure Count: 0
Permanent HW addr: 52:54:00:d5:e0:fb
Slave queue ID: 0

Slave Interface: enp8s0
MII Status: up
Speed: Unknown
Duplex: Unknown
```
8.14. CONFIGURING A NETWORK BRIDGE USING NM-CONNECTION-EDITOR

This section explains how to configure a network bridge using the `nm-connection-editor` application.

A network bridge is a link-layer device which forwards traffic between networks based on MAC addresses. The bridge device decides on forwarding packages based on a table of MAC addresses. The bridge builds the MAC addresses table by listening to network traffic and thereby learning what hosts are connected to each network.

For example, you can use a software bridge on a Red Hat Enterprise Linux 8 host:

- To emulate a hardware bridge
- In virtualization environments, to integrate virtual machines (VM) to the same network as the host

Due to the IEEE 802.11 standard which specifies the use of 3-address frames in Wi-Fi for the efficient use of airtime, you cannot configure a bridge over Wi-Fi networks operating in Ad-Hoc or Infrastructure modes.

Prerequisites

- Two or more physical or virtual network devices are installed in the server.

Procedure

1. Open a terminal, and enter `nm-connection-editor`:

   ```bash
   $ nm-connection-editor
   ```

2. Click the + button to add a new connection.

3. Select the Bridge connection type, and click Create.
   a. Optionally, set the name of the bridge interface in the Interface name field.
   b. Click the Add button to add a network interface as a slave to the bridge.
      i. Select the connection type of the interface. For example, select Ethernet for a wired connection.
      ii. Optionally, set a connection name for the slave device.
      iii. In the Device field on the Ethernet tab, select the network interface you want to add as a slave to the bridge.
   iv. Click Save.
   c. Repeat the previous step for each interface you want to add to the bridge.
i. Optionally, configure further bridge settings, such as Spanning Tree Protocol (STP) options.

4. On the **IPv4 Settings** tab, configure the IPv4 settings. For example, set a static IPv4 address, network mask, default gateway, and DNS server:
5. On the IPv6 Settings tab, configure the IPv6 settings. For example, set a static IPv6 address, network mask, default gateway, and DNS server:

![Editing Bridge connection 1](image)

- Addresses:
  - Address: 2001:db8::1, Prefix: 32, Gateway: 2001:db8::fff3
  - DNS servers: 2001:db8::fffd

6. Save the bridge connection.


8. Optionally, use the following command to display the status of the bridge:

```
# bridge link show bridge0
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state forwarding priority 32 cost 100
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state listening priority 32 cost 100
```

8.15. CONFIGURING A NETWORK TEAM USING NM-CONNECTION-EDITOR

This section describes the basics of network teaming, the differences between bonding and teaming, and how to configure a network team on Red Hat Enterprise Linux 8 using the `nm-connection-editor` application.

**Prerequisites**

- Red Hat Enterprise Linux 8 is installed.
- The system has an active subscription assigned.

8.15.1. Understanding network teaming

Network teaming is a feature that combines or aggregates network interfaces to provide a logical interface with higher throughput or redundancy.

Network teaming uses a kernel driver to implement fast handling of packet flows, as well as user-space libraries and services for other tasks. This way, network teaming is an easily extensible and scalable solution for load-balancing and redundancy requirements.
Note that in the context of network teaming, the term **port** is also known as **slave**. In the **teamd** service, the term **port** is preferred while in the **NetworkManager** service, the term **slave** refers to interfaces which create a team.

**IMPORTANT**

Certain network teaming features, such as the fail-over mechanism, do not support direct cable connections without a network switch. For further details, see *Is bonding supported with direct connection using crossover cables?*

### 8.15.2. Understanding the default behavior of master and slave interfaces

Consider the following default behavior of, when managing or troubleshooting team or bond port interfaces using the **NetworkManager** service:

- Starting the master interface does not automatically start the port interfaces.
- Starting a port interface always starts the master interface.
- Stopping the master interface also stops the port interface.
- A master without ports can start static IP connections.
- A master without ports waits for ports when starting DHCP connections.
- A master with a DHCP connection waiting for ports completes when you add a port with a carrier.
- A master with a DHCP connection waiting for ports continues waiting when you add a port without carrier.

### 8.15.3. Comparison of network teaming and bonding features

The following table compares features supported in network teams and network bonds:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Network bond</th>
<th>Network team</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcast Tx policy</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Round-robin Tx policy</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Active-backup Tx policy</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LACP (802.3ad) support</td>
<td>Yes (active only)</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Hash-based Tx policy</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>User can set hash function</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Tx load-balancing support (TLB)</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Network bond</td>
<td>Network team</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>LACP hash port select</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Load-balancing for LACP support</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Ethtool link monitoring</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>ARP link monitoring</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NS/NA (IPv6) link monitoring</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Ports up/down delays</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Port priorities and stickiness (“primary” option enhancement)</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Separate per-port link monitoring setup</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Multiple link monitoring setup</td>
<td>Limited</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Lockless Tx/Rx path</td>
<td>No (rwlock)</td>
<td>Yes (RCU)</td>
<td></td>
</tr>
<tr>
<td>VLAN support</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>User-space runtime control</td>
<td>Limited</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Logic in user-space</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Extensibility</td>
<td>Hard</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>Modular design</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Performance overhead</td>
<td>Low</td>
<td>Very low</td>
<td></td>
</tr>
<tr>
<td>D-Bus interface</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Multiple device stacking</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Zero config using LLDP</td>
<td>No</td>
<td>(in planning)</td>
<td></td>
</tr>
<tr>
<td>NetworkManager support</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

**8.15.4. Understanding the teamd service, runners, and link-watchers**
The team service, **teamd**, controls one instance of the team driver. This instance of the driver adds instances of a hardware device driver to form a team of network interfaces. The team driver presents a network interface, for example **team0**, to the kernel.

The **teamd** service implements the common logic to all methods of teaming. Those functions are unique to the different load sharing and backup methods, such as round-robin, and implemented by separate units of code referred to as **runners**. Administrators specify runners in JavaScript Object Notation (JSON) format, and the JSON code is compiled into an instance of **teamd** when the instance is created. Alternatively, when using **NetworkManager**, you can set the runner in the **team.runner** parameter, and **NetworkManager** auto-creates the corresponding JSON code.

The following runners are available:

- **broadcast**: Transmits data over all ports.
- **roundrobin**: Transmits data over all ports in turn.
- **activebackup**: Transmits data over one port while the others are kept as a backup.
- **loadbalance**: Transmits data over all ports with active Tx load balancing and Berkeley Packet Filter (BPF)-based Tx port selectors.
- **random**: Transmits data on a randomly selected port.
- **lacp**: Implements the 802.3ad Link Aggregation Control Protocol (LACP).

The **teamd** services uses a link watcher to monitor the state of subordinate devices. The following link-watchers are available:

- **ethtool**: The **libteam** library uses the **ethtool** utility to watch for link state changes. This is the default link-watcher.
- **arp_ping**: The **libteam** library uses the **arp_ping** utility to monitor the presence of a far-end hardware address using Address Resolution Protocol (ARP).
- **nsna_ping**: On IPv6 connections, the **libteam** library uses the Neighbor Advertisement and Neighbor Solicitation features from the IPv6 Neighbor Discovery protocol to monitor the presence of a neighbor’s interface.

Each runner can use any link watcher, with the exception of **lacp**. This runner can only use the **ethtool** link watcher.

### 8.15.5. Installing the teamd service

To configure a network team in **NetworkManager**, you require the **teamd** service and the team plug-in for **NetworkManager**. Both are installed on Red Hat Enterprise Linux 8 by default. This section describes how you install the required packages in case that you remove them.

**Prerequisites**

- An active Red Hat subscription is assigned to the host.

**Procedure**

1. Install the **teamd** and **NetworkManager-team** packages:

```
# yum install teamd NetworkManager-team
```
8.15.6. Configuring a network team using nm-connection-editor

This section describes how you configure a network team using the `nm-connection-editor` application.

Prerequisites

- Two or more network cards are installed in the server.
- The network cards are connected to a switch.

Procedure

1. Open a terminal, and enter `nm-connection-editor`:

   ```
   $ nm-connection-editor
   ```

2. Click the + button to add a new connection.

3. Select the **Team** connection type, and click **Create**.

4. On the **Team** tab:
   a. Optionally, set the name of the team interface in the **Interface name** field.
   b. Click the **Add** button to add a network interface as a slave to the team.
      i. Select the connection type of the interface. For example, select **Ethernet** for a wired connection.
      ii. Optionally, set a connection name for the slave device.
      iii. In the **Device** field on the **Ethernet** tab, select the network interface you want to add as a slave to the team.

   **IMPORTANT**

   You can only use network interfaces in a team that are not configured.

   iv. Click **Save**.

   c. Repeat the previous step for each interface you want to add to the team.
d. Click the **Advanced** button to set advanced options to the team connection.

i. On the **Runner** tab, select the runner.

ii. On the **Link Watcher** tab, set the link link watcher and its optional settings.

iii. Click **OK**.

5. On the **IPv4 Settings** tab, configure the IPv4 settings. For example, set a static IPv4 address, network mask, default gateway, and DNS server:
6. On the **IPv6 Settings** tab, configure the IPv6 settings. For example, set a static IPv6 address, network mask, default gateway, and DNS server:

![IPv6 Settings](image)

7. Click **Save** to save the team connection.

8. Close **nm-connection-editor**.

9. Optionally, display the status of the team:

```
# teamdctl team0 state
setup:
  runner: activebackup
ports:
enp7s0
  link watches:
    link summary: up
    instance[link_watch_0]:
      name: ethtool
      link: up
down count: 0
enp8s0
  link watches:
    link summary: up
    instance[link_watch_0]:
      name: ethtool
      link: up
down count: 0
runner:
  active port: enp7s0
```

**Additional resources**

- Section 8.15.4, “Understanding the teamd service, runners, and link-watchers”.

8.16. CONFIGURING VLAN TAGGING USING NM-CONNECTION-EDITOR

This section describes how to configure Virtual Local Area Network (VLAN) tagging using the nm-connection-editor application. A VLAN is a logical network within a physical network. The VLAN interface tags packets with the VLAN ID as they pass through the interface, and removes tags of returning packets.

You create a VLAN interface on top of another interface, such as Ethernet, bond, team, or bridge. This interface is called the parent interface.

Prerequisites

- The interface you plan to use as a parent to the virtual VLAN interface supports VLAN tags.
- If you configure the VLAN on top of a bond interface:
  - The slaves of the bond are up.
  - The bond is not configured with the fail_over_mac=follow option. A VLAN virtual device cannot change its MAC address to match the parent’s new MAC address. In such a case, the traffic would still be sent with the then incorrect source MAC address.
- The switch the host is connected to is configured to support VLAN tags. For details, see the documentation of your switch.

Procedure

1. Open a terminal, and enter nm-connection-editor:

   ```
   $ nm-connection-editor
   ```

2. Click the + button to add a new connection.

3. Select the VLAN connection type, and click Create.

4. On the VLAN tab:
   a. Select the parent interface.
   b. Select the VLAN id. Note that the VLAN must be within the range from 0 to 4094.
   c. By default, the VLAN connection inherits the maximum transmission unit (MTU) from the parent interface. Optionally, set a different MTU value.
   d. Optionally, set the name of the VLAN interface and further VLAN-specific options.
5. On the **IPv4 Settings** tab, configure the IPv4 settings. For example, set a static IPv4 address, network mask, default gateway, and DNS server:

```plaintext
Connection name: VLAN connection 1

<table>
<thead>
<tr>
<th>General</th>
<th>VLAN</th>
<th>Proxy</th>
<th>IPv4 Settings</th>
<th>IPv6 Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent interface:</td>
<td><strong>enp1s0 (52:54:00:72:2F:6E)</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VLAN id:</td>
<td>0</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>VLAN interface name:</td>
<td><strong>vlan0</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Cloned MAC address:</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MTU:</td>
<td><strong>automatic</strong></td>
<td>-</td>
<td>+</td>
<td>bytes</td>
</tr>
<tr>
<td>Flags:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. On the **IPv6 Settings** tab, configure the IPv6 settings. For example, set a static IPv6 address, network mask, default gateway, and DNS server:

```plaintext
Connection name: VLAN connection 1

<table>
<thead>
<tr>
<th>General</th>
<th>VLAN</th>
<th>Proxy</th>
<th>IPv4 Settings</th>
<th>IPv6 Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method:</td>
<td><strong>Manual</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Addresses**

<table>
<thead>
<tr>
<th>Address</th>
<th>Netmask</th>
<th>Gateway</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.0.2.1</td>
<td>24</td>
<td>192.0.2.254</td>
</tr>
</tbody>
</table>

| DNS servers: | 192.0.2.253 |
```
6. On the **IPv6 Settings** tab, configure the IPv6 settings. For example, set a static IPv6 address, network mask, default gateway, and DNS server:

```
<table>
<thead>
<tr>
<th>Address</th>
<th>Prefix</th>
<th>Gateway</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001:db8::1</td>
<td>32</td>
<td>2001:db8::ffe</td>
</tr>
</tbody>
</table>
```

7. Click **Save** to save the VLAN connection.

8. Close **nm-connection-editor**.

9. Optionally, verify the settings:

```bash
# ip -d addr show vlan10
4: vlan10@enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
   link/ether 52:54:00:d5:e0:fb brd ff:ff:ff:ff:ff:ff promiscuity 0
   vlan protocol 802.1Q id 10 <REORDER_HDR> numtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535
   inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute vlan10
      valid_lft forever preferred_lft forever
   inet6 fe80::8dd7:9030:6f8e:89e6/64 scope link noprefixroute
      valid_lft forever preferred_lft forever
```

![Editing VLAN connection 1](image_url)
CHAPTER 9. CONFIGURING IP NETWORKING WITH IFcfg FILES

This section describes how to configure a network interface manually by editing the ifcfg files.

Interface configuration (ifcfg) files control the software interfaces for individual network devices. As the system boots, it uses these files to determine what interfaces to bring up and how to configure them. These files are usually named ifcfg-name, where the suffix name refers to the name of the device that the configuration file controls. By convention, the ifcfg file’s suffix is the same as the string given by the DEVICE directive in the configuration file itself.

Note, that in RHEL 8 ifcfg files demand NetworkManager running to use the functionality of the current solution.

9.1. CONFIGURING AN INTERFACE WITH STATIC NETWORK SETTINGS USING IFcfg FILES

This procedure describes how to configure a network interface using ifcfg files.

Prerequisites

- NetworkManager running.

Procedure

To configure an interface with static network settings using ifcfg files, for an interface with the name eth0, create a file with the name ifcfg-eth0 in the /etc/sysconfig/network-scripts/ directory that contains:

- For IPv4 configuration:

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
PREFIX=24
IPADDR=10.0.1.27
GATEWAY=10.0.1.1
```

- For IPv6 configuration:

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
IPV6INIT=yes
IPV6ADDR=2001:db8::2/48
```

For more IPv6 ifcfg configuration options, see nm-settings-ifcfg-rh(5) man page.

9.2. CONFIGURING AN INTERFACE WITH DYNAMIC NETWORK SETTINGS USING IFcfg FILES

This procedure describes how to configure a network interface with dynamic network settings using ifcfg files.
Prerequisites

- NetworkManager running.

Procedure

1. To configure an interface named em1 with dynamic network settings using `ifcfg` files, create a file with the name `ifcfg-em1` in the `/etc/sysconfig/network-scripts/` directory that contains:

   ```
   DEVICE=em1
   BOOTPROTO=dhcp
   ONBOOT=yes
   ```

2. To configure an interface to send a different host name to the DHCP server, add the following line to the `ifcfg` file:

   ```
   DHCP_HOSTNAME=hostname
   ```

3. To configure an interface to send a different fully qualified domain name (FQDN) to the DHCP server, add the following line to the `ifcfg` file:

   ```
   DHCP_FQDN=fully.qualified.domain.name
   ```

   **NOTE**
   Only one directive, either `DHCP_HOSTNAME` or `DHCP_FQDN`, should be used in a given `ifcfg` file. In case both `DHCP_HOSTNAME` and `DHCP_FQDN` are specified, only the latter is used.

4. To configure an interface to use particular DNS servers, add the following lines to the `ifcfg` file:

   ```
   PEERDNS=no
   DNS1=ip-address
   DNS2=ip-address
   ```

   where `ip-address` is the address of a DNS server. This will cause the network service to update `/etc/resolv.conf` with the specified DNS servers specified. Only one DNS server address is necessary, the other is optional.

9.3. MANAGING SYSTEM-WIDE AND PRIVATE CONNECTION PROFILES WITH IFCFG FILES

This procedure describes how to configure `ifcfg` files to manage the system-wide and private connection profiles.

Prerequisites

- NetworkManager running.

Procedure

The permissions correspond to the `USERS` directive in the `ifcfg` files. If the `USERS` directive is not present, the network profile will be available to all users.
1. As an example, modify the `ifcfg` file with the following row, which will make the connection available only to the users listed:

```
    USERS="joe bob alice"
```
CHAPTER 10. CONFIGURING MACSEC

The following section provides information on how to configure **Media Control Access Security (MACsec)**, which is an 802.1AE IEEE standard security technology for secure communication in all traffic on Ethernet links.

10.1. INTRODUCTION TO MACSEC

**Media Access Control Security (MACsec, IEEE 802.1AE)** encrypts and authenticates all traffic in LANs with the GCM-AES-128 algorithm. **MACsec** can protect not only **IP** but also **Address Resolution Protocol (ARP)**, **Neighbor Discovery (ND)**, or **DHCP**. While **IPsec** operates on the network layer (layer 3) and **SSL** or **TLS** on the application layer (layer 7), **MACsec** operates in the data link layer (layer 2). Combine **MACsec** with security protocols for other networking layers to take advantage of different security features that these standards provide.

10.2. USING MACSEC WITH NMCLI TOOL

This procedure shows how to configure **MACsec** with **nmcli** tool.

**Prerequisites**

- The **NetworkManager** must be running.
- You already have a 16-byte hexadecimal CAK (**$MKA_CAK**) and a 32-byte hexadecimal CKN (**$MKA_CKN**).

**Procedure**

```bash
~]# nmcli connection add type macsec con-name test-macsec+ ifname macsec0 connection.autoconnect no macsec.parent eth0 macsec.mode psk macsec.mka-cak $MKA_CAK macsec.mka-ckn $MKA_CKN
~]# nmcli connection up test-macsec+
```

After this step, the `macsec0` device is configured and can be used for networking.

10.3. USING MACSEC WITH WPA_SUPPLICANT

This procedure shows how to enable **MACsec** with a switch that performs authentication using a pre-shared Connectivity Association Key/CAK Name (CAK/CKN) pair.

**Procedure**

1. Create a CAK/CKN pair. For example, the following command generates a 16-byte key in hexadecimal notation:

   ```bash
   ~]# dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2 "%02x"'
   ```

2. Create the `wpa_supplicant.conf` configuration file and add the following lines to it:

   ```text
   ctrl_interface=/var/run/wpa_supplicant
   ```
Use the values from the previous step to complete the `mka_cak` and `mka_ckn` lines in the `wpa_supplicant.conf` configuration file.

For more information, see the `wpa_supplicant.conf(5)` man page.

3. Assuming you are using `eth0` to connect to your network, start `wpa_supplicant` using the following command:

```
~# wpa_supplicant -i eth0 -Dmacsec_linux -c wpa_supplicant.conf
```

10.4. RELATED INFORMATION

For more details, see the What’s new in MACsec: setting up MACsec using wpa_supplicant and (optionally) NetworkManager article. In addition, see the MACsec: a different solution to encrypt network traffic article for more information about the architecture of a MACsec network, use case scenarios, and configuration examples.
CHAPTER 11. GETTING STARTED WITH IPVLAN

This document describes the IPVLAN driver.

11.1. 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.

11.2. 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.

11.3. OVERVIEW OF MACVLAN

The MACVLAN driver allows to create multiple virtual network devices on top of a single NIC, each of them identified by its own unique MAC address. Packets which land on the physical NIC are demultiplexed towards the relevant MACVLAN device via MAC address of the destination. MACVLAN devices do not add any level of encapsulation.

11.4. COMPARISON OF IPVLAN AND MACVLAN

The following table shows the major differences between MACVLAN and IPVLAN.
MACVLAN | IPVLAN
---|---
Uses MAC address for each MACVLAN device. The overlimit of MAC addresses of MAC table in switch might cause loosing the connectivity. | Uses single MAC address which does not limit the number of IPVLAN devices.

Netfilter rules for global namespace cannot affect traffic to or from MACVLAN device in a child namespace. | It is possible to control traffic to or from IPVLAN device in L3 mode and L3S mode.

Note that both IPVLAN and MACVLAN do not require any level of encapsulation.

### 11.5. CONFIGURING IPVLAN NETWORK

#### 11.5.1. Creating and configuring the IPVLAN device using iproute2

This procedure shows how to set up the IPVLAN device using iproute2.

**Procedure**

1. To create an IPVLAN device, enter the following command:

   ```
   ~
   # ip link add link real_NIC_device name IPVLAN_device type ipvlan mode l2
   ```

   Note that network interface controller (NIC) is a hardware component which connects a computer to a network.

   **Example 11.1. Creating an IPVLAN device**

   ```
   ~
   # ip link add link enp0s31f6 name my_ipvlan type ipvlan mode l2
   ~
   # ip link
   47: my_ipvlan@enp0s31f6: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000 link/ether e8:6a:6e:8a:a2:44 brd ff:ff:ff:ff:ff:ff
   ```

2. To assign an IPv4 or IPv6 address to the interface, enter the following command:

   ```
   ~
   # ip addr add dev IPVLAN_device IP_address/subnet_mask_prefix
   ```

3. In case of configuring an IPVLAN device in L3 mode or L3S mode, make the following setups:
   a. Configure the neighbor setup for the remote peer on the remote host:

   ```
   ~
   # ip neigh add dev peer_device IPVLAN_device_IP_address lladdr MAC_address
   ```

   where MAC_address is the MAC address of the real NIC on which an IPVLAN device is based on.

   b. Configure an IPVLAN device for L3 mode with the following command:
```
~]$ ip neigh add dev real_NIC_device peer_IP_address lladdr peer_MAC_address
```

For L3S mode:

```
~]$ ip route dev add real_NIC_device peer_IP_address/32
```

where IP-address represents the address of the remote peer.

4. To set an IPVLAN device active, enter the following command:

```
~]$ ip link set dev IPVLAN_device up
```

5. To check if the IPVLAN device is active, execute the following command on the remote host:

```
~]$ ping IP_address
```

where the IP_address uses the IP address of the IPVLAN device.
CHAPTER 12. CONFIGURING VIRTUAL ROUTING AND FORWARDING (VRF)

With Virtual routing and forwarding (VRF), Administrators can use multiple routing tables simultaneously on the same host. For that, VRF partitions a network at layer 3. This enables the administrator to isolate traffic using separate and independent route tables per VRF domain. This technique is similar to virtual LANs (VLAN), which partitions a network at layer 2, where the operating system uses different VLAN tags to isolate traffic sharing the same physical medium.

One benefit of VRF over partitioning on layer 2 is that routing scales better considering the number of peers involved.

Red Hat Enterprise Linux uses a virtual vrt device for each VRF domain and adds routes to a VRF domain by enslaving existing network devices to a VRF device. Addresses and routes previously attached to the enslaved device will be moved inside the VRF domain.

Note that each VRF domain is isolated from each other.

12.1. TEMPORARILY REUSING THE SAME IP ADDRESS ON DIFFERENT INTERFACES

The procedure in this section describes how to temporarily use the same IP address on different interfaces in one server by using the virtual routing and forwarding (VRF) feature. Use this procedure only for testing purposes, because the configuration is temporary and lost after you reboot the system.

For a permanent solution, use a NetworkManager dispatcher script as described in Section 12.2, “Permanently reusing the same IP address on different interfaces”.

**IMPORTANT**

To enable remote peers to contact both VRF interfaces while reusing the same IP address, the network interfaces must belong to different broadcasting domains. A broadcast domain in a network is a set of nodes which receive broadcast traffic sent by any of them. In most configurations, all nodes connected to the same switch belong to the same broadcasting domain.

**Prerequisites**

- You are logged in as the root user.
- The network interfaces are not configured.

**Procedure**

1. Create and configure the first VRF device:

   a. Create the VRF device and assign it to a routing table. For example, to create a VRF device named blue that is assigned to the 1001 routing table:

      ```
      # ip link add dev blue type vrf table 1001
      ```

   b. Enable the blue device:

      ```
      # ip link set dev blue up
      ```
c. Assign a network device to the VRF device. For example, to add the eth0 Ethernet device to the blue VRF device:

```
# ip link set dev eth0 master blue
```

d. Enable the eth0 device:

```
# ip link set dev eth0 up
```

e. Assign an IP address and subnet mask to the eth0 device. For example, to set it to 192.0.2.1/24:

```
# ip addr add dev eth0 192.0.2.1/24
```

2. Create and configure the next VRF device:

a. Create the VRF device and assign it to a routing table. For example, to create a VRF device named red that is assigned to the 1002 routing table:

```
# ip link add dev red type vrf table 1002
```

b. Enable the red device:

```
# ip link set dev red up
```

c. Assign a network device to the VRF device. For example, to add the eth1 Ethernet device to the red VRF device:

```
# ip link set dev eth1 master red
```

d. Enable the eth1 device:

```
# ip link set dev eth1 up
```

e. Assign the same IP address and subnet mask to the eth1 device as you used for eth0 in the blue VRF domain:

```
# ip addr add dev eth1 192.0.2.1/24
```

3. Optionally, create further VRF devices as described above.

### 12.2. PERMANENTLY REUSING THE SAME IP ADDRESS ON DIFFERENT INTERFACES

NetworkManager does not explicitly support configuring virtual routing and forwarding (VRF) devices to permanently use the same IP address on different interfaces in one server. However, you can use NetworkManager to assign an IP address to the interfaces, and create a NetworkManager dispatcher script that creates and enables the VRF device. Use this procedure for production environments.

For a temporary solution whose configuration is lost after you reboot the system, see Section 12.1, “Temporarily reusing the same IP address on different interfaces”.
Prerequisites

- You are logged in as the root user.
- You assigned the 192.0.2.1/24 IP and subnet mask to the eth0 and eth1 Ethernet interface. For details, see Section 7.8, “Configuring a static Ethernet connection”.

Procedure

1. Create the /etc/NetworkManager/dispatcher.d/pre-up.d/01-vrf file with the following content:

```bash
#!/bin/sh

interface=$1

if [ $interface = eth0 ]; then
    # If the interface is "eth0", set the variable for the VRF
    # device name to "blue" and the variable for the routing
    # table to "1001"
    vrf=blue
    id=1001
elif [ $interface = eth1 ]; then
    # If the interface is "eth1", set the variable for the VRF
    # device name to "red" and the variable for the routing
    # table to "1002"
    vrf=red
    id=1002
else
    # For all other devices stop executing the script here
    exit 0
fi

# Create the VRF device if it does not exist, and assign
# the VRF device to its routing table
ip link show dev $vrf || ip link add dev $vrf type vrf table $id

# Enable the VRF device
ip link set dev $vrf up

# Assign the Ethernet interface to the VRF device
ip link set dev $interface master $vrf
```

2. Set the x bits to the /etc/NetworkManager/dispatcher.d/pre-up.d/01-vrf file to make it executable:

```bash
# chmod 0755 /etc/NetworkManager/dispatcher.d/pre-up.d/01-vrf
```

NetworkManager will run this script when an interface changes its mode to up.

Additional resources

- The script described in this section creates the same VRF configuration as described in Section 12.1, “Temporarily reusing the same IP address on different interfaces”, but as a permanent solution. For further details about the commands used in the script, see the mentioned section.
Related information

CHAPTER 13. SETTING THE ROUTING PROTOCOLS FOR YOUR SYSTEM

This section describes how to use the Free Range Routing (FRRouting, or FRR) feature to enable and set the required routing protocols for your system.

13.1. INTRODUCTION TO FRROUTING

Free Range Routing (FRRouting, or FRR) is a routing protocol stack, which is provided by the frr package available in the AppStream repository.

FRR replaces Quagga that was used on previous RHEL versions. As such, FRR provides TCP/IP-based routing services with support for multiple IPv4 and IPv6 routing protocols.

The supported protocols are:

- Border Gateway Protocol (BGP)
- Intermediate System to Intermediate System (IS-IS)
- Open Shortest Path First (OSPF)
- Protocol-Independent Multicast (PIM)
- Routing Information Protocol (RIP)
- Routing Information Protocol next generation (RIPng)
- Enhanced Interior Gateway Routing Protocol (EIGRP)
- Next Hop Resolution Protocol (NHRP)
- Bidirectional Forwarding Detection (BFD)
- Policy-based Routing (PBR)

FRR is a collection of the following services:

- zebra
- bgpd
- isisd
- ospfd
- ospf6d
- pimd
- ripd
- ripngd
- eigrpd
If \texttt{frr} is installed, the system can act as a dedicated router, which exchanges routing information with other routers in either internal or external network using the routing protocols.

\section*{13.2. SETTING UP FRROUTING}

\subsection*{Prerequisites}
- Make sure that the \texttt{frr} package is installed on your system:

\begin{verbatim}
# yum install frr
\end{verbatim}

\subsection*{Procedure}
\begin{enumerate}
\item Edit the \texttt{/etc/frr/daemons} configuration file, and enable the required daemons for your system. For example, to enable the \texttt{ripd} daemon, include the following line:

\begin{verbatim}
ripd=yes
\end{verbatim}

\begin{warning}
\textbf{WARNING}

The \texttt{zebra} daemon must always be enabled, so that you must set \texttt{zebra=yes} to be able to use \texttt{FRR}.
\end{warning}

\begin{important}
\textbf{IMPORTANT}

By default, \texttt{/etc/frr/daemons} contains \texttt{[daemon\_name]=no} entries for all daemons. Therefore, all daemons are disabled, and starting \texttt{FRR} after a new installation of the system has no effect.
\end{important}

\item Start the \texttt{frr} service:

\begin{verbatim}
# systemctl start frr
\end{verbatim}

\item Optionally, you can also set \texttt{FRR} to start automatically on boot:

\begin{verbatim}
# systemctl enable frr
\end{verbatim}
\end{enumerate}

\section*{13.3. MODIFYING THE CONFIGURATION OF FRR}
This section describes:

- How to enable an additional daemon after you set up FRR
- How to disable a daemon after you set up FRR

### Enabling an additional daemon

#### Prerequisites

- FRR is set up as described in Section 13.2, “Setting up FRRouting”.

#### Procedure

To enable one or more additional daemons:

1. Edit the `/etc/frr/daemons` configuration file, and modify the line for the required daemons to state `yes` instead of `no`.
   
   For example, to enable the ripd daemon:
   ```
   ripd=yes
   ```

2. Reload the `frr` service:
   ```
   # systemctl reload frr
   ```

### Disabling a daemon

#### Prerequisites

- FRR is set up as described in Section 13.2, “Setting up FRRouting”.

#### Procedure

To disable one or more daemons:

1. Edit the `/etc/frr/daemons` configuration file, and modify the line for the required daemons to state `no` instead of `yes`.
   
   For example, to disable the ripd daemon:
   ```
   ripd=no
   ```

2. Reload the `frr` service:
   ```
   # systemctl reload frr
   ```

### 13.4. MODIFYING A CONFIGURATION OF A PARTICULAR DAEMON

With the default configuration, every routing daemon in FRR can only act as a plain router.

For any additional configuration of a daemon, use the following procedure.

#### Procedure

1. Within the `/etc/frr/` directory, create a configuration file for the required daemon, and name the file as follows:
   ```
   [daemon_name].conf
   ```
For example, to further configure the `eigrpd` daemon, create the `eigrpd.conf` file in the mentioned directory.

2. Populate the new file with the required content.  
   For configuration examples of particular FRR daemons, see the `/usr/share/doc/frr/` directory.

3. Reload the `frr` service:

   ```bash
   # systemctl reload frr
   ```
CHAPTER 14. PROVIDING DHCP SERVICES

The Dynamic Host Configuration Protocol (DHCP) is a network protocol that automatically assigns IP information to clients.

This section explains general information on the `dhcpd` service, as well as how to set up a DHCP server and DHCP relay.

If a procedure requires different steps for providing DHCP in IPv4 and IPv6 networks, the sections in this chapter contain procedures for both protocols.

14.1. THE DIFFERENCES WHEN USING DHCPD FOR DHCPV4 AND DHCPV6

The `dhcpd` service supports providing both DHCPv4 and DHCPv6 on one server. However, you need a separate instance of `dhcpd` with separate configuration files to provide DHCP for each protocol.

**DHCPv4**
- Configuration file: `/etc/dhcp/dhcpd.conf`
- Systemd service name: `dhcpd`

**DHCPv6**
- Configuration file: `/etc/dhcp/dhcpd6.conf`
- Systemd service name: `dhcpd6`

14.2. THE LEASE DATABASE OF THE DHCPD SERVICE

A DHCP lease is the time period for which the `dhcpd` service allocates a network address to a client. The `dhcpd` service stores the DHCP leases in the following databases:

- For DHCPv4: `/var/lib/dhcp/dhcpd.leases`
- For DHCPv6: `/var/lib/dhcp/dhcpd6.leases`

**WARNING**

Manually updating the database files can corrupt the databases.

The lease databases contain information about the allocated leases, such as the IP address assigned to a media access control (MAC) address or the time stamp when the lease expires. Note that all time stamps in the lease databases are in Coordinated Universal Time (UTC).

The `dhcpd` service recreates the databases periodically:

1. The service renames the existing files:
2. The service writes all known leases to the newly created \texttt{/var/lib/dhcpd/dhcpd.leases} and \texttt{/var/lib/dhcpd/dhcpd6.leases} files.

### Additional resources

- For further details about what is stored in the lease database, see the \texttt{dhcpd.leases(5)} man page.
- Section 14.9, “Restoring a corrupt lease database”

## 14.3. DYNAMIC IP ADDRESS ASSIGNMENT IN IPV6 NETWORKS

In an IPv6 network, only router advertisement messages provide information on an IPv6 default gateway. As a consequence, if you want to use DHCPv6 in subnets that require a default gateway setting, you must additionally configure a router advertisement service, such as Router Advertisement Daemon (\texttt{radvd}).

The \texttt{radvd} service uses flags in router advertisement packets to announce the availability of a DHCPv6 server.

This section compares DHCPv6 and \texttt{radvd}, and provides information about configuring \texttt{radvd}.

### 14.3.1. Comparison of DHCPv6 to \texttt{radvd}

<table>
<thead>
<tr>
<th></th>
<th>DHCPv6</th>
<th>\texttt{radvd}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provides information on the default gateway</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Guarantees random addresses to protect privacy</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Sends further network configuration options</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Maps media access control (MAC) addresses to IPv6 addresses</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

### 14.3.2. Configuring the \texttt{radvd} service for IPv6 routers

The router advertisement daemon (\texttt{radvd}) sends router advertisement messages that are required for IPv6 stateless autoconfiguration. This enables users to automatically configure their addresses, settings, routes, and to choose a default router based on these advertisements.

The procedure in this section explains how to configure \texttt{radvd}.

### Prerequisites

- You are logged in as the \texttt{root} user.

### Procedure

1. Install the \texttt{radvd} package:
2. Edit the `/etc/radvd.conf` file, and add the following configuration:

```
interface enp1s0
{
    AdvSendAdvert on;
    AdvManagedFlag on;
    AdvOtherConfigFlag on;

    prefix 2001:db8:0:1::/64 {
    };
};
```

These settings configure `radvd` to send router advertisement messages on the `enp1s0` device for the `2001:db8:0:1::/64` subnet. The `AdvManagedFlag on` setting defines that the client should receive the IP address from a DHCP server, and the `AdvOtherConfigFlag` parameter set to `on` defines that clients should receive non-address information from the DHCP server as well.

3. Optionally, configure that `radvd` automatically starts when the system boots:

```
# systemctl enable radvd
```

4. Start the `radvd` service:

```
# systemctl start radvd
```

5. Optionally, display the content of router advertisement packages and the configured values `radvd` sends:

```
# radvdump
```

Additional resources

- For further details about configuring `radvd`, see the `radvd.conf(5)` man page.
- For an example configuration of `radvd`, see the `/usr/share/doc/radvd/radvd.conf.example` file.

## 14.4. SETTING NETWORK INTERFACES FOR THE DHCP SERVERS

By default, the `dhcpd` service processes requests only on network interfaces that have an IP address in the subnet defined in the configuration file of the service.

For example, in the following scenario, `dhcpd` listens only on the `enp0s1` network interface:

- You have only a `subnet` definition for the `192.0.2.0/24` network in the `/etc/dhcp/dhcpd.conf` file.
- The `enp0s1` network interface is connected to the `192.0.2.0/24` subnet.
- The `enp7s0` interface is connected to a different subnet.

Only follow the procedure in this section if the DHCP server contains multiple network interfaces connected to the same network but the service should listen only on specific interfaces.
Depending on whether you want to provide DHCP for IPv4, IPv6, or both protocols, see the procedure for:

- IPv4 networks
- IPv6 networks

**Prerequisites**

- You are logged in as the root user.
- The dhcp-server package is installed.

**Procedure**

**For IPv4 networks**

1. Copy the /usr/lib/systemd/system/dhcpd.service file to the /etc/systemd/system/ directory:
   
   ```bash
   # cp /usr/lib/systemd/system/dhcpd.service /etc/systemd/system/
   ```

   Do not edit the /usr/lib/systemd/system/dhcpd.service file. Future updates of the dhcp-server package can override the changes.

2. Edit the /etc/systemd/system/dhcpd.service file, and append the names of the interface, that dhcpd should listen on to the command in the ExecStart parameter:

   ```bash
   ExecStart=/usr/sbin/dhcpd -f -cf /etc/dhcp/dhcpd.conf -user dhcpd -group dhcpd --no-pid $DHCPDARGS enp0s1 enp7s0
   ```

   This example configures that dhcpd listens only on the enp0s1 and enp7s0 interfaces.

3. Reload the systemd manager configuration:

   ```bash
   # systemctl daemon-reload
   ```

4. Restart the dhcpd service:

   ```bash
   # systemctl restart dhcpd.service
   ```

**For IPv6 networks**

1. Copy the /usr/lib/systemd/system/dhcpd6.service file to the /etc/systemd/system/ directory:

   ```bash
   # cp /usr/lib/systemd/system/dhcpd6.service /etc/systemd/system/
   ```

   Do not edit the /usr/lib/systemd/system/dhcpd6.service file. Future updates of the dhcp-server package can override the changes.

2. Edit the /etc/systemd/system/dhcpd6.service file, and append the names of the interface, that dhcpd should listen on to the command in the ExecStart parameter:

   ```bash
   ExecStart=/usr/sbin/dhcpd -f -6 -cf /etc/dhcp/dhcpd6.conf -user dhcpd -group dhcpd --no-pid $DHCPDARGS enp0s1 enp7s0
   ```

   This example configures that dhcpd listens only on the enp0s1 and enp7s0 interfaces.
3. Reload the `systemd` manager configuration:

   ```
   # systemctl daemon-reload
   ```

4. Restart the `dhcpp` service:

   ```
   # systemctl restart dhcpp.service
   ```

### 14.5. SETTING UP THE DHCP SERVICE FOR SUBNETS DIRECTLY CONNECTED TO THE DHCP SERVER

Use the following procedure if the DHCP server is directly connected to the subnet for which the server should answer DHCP requests. This is the case if a network interface of the server has an IP address of this subnet assigned.

Depending on whether you want to provide DHCP for IPv4, IPv6, or both protocols, see the procedure for:

- IPv4 networks
- IPv6 networks

#### Prerequisites

- You are logged in as the **root** user.
- The `dhcpd-server` package is installed.

#### Procedure

**For IPv4 networks**

1. Edit the `/etc/dhcp/dhcpd.conf` file:
   a. Optionally, add global parameters that `dhcpp` uses as default if no other directives contain these settings:
      ```
      option domain-name "example.com";
      default-lease-time 86400;
      ```
      This example sets the default domain name for the connection to **example.com**, and the default lease time to **86400** seconds (1 day).
   b. Add the `authoritative` statement on a new line:
      ```
      authoritative;
      ```
      **IMPORTANT**
      Without the `authoritative` statement, the `dhcpp` service does not answer **DHCPREQUEST** messages with **DCHPNACK** if a client asks for an address that is outside of the pool.
   c. For each IPv4 subnet directly connected to an interface of the server, add a `subnet` declaration:
This example adds a subnet declaration for the 192.0.2.0/24 network. With this configuration, the DHCP server assigns the following settings to a client that sends a DHCP request from this subnet:

- A free IPv4 address from the range defined in the range parameter
- IP of the DNS server for this subnet: 192.0.2.1
- Default gateway for this subnet: 192.0.2.1
- Broadcast address for this subnet: 192.0.2.255
- The maximum lease time, after which clients in this subnet release the IP and send a new request to the server: 172800 seconds (2 days)

2. Optionally, configure that **dhcpd** starts automatically when the system boots:

```
# systemctl enable dhcpd
```

3. Start the **dhcpd** service:

```
# systemctl start dhcpd
```

For IPv6 networks

1. Edit the `/etc/dhcp/dhcpd6.conf` file:

   a. Optionally, add global parameters that **dhcpd** uses as default if no other directives contain these settings:

   ```
   option dhcp6.domain-search "example.com";
   default-lease-time 86400;
   ```

   This example sets the default domain name for the connection to **example.com**, and the default lease time to **86400 seconds** (1 day).

   b. Add the **authoritative** statement on a new line:

   ```
   authoritative;
   ```

   **IMPORTANT**

   Without the **authoritative** statement, the **dhcpd** service does not answer **DHCPREQUEST** messages with **DHCPNAK** if a client asks for an address that is outside of the pool.
c. For each IPv6 subnet directly connected to an interface of the server, add a subnet declaration:

```bash
subnet6 2001:db8:0:1::/64 {
  range6 2001:db8:0:1::20 2001:db8:0:1::100;
  option dhcp6.name-servers 2001:db8:0:1::1;
  max-lease-time 172800;
}
```

This example adds a subnet declaration for the 2001:db8:0:1::/64 network. With this configuration, the DHCP server assigns the following settings to a client that sends a DHCP request from this subnet:

- A free IPv6 address from the range defined in the range6 parameter.
- The IP of the DNS server for this subnet is 2001:db8:0:1::1.
- The maximum lease time, after which clients in this subnet release the IP and send a new request to the server is 172800 seconds (2 days). Note that IPv6 requires uses router advertisement messages to identify the default gateway.

2. Optionally, configure that dhcpd6 starts automatically when the system boots:

```bash
# systemctl enable dhcpd6
```

3. Start the dhcpd6 service:

```bash
# systemctl start dhcpd6
```

Additional resources

- For a list of all parameters you can set in `/etc/dhcp/dhcpd.conf` and `/etc/dhcp/dhcpd6.conf`, see the `dhcp-options(5)` man page.

- For further details about the authoritative statement, see `The authoritative statement` section in the `dhcpd.conf(5)` man page.

- For example configurations, see the `/usr/share/doc/dhcp-server/dhcpd.conf.example` and `/usr/share/doc/dhcp-server/dhcpd6.conf.example` files.

- For details about configuring the radvd service for IPv6 router advertisement, see `Section 14.3.2, “Configuring the radvd service for IPv6 routers”`

### 14.6. Setting Up the DHCP Service for Subnets That Are Not Directly Connected to the DHCP Server

Use the following procedure if the DHCP server is not directly connected to the subnet for which the server should answer DHCP requests. This is the case if a DHCP relay agent forwards requests to the DHCP server, because none of the DHCP server’s interfaces is directly connected to the subnet the server should serve.

Depending on whether you want to provide DHCP for IPv4, IPv6, or both protocols, see the procedure for:
IPv4 networks
IPv6 networks

Prerequisites

- You are logged in as the root user.
- The dhcpd-server package is installed.

Procedure

For IPv4 networks

1. Edit the /etc/dhcp/dhcpd.conf file:
   a. Optionally, add global parameters that dhcpd uses as default if no other directives contain these settings:
      
      ```
      option domain-name "example.com";
      default-lease-time 86400;
      
      This example sets the default domain name for the connection to example.com, and the default lease time to 86400 seconds (1 day).
      ```
   
   b. Add the authoritative statement on a new line:
      
      ```
      authoritative;
      
      IMPORTANT
      Without the authoritative statement, the dhcpd service does not answer DHCPREQUEST messages with DHCPNAK if a client asks for an address that is outside of the pool.
      ```
   
   c. Add a shared-network declaration, such as the following, for IPv4 subnets that are not directly connected to an interface of the server:
      
      ```
      shared-network example {  
          option domain-name-servers 192.0.2.1;  
          ...

          subnet 192.0.2.0 netmask 255.255.255.0 {
              range 192.0.2.20 192.0.2.100;
              option routers 192.0.2.1;
          }

          subnet 198.51.100.0 netmask 255.255.255.0 {
              range 198.51.100.20 198.51.100.100;
              option routers 198.51.100.1;
          }
          ...
      }
      ```

      This example adds a shared network declaration, that contains a subnet declaration for both the 192.0.2.0/24 and 198.51.100.0/24 networks. With this configuration, the DHCP
server assigns the following settings to a client that sends a DHCP request from one of these subnets:

- The IP of the DNS server for clients from both subnets is: 192.0.2.1.
- A free IPv4 address from the range defined in the `range` parameter, depending on from which subnet the client sent the request.
- The default gateway is either 192.0.2.1 or 198.51.100.1 depending on from which subnet the client sent the request.

d. Add a `subnet` declaration for the subnet the server is directly connected to and that is used to reach the remote subnets specified in `shared-network` above:

```plaintext
subnet 203.0.113.0 netmask 255.255.255.0 {

NOTE
If the server does not provide DHCP service to this subnet, the `subnet` declaration must be empty as shown in the example. Without a declaration for the directly connected subnet, `dhcmd` does not start.
```

2. Optionally, configure that `dhcmd` starts automatically when the system boots:

```plaintext
# systemctl enable dhcmd
```

3. Start the `dhcmd` service:

```plaintext
# systemctl start dhcmd
```

For IPv6 networks

1. Edit the `/etc/dhcp/dhcmd6.conf` file:

   a. Optionally, add global parameters that `dhcmd` uses as default if no other directives contain these settings:

```plaintext
option dhcp6.domain-search "example.com";
default-lease-time 86400;
```

   This example sets the default domain name for the connection to `example.com`, and the default lease time to 86400 seconds (1 day).

   b. Add the `authoritative` statement on a new line:

```plaintext
authoritative;
```

**IMPORTANT**

Without the `authoritative` statement, the `dhcmd` service does not answer `DHCPREQUEST` messages with `DHCPNAK` if a client asks for an address that is outside of the pool.
c. Add a **shared-network** declaration, such as the following, for IPv6 subnets that are not directly connected to an interface of the server:

```
shared-network example {
  option domain-name-servers 2001:db8:0:1::1:1
  ...
  subnet6 2001:db8:0:1::1:0/120 {
    range6 2001:db8:0:1::1:20 2001:db8:0:1::1:100
  }
  subnet6 2001:db8:0:1::2:0/120 {
    range6 2001:db8:0:1::2:20 2001:db8:0:1::2:100
  }
  ...
}
```

This example adds a shared network declaration that contains a **subnet6** declaration for both the 2001:db8:0:1::1:0/120 and 2001:db8:0:1::2:0/120 networks. With this configuration, the DHCP server assigns the following settings to a client that sends a DHCP request from one of these subnets:

- The IP of the DNS server for clients from both subnets is **2001:db8:0:1::1:1**.
- A free IPv6 address from the range defined in the **range6** parameter, depending on from which subnet the client sent the request.
  Note that IPv6 requires uses router advertisement messages to identify the default gateway.

d. Add a **subnet6** declaration for the subnet the server is directly connected to and that is used to reach the remote subnets specified in **shared-network** above:

```
subnet6 2001:db8:0:1::50:0/120 {
}
```

**NOTE**

If the server does not provide DHCP service to this subnet, the **subnet6** declaration must be empty as shown in the example. Without a declaration for the directly connected subnet, **dhcpd** does not start.

2. Optionally, configure that **dhcpd6** starts automatically when the system boots:

```
# systemctl enable dhcpd6
```

3. Start the **dhcpd6** service:

```
# systemctl start dhcpd6
```

Additional resources

- For a list of all parameters you can set in `/etc/dhcp/dhcpd.conf` and `/etc/dhcp/dhcpd6.conf`, see the **dhcp-options(5)** man page.
14.7. ASSIGNING A STATIC ADDRESS TO A HOST USING DHCP

Using a host declaration, you can configure the DHCP server to assign a fixed IP address to a media access control (MAC) address of a host. For example, use this method to always assign the same IP address to a server or network device.

**IMPORTANT**

If you configure a fixed IP address for a MAC address, the IP address must be outside of the address pool you specified in the fixed-address and fixed-address6 parameters.

Depending on whether you want to configure fixed addresses for IPv4, IPv6, or both protocols, see the procedure for:

- IPv4 networks
- IPv6 networks

**Prerequisites**

- The dhcpd service is configured and running.
- You are logged in as the root user.

**Procedure**

For IPv4 networks

1. Edit the /etc/dhcp/dhcpd.conf file:
   
   a. Add a host declaration:

   ```
   host server.example.com {
   hardware ethernet 52:54:00:72:2f:6e;
   fixed-address 192.0.2.130;
   }
   ```

   This example configures the DHCP server to always assigns the 192.0.2.130 IP address to the host with the 52:54:00:72:2f:6e MAC address.

   The dhcpd service identifies systems by the MAC address specified in the fixed-address parameter, and not by the name in the host declaration. As a consequence, you can set this name to any string that does not match other host declarations. To configure the same system for multiple networks, use a different name, otherwise, dhcpd fails to start.

   b. Optionally, add further settings to the host declaration that are specific for this host.
2. Restart the `dhcpd` service:

```
# systemctl start dhcpd
```

For IPv6 networks

1. Edit the `/etc/dhcp/dhcpd6.conf` file:
   a. Add a `host` declaration:
      ```
      host server.example.com {
        hardware ethernet 52:54:00:72:2f:6e;
        fixed-address6 2001:db8:0:1::200;
      }
      ```
      This example configures the DHCP server to always assign the `2001:db8:0:1::20` IP address to the host with the `52:54:00:72:2f:6e` MAC address.

      The `dhcpd` service identifies systems by the MAC address specified in the `fixed-address6` parameter, and not by the name in the `host` declaration. As a consequence, you can set this name to any string, as long as it is unique to other `host` declarations. To configure the same system for multiple networks, use a different name because, otherwise, `dhcpd` fails to start.

   b. Optionally, add further settings to the `host` declaration that are specific for this host.

2. Restart the `dhcpd6` service:

```
# systemctl start dhcpd6
```

Additional resources

- For a list of all parameters you can set in `/etc/dhcp/dhcpd.conf` and `/etc/dhcp/dhcpd6.conf`, see the `dhcp-options(5)` man page.
- For example configurations, see the `/usr/share/doc/dhcp-server/dhcpd.conf.example` and `/usr/share/doc/dhcp-server/dhcpd6.conf.example` files.

### 14.8. USING A GROUP DECLARATION TO APPLY PARAMETERS TO MULTIPLE HOSTS, SUBNETS, AND SHARED NETWORKS AT THE SAME TIME

Using a `group` declaration, you can apply the same parameters to multiple hosts, subnets, and shared networks.

Note that the procedure in this section describes using a `group` declaration for hosts, but the steps are the same for subnets and shared networks.

Depending on whether you want to configure a group for IPv4, IPv6, or both protocols, see the procedure for:

- IPv4 networks
- IPv6 networks

Prerequisites
The **dhcpd** service is configured and running.

You are logged in as the **root** user.

**Procedure**

**For IPv4 networks**

1. Edit the `/etc/dhcp/dhcpd.conf` file:
   
   a. Add a **group** declaration:

   ```
   group {
     option domain-name-servers 192.0.2.1;
     
     host server1.example.com {
       hardware ethernet 52:54:00:72:2f:6e;
       fixed-address 192.0.2.130;
     }
     
     host server2.example.com {
       hardware ethernet 52:54:00:1b:f3:cf;
       fixed-address 192.0.2.140;
     }
   }
   ```

   This **group** definition groups two **host** entries. The **dhcpd** service applies the value set in the **option domain-name-servers** parameter to both hosts in the group.

   b. Optionally, add further settings to the **group** declaration that are specific for these hosts.

2. Restart the **dhcpd** service:

   ```
   # systemctl start dhcpd
   ```

**For IPv6 networks**

1. Edit the `/etc/dhcp/dhcpd6.conf` file:

   a. Add a **group** declaration:

   ```
   group {
     option dhcp6.domain-search "example.com";
     
     host server1.example.com {
       hardware ethernet 52:54:00:72:2f:6e;
       fixed-address 2001:db8:0:1::200;
     }
     
     host server2.example.com {
       hardware ethernet 52:54:00:1b:f3:cf;
       fixed-address 2001:db8:0:1::ba3;
     }
   }
   ```

   This **group** definition groups two **host** entries. The **dhcpd** service applies the value set in the **option dhcp6.domain-search** parameter to both hosts in the group.
b. Optionally, add further settings to the group declaration that are specific for these hosts.

2. Restart the dhcpd6 service:

```
# systemctl start dhcpd6
```

**Additional resources**

- For a list of all parameters you can set in `/etc/dhcp/dhcpd.conf` and `/etc/dhcp/dhcpd6.conf`, see the `dhcp-options(5)` man page.

- For example configurations, see the `/usr/share/doc/dhcp-server/dhcpd.conf.example` and `/usr/share/doc/dhcp-server/dhcpd6.conf.example` files.

### 14.9. RESTORING A CORRUPT LEASE DATABASE

If the DHCP server logs an error that is related to the lease database, such as **Corrupt lease file - possible data loss!**, you can restore the lease database from the copy the `dhcppd` service created. Note that this copy might not reflect the latest status of the database.

**WARNING**

If you remove the lease database instead of replacing it with a backup, you lose all information about the currently assigned leases. As a consequence, the DHCP server could assign leases to clients that have been previously assigned to other hosts and are not expired yet. This leads to IP conflicts.

Depending on whether you want to restore the DHCPv4, DHCPv6, or both databases, see the procedure for:

- Restoring the DHCPv4 lease database
- Restoring the DHCPv6 lease database

**Prerequisites**

- You are logged in as the **root** user.
- The lease database is corrupt.

**Procedure**

**Restoring the DHCPv4 lease database**

1. Stop the `dhcppd` service:

```
# systemctl stop dhcppd
```

2. Rename the corrupt lease database:

```
# mv /var/lib/dhcp/dhcppd.leases /var/lib/dhcp/dhcppd.leases.corrupt
```
3. Restore the copy of the lease database that the `dhcp` service created when it refreshed the lease database:

```bash
# cp -p /var/lib/dhcpd/dhcpd.leases~ /var/lib/dhcpd/dhcpd.leases
```

**IMPORTANT**

If you have a more recent backup of the lease database, restore this backup instead.

4. Start the `dhcpd` service:

```bash
# systemctl start dhcpd
```

### Restoring the DHCPv6 lease database

1. Stop the `dhcppd` service:

```bash
# systemctl stop dhcpd6
```

2. Rename the corrupt lease database:

```bash
# mv /var/lib/dhcpd/dhcpd6.leases /var/lib/dhcpd/dhcpd6.leases.corrupt
```

3. Restore the copy of the lease database that the `dhcp` service created when it refreshed the lease database:

```bash
# cp -p /var/lib/dhcpd/dhcpd6.leases~ /var/lib/dhcpd/dhcpd6.leases
```

**IMPORTANT**

If you have a more recent backup of the lease database, restore this backup instead.

4. Start the `dhcppd` service:

```bash
# systemctl start dhcppd
```

### Additional resources

- Section 14.2, “The lease database of the dhcppd service”

### 14.10. SETTING UP A DHCP RELAY AGENT

The DHCP Relay Agent (`dhcrelay`) enables the relay of DHCP and BOOTP requests from a subnet with no DHCP server on it to one or more DHCP servers on other subnets. When a DHCP client requests information, the DHCP Relay Agent forwards the request to the list of DHCP servers specified. When a DHCP server returns a reply, the DHCP Relay Agent forwards this request to the client.

Depending on whether you want to set up a DHCP relay for IPv4, IPv6, or both protocols, see the procedure for:
Prerequisites

- You are logged in as the **root** user.

Procedure

For **IPv4 networks**

1. Install the **dhcp-relay** package:
   
   ```bash
   # yum install dhcp-relay
   ```

2. Copy the `/lib/systemd/system/dhcrelay.service` file to the `/etc/systemd/system/` directory:
   
   ```bash
   # cp /lib/systemd/system/dhcrelay.service /etc/systemd/system/
   ```

   Do not edit the `/usr/lib/systemd/system/dhcrelay.service` file. Future updates of the `dhcp-relay` package can override the changes.

3. Edit the `/etc/systemd/system/dhcrelay.service` file, and append the `-i interface` parameter, together with a list of IP addresses of DHCPv4 servers that are responsible for the subnet:
   
   ```bash
   ExecStart=/usr/sbin/dhcrelay -d --no-pid -i enp1s0 192.0.2.1
   ```

   With these additional parameters, `dhcrelay` listens for DHCPv4 requests on the `enp1s0` interface and forwards them to the DHCP server with the IP `192.0.2.1`.

4. Reload the **systemd** manager configuration:
   
   ```bash
   # systemctl daemon-reload
   ```

5. Optionally, configure that the **dhcrelay** service starts when the system boots:
   
   ```bash
   # systemctl enable dhcrelay.service
   ```

6. Start the **dhcrelay** service:
   
   ```bash
   # systemctl start dhcrelay.service
   ```

For **IPv6 networks**

1. Install the **dhcp-relay** package:
   
   ```bash
   # yum install dhcp-relay
   ```

2. Copy the `/lib/systemd/system/dhcrelay.service` file to the `/etc/systemd/system/` directory and name the file `dhcrelay6.service`:
   
   ```bash
   # cp /lib/systemd/system/dhcrelay.service /etc/systemd/system/dhcrelay6.service
   ```
Do not edit the `/usr/lib/systemd/system/dhcrelay.service` file. Future updates of the `dhcrelay` package can override the changes.

3. Edit the `/etc/systemd/system/dhcrelay6.service` file, and append the `-l receiving_interface` and `-u outgoing_interface` parameters:

   ```
   ExecStart=/usr/sbin/dhcrelay -d --no-pid -l enp1s0 -u enp7s0
   ```

With these additional parameters, `dhcrelay` listens for DHCPv6 requests on the `enp1s0` interface and forwards them to the network connected to the `enp7s0` interface.

4. Reload the `systemd` manager configuration:

   ```
   # systemctl daemon-reload
   ```

5. Optionally, configure that the `dhcrelay6` service starts when the system boots:

   ```
   # systemctl enable dhcrelay6.service
   ```

6. Start the `dhcrelay6` service:

   ```
   # systemctl start dhcrelay6.service
   ```

**Additional resources**

- For further details about `dhcrelay`, see the `dhcrelay(8)` man page.

**Additional resources**

- Section 14.1, “The differences when using dhcpcd for DHCPv4 and DHCPv6”
CHAPTER 15. 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.

15.1. GETTING STARTED WITH FIREWALLD

15.1.1. 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.

Additional resources

- *firewalld(1)* man page

15.1.2. 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*
- *firewall-config* tool
- *firewall-cmd* command-line tool
- The RHEL web console

The latter three can only edit the appropriate *NetworkManager* configuration files. If you change the zone of the interface using the web console, *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 default settings of the predefined zones are as follows:

- **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.

Additional resources
`firewalld.zone(5)` man page

15.1.3. 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
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.

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.

Additional resources

- firewalld.service(5) man page

15.2. INSTALLING THE FIREWALL-CONFIG GUI CONFIGURATION TOOL

To use the firewall-config GUI configuration tool, install the firewall-config package.

Procedure

1. Enter the following command as root:

   # yum install firewall-config

   Alternatively, in GNOME, use the Super key and type `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.

2. 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.

15.3. VIEWING THE CURRENT STATUS AND SETTINGS OF FIREWALLD

15.3.1. 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.

Procedure

1. To see the status of the service:

   # firewall-cmd --state

2. 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 proc
   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

Additional resources
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 15.3.2, “Viewing current firewalld settings”

15.3.2. Viewing current firewalld settings

15.3.2.1. 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.

15.3.2.2. 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:
```

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
```
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

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

```
# firewall-cmd --list-services
```

```
ssh dhcpv6-client
```

**NOTE**
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.

### 15.4. STARTING FIREWALLD

**Procedure**

1. To start `firewalld`, enter the following command as **root**:

   ```
   # systemctl unmask firewalld
   # systemctl start firewalld
   ```

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

   ```
   # systemctl enable firewalld
   ```

### 15.5. STOPPING FIREWALLD
Procedure

1. To stop firewalld, enter the following command as root:
   # systemctl stop firewalld

2. To prevent firewalld from starting automatically at system start:
   # systemctl disable firewalld

3. To make sure firewalld is not started by accessing the firewalld D-Bus interface and also if other services require firewalld:
   # systemctl mask firewalld

15.6. 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.

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.
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
```

### 15.7. VERIFYING THE PERMANENT FIREWALLD CONFIGURATION

In certain situations, for example after manually editing firewalld configuration files, administrators want to verify that the changes are correct. This section describes how to verify the permanent configuration of the firewalld service.

**Prerequisites**

- The firewalld service is running.

**Procedure**

1. Verify the permanent configuration of the firewalld service:

   ```
   # firewall-cmd --check-config
   success
   ```

   If the permanent configuration is valid, the command returns success. In other cases, the command returns an error with further details, such as the following:

   ```
   # firewall-cmd --check-config
   Error: INVALID_PROTOCOL: 'public.xml': 'tcpx' not from {'tcp'|'udp'|'sctp'|'dccp'}
   ```

### 15.8. CONTROLLING NETWORK TRAFFIC USING FIREWALLD

#### 15.8.1. 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.

**Procedure**

1. 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
```

### 15.8.2. 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.

**Procedure**

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 ...
   ```
   
   [trimmed for clarity]

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
   ```

### 15.8.3. Controlling traffic with predefined services using GUI

To enable or disable a predefined or custom service:

1. Start the `firewall-config` tool and select the network zone whose services are to be configured.

2. Select the **Services** tab.

3. Select the check box for each type of service you want to trust or clear the check box to block a service.

To edit a service:

1. Start the `firewall-config` tool.

2. Select **Permanent** from the menu labeled **Configuration**. Additional icons and menu buttons appear at the bottom of the **Services** window.
3. 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.

### 15.8.4. 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.

**Procedure**

To add a new service in a terminal, use `firewall-cmd`, or `firewall-offline-cmd` in case of not active `firewalld`.

1. Enter the following command to add a new and empty service:

   ```bash
   $ firewall-cmd --new-service=service-name
   ```

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

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

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

3. 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:

   ```bash
   # 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 overridden 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.

### 15.8.5. 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.
15.8.5.1. 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.

Procedure

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

1. List all allowed ports:
   
   ```bash
   # firewall-cmd --list-ports
   ```

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

   ```bash
   # firewall-cmd --add-port=port-number/port-type
   ```

3. Make the new settings persistent:

   ```bash
   # firewall-cmd --runtime-to-permanent
   ```

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

15.8.5.2. 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.

Procedure

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

1. List all allowed ports:

   ```bash
   # 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:

   ```bash
   # firewall-cmd --remove-port=port-number/port-type
   ```

3. Make the new settings persistent:

   ```bash
   # firewall-cmd --runtime-to-permanent
   ```

15.8.6. Opening ports using GUI

To permit traffic through the firewall to a certain port:
1. Start the **firewall-config** tool and select the network zone whose settings you want to change.

2. Select the **Ports** tab and click the **Add** button on the right-hand side. The **Port and Protocol** window opens.

3. Enter the port number or range of ports to permit.

4. Select **tcp** or **udp** from the list.

**15.8.7. Controlling traffic with protocols using GUI**

To permit traffic through the firewall using a certain protocol:

1. Start the **firewall-config** tool and select the network zone whose settings you want to change.

2. Select the **Protocols** tab and click the **Add** button on the right-hand side. The **Protocol** window opens.

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

**15.8.8. Opening source ports using GUI**

To permit traffic through the firewall from a certain port:

1. Start the firewall-config tool and select the network zone whose settings you want to change.

2. Select the **Source Port** tab and click the **Add** button on the right-hand side. The **Source Port** window opens.

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

**15.9. WORKING WITH FIREWALLD 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.

**15.9.1. Listing zones**

**Procedure**

1. To see which zones are available on your system:

   ```bash
   # 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.

2. To see detailed information for all zones:

   ```bash
   # firewall-cmd --list-all-zones
   ```

3. To see detailed information for a specific zone:
15.9.2. Modifying firewalld settings for a certain zone

The Section 15.8.2, “Controlling traffic with predefined services using CLI” and Section 15.8.5, “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.

Procedure
1. 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
   ```

15.9.3. 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.

Procedure
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.

15.9.4. 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.

Procedure
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:
15.9.5. 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.

Procedure

1. 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
```

15.9.6. 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.

Procedure

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
```

15.9.7. Zone configuration files

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
<?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.

Additional resources

- For more information, see the `firewalld.zone` manual pages.

### 15.9.8. 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.

**Procedure**

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>
   ```

### 15.10. USING ZONES TO MANAGE INCOMING TRAFFIC DEPENDING ON A SOURCE

#### 15.10.1. 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.

#### 15.10.2. 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.

- To set the source in the current zone:
  
  ```
  # firewall-cmd --add-source=<source>
  ```

- 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:

**Procedure**

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
   ```

**15.10.3. Removing a source**

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

**Procedure**

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
   ```

**15.10.4. 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.

**Procedure**
1. To add a source port:

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

### 15.10.5. Removing a source port

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

**Procedure**

1. To remove a source port:

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

### 15.10.6. 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. The following procedure allows traffic from `192.168.1.0/24` to be able to reach the **HTTP** service while any other traffic is blocked.

**Procedure**

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
```

### 15.10.7. 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.
15.10.7.1. 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.

Procedure

1. 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.

15.10.7.2. 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.

Procedure

1. To remove a protocol from a zone:

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

15.11. CONFIGURING IP ADDRESS MASQUERADING

The following procedure describes how to enable IP masquerading on your system. IP masquerading hides individual machines behind a gateway when accessing the Internet.

Procedure

1. 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.

2. To enable IP masquerading, enter the following command as `root`:

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

3. 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 --permanent
   ```

15.12. PORT FORWARDING
Redirecting ports using this method only works for IPv4-based traffic. For IPv6 redirecting setup, you must use rich rules.

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

15.12.1. Adding a port to redirect

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.

Prerequisites

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

Procedure

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
```

15.12.2. Redirecting TCP port 80 to port 88 on the same machine

Follow the steps to redirect the TCP port 80 to port 88.

Procedure

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
```

15.12.3. Removing a redirected port

Red Hat Enterprise Linux 8 Configuring and managing networking
To remove a redirected port:

```bash
~# 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, use the following procedure.

**Procedure**

1. Remove the forwarded port:

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

2. Disable masquerade:

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

### 15.12.4. Removing TCP port 80 forwarded to port 88 on the same machine

To remove the port redirection:

**Procedure**

1. List redirected ports:

```bash
~# firewall-cmd --list-forward-ports
```

```
port=80:proto=tcp:toport=88:toaddr=
```

2. Remove the redirected port from the firewall:

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

3. Make the new settings persistent:

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

### 15.13. 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.

#### 15.13.1. Listing and blocking ICMP requests

**Listing ICMP requests**

- **To list** the ICMP requests:
  ```bash
  ~> firewall-cmd --list-icmp
  ```

- **To block** all incoming **ICMP** requests:
  ```bash
  ~> firewall-cmd --block-all-icmp
  ```

- **To block** specific **ICMP** requests:
  ```bash
  ~> firewall-cmd --block-icmp=type=type-number
  ```

- **To unblock** specific **ICMP** requests:
  ```bash
  ~> firewall-cmd --unblock-icmp=type=type-number
  ```

- **To unblock** all incoming **ICMP** requests:
  ```bash
  ~> firewall-cmd --unblock-all-icmp
  ```
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>

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>

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 the block inversion 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
   ```

### 15.13.2. 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.

15.14. 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:net,net hash:net,port hash:net,port,net
```


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
  ```

  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
  ```

  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`:

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

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

```bash
# 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`:

```bash
# 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`:

```bash
# 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`:

```bash
# 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`:

```bash
# 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.
15.15. PRIORITIZING RICH RULES

By default, rich rules are organized based on their rule action. For example, deny rules have precedence over allow rules. The priority parameter in rich rules provides administrators fine-grained control over rich rules and their execution order.

15.15.1. How the priority parameter organizes rules into different chains

You can set the priority parameter in a rich rule to any number between \(-32768\) and \(32767\), and lower values have higher precedence.

The firewalld service organizes rules based on their priority value into different chains:

- Priority lower than 0: the rule is redirected into a chain with the \(_pre\) suffix.
- Priority higher than 0: the rule is redirected into a chain with the \(_post\) suffix.
- Priority equals 0: based on the action, the rule is redirected into a chain with the \(_log\), \(_deny\), or \(_allow\) the action.

Inside these sub-chains, firewalld sorts the rules based on their priority value.

15.15.2. Setting the priority of a rich rule

The procedure describes an example of how to create a rich rule that uses the priority parameter to log all traffic that is not allowed or denied by other rules. You can use this rule to flag unexpected traffic.

Procedure

1. Add a rich rule with a very low precedence to log all traffic that has not been matched by other rules:

   ```
   # firewall-cmd --add-rich-rule='rule priority=32767 log prefix="UNEXPECTED: " limit value="5/m"
   ```

   The command additionally limits the number of log entries to 5 per minute.

2. Optionally, display the nftables rule that the command in the previous step created:

   ```
   # nft list chain inet firewalld filter_IN_public_post
   table inet firewalld {
     chain filter_IN_public_post {
   ```
15.16. 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.

15.16.1. Configuring lockdown with using CLI

- 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

15.16.2. Configuring lockdown whitelist options using CLI

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.

### 15.16.3. Configuring lockdown whitelist options using 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/libexec/platform-python -s /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.

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 entered 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 are 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 match. If the * is not there then the absolute command including arguments must match.
15.17. LOG 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.

15.18. RELATED INFORMATION

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

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`.

Online documentation

CHAPTER 16. GETTING STARTED WITH NFTABLES

The nftables framework enables administrators to configure packet-filtering rules used by the Linux kernel firewall.

16.1. 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 rule set
- support for debugging and tracing in the rule set (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.

Effect of the modules on the nftables rules set can be observed using the nft list rule set command. Since these tools add tables, chains, rules, sets, and other objects to the nftables rule set, be aware that nftables rule-set operations, such as the nft flush ruleset command, might affect rule sets installed using the formerly separate legacy commands.

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.

16.2. WHEN TO USE FIREWALLD, NFTABLES, OR IPTABLES

The following is a brief overview in which scenario you should use one of the following utilities:

- firewalld: Use the firewalld utility to configure a firewall on workstations. The utility is easy to use and covers the typical use cases for this scenario.
- nftables: Use the nftables utility to set up complex firewalls, such as for a whole network.
- iptables: The iptables utility is deprecated in Red Hat Enterprise Linux 8. Use instead nftables.

16.3. CONVERTING IPTABLES RULES TO NFTABLES RULES

Red Hat Enterprise Linux 8 provides the iptables-translate and ip6tables-translate tools to convert existing iptables or ip6tables rules into the equivalent ones for nftables.
Note that some extensions lack translation support. If such an extension exists, the tool prints the untranslated rule prefixed with the # sign. For example:

```
# iptables-translate -A INPUT -j CHECKSUM --checksum-fill
nft # -A INPUT -j CHECKSUM --checksum-fill
```

Additionally, users can use the `iptables-restore-translate` and `ip6tables-restore-translate` tools to translate a dump of rules. Note that before that, users can use the `iptables-save` or `ip6tables-save` commands to print a dump of current rules. For example:

```
# iptables-save >/tmp/iptables.dump
# iptables-restore-translate -f /tmp/iptables.dump

# Translated by iptables-restore-translate v1.8.0 on Wed Oct 17 17:00:13 2018
add table ip nat
...
```

For more information and a list of possible options and values, enter the `iptables-translate --help` command.

### 16.4. WRITING AND EXECUTING NFTABLES SCRIPTS

The `nftables` framework provides a native scripting environment that brings a major benefit over using shell scripts to maintain firewall rules: the execution of scripts is atomic. This means that the system either applies the whole script or prevents the execution if an error occurs. This guarantees that the firewall is always in a consistent state.

Additionally, the `nftables` script environment enables administrators to:

- add comments
- define variables
- include other rule set files

This section explains how to use these features, as well as creating and executing `nftables` scripts.

When you install the `nftables` package, Red Hat Enterprise Linux automatically creates `*.nft` scripts in the `/etc/nftables/` directory. These scripts contain commands that create tables and empty chains for different purposes. You can either extend these files or write your scripts.

#### 16.4.1. The required script header in nftables script

Similar to other scripts, `nftables` scripts require a shebang sequence in the first line of the script that sets the interpreter directive.

An `nftables` script must always start with the following line:

```
#!/usr/sbin/nft -f
```

**IMPORTANT**

If you omit the `-f` parameter, the `nft` utility does not read the script and displays **Error:** syntax error, unexpected newline, expecting string.
16.4.2. Supported nftables script formats

The nftables scripting environment supports scripts in the following formats:

- You can write a script in the same format as the nft list ruleset command displays the rule set:

```
#!/usr/sbin/nft -f

# Flush the rule set
flush ruleset

table inet example_table {
    chain example_chain {
        # Chain for incoming packets that drops all packets that
        # are not explicitly allowed by any rule in this chain
        type filter hook input priority 0; policy drop;

        # Accept connections to port 22 (ssh)
        tcp dport ssh accept
    }
}
```

- You can use the same syntax for commands as in nft commands:

```
#!/usr/sbin/nft -f

# Flush the rule set
flush ruleset

# Create a table
add table inet example_table

# Create a chain for incoming packets that drops all packets
# that are not explicitly allowed by any rule in this chain
add chain inet example_table example_chain { type filter hook input priority 0; policy drop ; }

# Add a rule that accepts connections to port 22 (ssh)
add rule inet example_table example_chain tcp dport ssh accept
```

16.4.3. Running nftables scripts

To run an nftables script, the script must be executable. Only if the script is included in another script, it does not require to be executable. The procedure describes how to make a script executable and run the script.

**Prerequisites**

- The procedure of this section assumes that you stored an nftables script in the /etc/nftables/example_firewall.nft file.

**Procedure**

1. Steps that are required only once:
   a. Optionally, set the owner of the script to root:
b. Make the script executable for the owner:

```
# chmod u+x /etc/nftables/example_firewall.nft
```

2. Run the script:

```
# /etc/nftables/example_firewall.nft
```

If no output is displayed, the system executed the script successfully.

**IMPORTANT**

Even if `nft` executes the script successfully, incorrectly placed rules, missing parameters, or other problems in the script can cause that the firewall behaves not as expected.

### Additional resources

- For details about setting the owner of a file, see the `chown(1)` man page.
- For details about setting permissions of a file, see the `chmod(1)` man page.
- Section 16.4.7, "Automatically loading nftables rules when the system boots"

#### 16.4.4. Using comments in nftables scripts

The `nftables` scripting environment interprets everything to the right of a `#` character as a comment.

**Example 16.1. Comments in an nftables script**

Comments can start at the beginning of a line, as well as next to a command:

```
...  
# Flush the rule set
flush ruleset

add table inet example_table  # Create a table
...  
```

#### 16.4.5. Using variables in an nftables script

To define a variable in an `nftables` script, use the `define` keyword. You can store single values and anonymous sets in a variable. For more complex scenarios, use sets or verdict maps.

**Variables with a single value**

The following example defines a variable named `INET_DEV` with the value `enp1s0`:

```nftables
define INET_DEV = enp1s0
```
You can use the variable in the script by writing the $ sign followed by the variable name:

```
... add rule inet example_table example_chain ifname $INET_DEV tcp dport ssh accept ...
```

Variables that contain an anonymous set

The following example defines a variable that contains an anonymous set:

```
define DNS_SERVERS = { 192.0.2.1, 192.0.2.2 }
```

You can use the variable in the script by writing the $ sign followed by the variable name:

```
add rule inet example_table example_chain ip daddr $DNS_SERVERS accept
```

**NOTE**

Note that curly braces have special semantics when you use them in a rule because they indicate that the variable represents a set.

Additional resources

- For details about sets, see Section 16.10, “Using sets in nftables commands”.
- For details about verdict maps, see Section 16.11, “Using verdict maps in nftables commands”.

### 16.4.6. Including files in an nftables script

The *nftables* scripting environment enables administrators to include other scripts by using the `include` statement.

If you specify only a file name without an absolute or relative path, *nftables* includes files from the default search path, which is set to `/etc` on Red Hat Enterprise Linux.

**Example 16.2. Including files from the default search directory**

To include a file from the default search directory:

```
include "example.nft"
```

**Example 16.3. Including all *.nft files from a directory**

To include all files ending in *.nft that are stored in the `/etc/nftables/rulesets/` directory:

```
include "/etc/nftables/rulesets/*.nft"
```

Note that the `include` statement does not match files beginning with a dot.

Additional resources
• For further details, see the include files section in the nft(8) man page.

16.4.7. Automatically loading nftables rules when the system boots

The nftables systemd service loads firewall scripts that are included in the /etc/sysconfig/nftables.conf file. This section explains how to load firewall rules when the system boots.

Prerequisites

• The nftables scripts are stored in the /etc/nftables/ directory.

Procedure

1. Edit the /etc/sysconfig/nftables.conf file.

   • If you enhance *.nft scripts created in /etc/nftables/ when you installed the nftables package, uncomment the include statement for these scripts.

   • If you write scripts from scratch, add include statements to include these scripts. For example, to load the /etc/nftables/example.nft script when the nftables service starts, add:

     ```
     include "/etc/nftables/example.nft"
     ```

2. Enable the nftables service.

   ```
   # systemctl enable nftables
   ```

3. Optionally, start the nftables service to load the firewall rules without rebooting the system:

   ```
   # systemctl start nftables
   ```

Additional resources

• Section 16.4.2, “Supported nftables script formats”

16.5. DISPLAYING NFTABLES RULE SETS

Rule sets of nftables contain tables, chains, and rules. This section explains how to display these rule sets.

Procedure

1. To display all rule sets, enter:

   ```
   # nft list ruleset
   table inet example_table {
   chain example_chain {
     type filter hook input priority 0; policy accept;
     tcp dport http accept
     tcp dport ssh accept
   }
   }
   ```
NOTE

By default, nftables does not pre-create tables. As a consequence, displaying the rule set on a host without any tables, the nft list ruleset command shows no output.

16.6. CREATING AN NFTABLES TABLE

A table in nftables is a name space that contains a collection of chains, rules, sets, and other objects. This section explains how to create a table.

Each table must have an address family defined. The address family of a table defines what address types the table processes. You can set one of the following address families when you create a table:

- **ip**: Matches only IPv4 packets. This is the default if you do not specify an address family.
- **ip6**: Matches only IPv6 packets.
- **inet**: Matches both IPv4 and IPv6 packets.
- **arp**: Matches IPv4 address resolution protocol (ARP) packets.
- **bridge**: Matches packets that traverse a bridge device.
- **netdev**: Matches packets from ingress.

**Procedure**

1. Use the nft add table command to create a new table. For example, to create a table named example_table that processes IPv4 and IPv6 packets:

   ```
   # nft add table inet example_table
   ```

2. Optionally, list all tables in the rule set:

   ```
   # nft list tables
   table inet example_table
   ```

**Additional resources**

- For further details about address families, see the Address families section in the nft(8) man page.
- For details on other actions you can run on tables, see the Tables section in the nft(8) man page.

16.7. CREATING AN NFTABLES CHAIN

Chains are containers for rules. The following two rule types exist:

- **Base chain**: You can use base chains as an entry point for packets from the networking stack.
- **Regular chain**: You can use regular chains as a jump target and to better organize rules.

The procedure describes how to add a base chain to an existing table.
Prerequisites

- The table to which you want to add the new chain exists.

Procedure

1. Use the `nft add chain` command to create a new chain. For example, to create a chain named `example_chain` in `example_table`:

   ```
   # nft add chain inet example_table example_chain { type filter hook input priority 0; policy accept; }
   ```

   **IMPORTANT**
   
   To avoid that the shell interprets the semicolons as the end of the command, you must escape the semicolons with a backslash.

   This chain filters incoming packets. The `priority` parameter specifies the order in which `nftables` processes chains with the same hook value. A lower priority value has precedence over higher ones. The `policy` parameter sets the default action for rules in this chain. Note that if you are logged in to the server remotely and you set the default policy to `drop`, you are disconnected immediately if no other rule allows the remote access.

2. Optionally, display all chains:

   ```
   # nft list chains
   table inet example_table {
   chain example_chain {
   type filter hook input priority 0; policy accept;
   }
   }
   ```

Additional resources

- For further details about address families, see the `Address families` section in the `nft(8)` man page.

- For details on other actions you can run on chains, see the `Chains` section in the `nft(8)` man page.

16.8. ADDING A RULE TO AN NFTABLES CHAIN

This section explains how to add a rule to an existing `nftables` chain. By default, the `nftables add rule` command appends a new rule to the end of the chain.

If you instead want to insert a rule at the beginning of chain, see Section 16.9, “Inserting a rule into an nftables chain”.

Prerequisites

- The chain to which you want to add the rule exists.

Procedure

1. To add a new rule, use the `nft add rule` command. For example, to add a rule to the `example_chain` in the `example_table` that allows TCP traffic on port 22:

   ```
   # nft add rule inet example_table example_chain { dport 22 protocol tcp accept; }
   ```
**example_chain** in the **example_table** that allows TCP traffic on port 22:

```
# nft add rule inet example_table example_chain tcp dport 22 accept
```

Instead of the port number, you can alternatively specify the name of the service. In the example, you could use `ssh` instead of the port number 22. Note that a service name is resolved to a port number based on its entry in the `/etc/services` file.

2. Optionally, display all chains and their rules in **example_table**:

```
# nft list table inet example_table
table inet example_table {
  chain example_chain {
    type filter hook input priority 0; policy accept;
    ...
    tcp dport ssh accept
  }
}
```

**Additional resources**

- For further details about address families, see the **Address families** section in the `nft(8)` man page.
- For details on other actions you can run on rules, see the **Rules** section in the `nft(8)` man page.

### 16.9. INSERTING A RULE INTO AN NFTABLES CHAIN

This section explains how to insert a rule at the beginning of an existing **nftables** chain using the `nftables insert rule` command. If you instead want to add a rule to the end of a chain, see [Section 16.8, “Adding a rule to an nftables chain”](#).

**Prerequisites**

- The chain to which you want to add the rule exists.

**Procedure**

1. To insert a new rule, use the `nft insert rule` command. For example, to insert a rule to the **example_chain** in the **example_table** that allows TCP traffic on port 22:

```
# nft add rule inet example_table example_chain tcp dport 22 accept
```

You can alternatively specify the name of the service instead of the port number. In the example, you could use `ssh` instead of the port number 22. Note that a service name is resolved to a port number based on its entry in the `/etc/services` file.

2. Optionally, display all chains and their rules in **example_table**:

```
# nft list table inet example_table
table inet example_table {
  chain example_chain {
    type filter hook input priority 0; policy accept;
    tcp dport ssh accept
  }
}
```
Additional resources

- For further details about address families, see the Address families section in the nft(8) man page.
- For details on other actions you can run on rules, see the Rules section in the nft(8) man page.

16.10. USING SETS IN NFTABLES COMMANDS

The nftables framework natively supports sets. You can use sets, for example, if a rule should match multiple IP addresses, port numbers, interfaces, or any other match criteria.

16.10.1. Using an anonymous sets in nftables

An anonymous set contain comma-separated values enclosed in curly brackets, such as \{ 22, 80, 443 \}, that you use directly in a rule. You can also use anonymous sets also for IP addresses or any other match criteria.

The drawback of anonymous sets is that if you want to change the set, you must replace the rule. For a dynamic solution, use named sets as described in Section 16.10.2, “Using named sets in nftables”.

Prerequisites

- The example_chain chain and the example_table table in the inet family exists.

Procedure

1. For example, to add a rule to example_chain in example_table that allows incoming traffic to port 22, 80, and 443:

   ```
   # nft add rule inet example_table example_chain tcp dport { 22, 80, 443 } accept
   ```

2. Optionally, display all chains and their rules in example_table:

   ```
   # nft list table inet example_table
   table inet example_table {
     chain example_chain {
       type filter hook input priority 0; policy accept;
       tcp dport { ssh, http, https } accept
     }
   }
   ```

16.10.2. Using named sets in nftables

The nftables framework supports mutable named sets. A named set is a list or range of elements that you can use in multiple rules within a table. Another benefit over anonymous sets is that you can update a named set without replacing the rules that use the set.

When you create a named set, you must specify the type of elements the set contains. You can set the following types:
- **ipv4_addr** for a set that contains IPv4 addresses or ranges, such as 192.0.2.1 or 192.0.2.0/24.
- **ipv6_addr** for a set that contains IPv6 addresses or ranges, such as 2001:db8::1 or 2001:db8::1/24.
- **ether_addr** for a set that contains a list of media access control (MAC) addresses, such as 52:54:00:6b:66:42.
- **inet_proto** for a set that contains a list of internet protocol types, such as tcp.
- **inet_service** for a set that contains a list of internet services, such as ssh.
- **mark** for a set that contains a list of packet marks. Packet marks can be any positive 32-bit integer value (0 to 2147483647).

**Prerequisites**
- The **example_chain** chain and the **example_table** table exists.

**Procedure**

1. Create an empty set. The following examples create a set for IPv4 addresses:
   - To create a set that can store multiple individual IPv4 addresses:
     ```
     # nft add set example_table example_set { type ipv4_addr; }
     ```
   - To create a set that can store IPv4 address ranges:
     ```
     # nft add set example_table example_set { type ipv4_addr; flags interval; }
     ```
   
   **IMPORTANT**
   To avoid that the shell interprets the semicolons as the end of the command, you must escape the semicolons with a backslash.

2. Optionally, create rules that use the set. For example, the following command adds a rule to the **example_chain** in the **example_table** that will drop all packets from IPv4 addresses in **example_set**.
   ```
   # nft add rule example_table example_chain ip saddr @example_set drop
   ```
   Because **example_set** is still empty, the rule has currently no effect.

3. Add IPv4 addresses to **example_set**:
   - If you create a set that stores individual IPv4 addresses, enter:
     ```
     # nft add element example_table example_set { 192.0.2.1, 192.0.2.2 }
     ```
   - If you create a set that stores IPv4 ranges, enter:
     ```
     # nft add element example_table example_set { 192.0.2.0-192.0.2.255 }
     ```
When you specify an IP address range, you can alternatively use the Classless Inter-Domain Routing (CIDR) notation, such as \texttt{192.0.2.0/24} in the above example.

\textbf{Related information}

- For further details about sets, see the \texttt{Sets} section in the \texttt{nft(8)} man page.

\section*{16.11. USING VERDICT MAPS IN NFTABLES COMMANDS}

Verdict maps, which are also known as dictionaries, enable \texttt{nft} to perform an action based on packet information by mapping match criteria to an action.

\subsection*{16.11.1. Using literal maps in nftables}

A literal map is a \{ \texttt{match_criteria : action} \} statement that you use directly in a rule. The statement can contain multiple comma-separated mappings.

The drawback of a literal map is that if you want to change the map, you must replace the rule. For a dynamic solution, use named verdict maps as described in Section 16.11.2, "Using mutable verdict maps in nftables".

The example describes how to use a literal map to route both TCP and UDP packets of the IPv4 and IPv6 protocol to different chains to count incoming TCP and UDP packets separately.

\textbf{Procedure}

1. Create the \texttt{example_table}:
   \begin{verbatim}
   # nft add table inet example_table
   \end{verbatim}

2. Create the \texttt{tcp_packets} chain in \texttt{example_table}:
   \begin{verbatim}
   # nft add chain inet example_table tcp_packets
   \end{verbatim}

3. Add a rule to \texttt{tcp_packets} that counts the traffic in this chain:
   \begin{verbatim}
   # nft add rule inet example_table tcp_packets counter
   \end{verbatim}

4. Create the \texttt{udp_packets} chain in \texttt{example_table}
   \begin{verbatim}
   # nft add chain inet example_table udp_packets
   \end{verbatim}

5. Add a rule to \texttt{udp_packets} that counts the traffic in this chain:
   \begin{verbatim}
   # nft add rule inet example_table udp_packets counter
   \end{verbatim}

6. Create a chain for incoming traffic. For example, to create a chain named \texttt{incoming_traffic} in \texttt{example_table} that filters incoming traffic:
   \begin{verbatim}
   # nft add chain inet example_table incoming_traffic { type filter hook input priority 0 \; }
   \end{verbatim}

7. Add a rule with a literal map to \texttt{incoming_traffic}:
# nft add rule inet example_table incoming_traffic ip protocol vmap { tcp : jump tcp_packets, udp : jump udp_packets }

The literal map distinguishes the packets and sends them to the different counter chains based on their protocol.

8. To list the traffic counters, display `example_table`:

```
# nft list table inet example_table
table inet example_table {
  chain tcp_packets {
    counter packets 36379 bytes 2103816
  }
  chain udp_packets {
    counter packets 10 bytes 1559
  }
  chain incoming_traffic {
    type filter hook input priority 0; policy accept;
    ip protocol vmap { tcp : jump tcp_packets, udp : jump udp_packets }
  }
}
```

The counters in the `tcp_packets` and `udp_packets` chain display both the number of received packets and bytes.

16.11.2. Using mutable verdict maps in nftables

The `nftables` framework supports mutable verdict maps. You can use these maps in multiple rules within a table. Another benefit over literal maps is that you can update a mutable map without replacing the rules that use it.

When you create a mutable verdict map, you must specify the type of elements

- `ipv4_addr` for a map whose match part contains an IPv4 address, such as `192.0.2.1`.
- `ipv6_addr` for a map whose match part contains an IPv6 address, such as `2001:db8::1`.
- `ether_addr` for a map whose match part contains a media access control (MAC) address, such as `52:54:00:6b:66:42`.
- `inet_proto` for a map whose match part contains an internet protocol type, such as `tcp`.
- `inet_service` for a map whose match part contains an internet services name port number, such as `ssh` or `22`.
- `mark` for a map whose match part contains a packet mark. A packet mark can be any positive 32-bit integer value (0 to `2147483647`).
- `counter` for a map whose match part contains a counter value. The counter value can be any positive 64-bit integer value.
- `quota` for a map whose match part contains a quota value. The quota value can be any positive 64-bit integer value.
The example describes how to allow or drop incoming packets based on their source IP address. Using a mutable verdict map, you require only a single rule to configure this scenario while the IP addresses and actions are dynamically stored in the map. The procedure also describes how to add and remove entries from the map.

**Procedure**

1. Create a table. For example, to create a table named `example_table` that processes IPv4 packets:
   ```
   # nft add table ip example_table
   ```

2. Create a chain. For example, to create a chain named `example_chain` in `example_table`:
   ```
   # nft add chain ip example_table example_chain { type filter hook input priority 0 ; }
   ```

   **IMPORTANT**
   
   To avoid that the shell interprets the semicolons as the end of the command, you must escape the semicolons with a backslash.

3. Create an empty map. For example, to create a map for IPv4 addresses:
   ```
   # nft add map ip example_table example_map { type ipv4_addr : verdict ; }
   ```

4. Create rules that use the map. For example, the following command adds a rule to `example_chain` in `example_table` that applies actions to IPv4 addresses which are both defined in `example_map`:
   ```
   # nft add rule example_table example_chain ip saddr vmap @example_map
   ```

5. Add IPv4 addresses and corresponding actions to `example_map`:
   ```
   # nft add element ip example_table example_map { 192.0.2.1 : accept, 192.0.2.2 : drop }
   ```

   This example defines the mappings of IPv4 addresses to actions. In combination with the rule created above, the firewall accepts packet from 192.0.2.1 and drops packets from 192.0.2.2.

6. Optionally, enhance the map by adding another IP address and action statement:
   ```
   # nft add element ip example_table example_map { 192.0.2.3 : accept }
   ```

7. Optionally, remove an entry from the map:
   ```
   # nft delete element ip example_table example_map { 192.0.2.1 }
   ```

8. Optionally, display the rule set:
   ```
   # nft list ruleset
table ip example_table {
  map example_map {
  type ipv4_addr : verdict
  elements = { 192.0.2.2 : drop, 192.0.2.3 : accept }
}
16.12. LIMITING THE NUMBER OF CONNECTIONS USING NFTABLES

The `ct count` parameter of the `nft` utility enables administrators to limit the number of connections. The procedure describes a basic example of how to limit incoming connections.

**Prerequisites**
- The base `example_chain` in `example_table` exists.

**Procedure**

1. Add a rule that allows only two simultaneous connections to the SSH port (22) from an IPv4 address and rejects all further connections from the same IP:

   ```
   # nft add rule ip example_table example_chain tcp dport ssh meter example_meter { ip saddr ct count over 2 } counter reject
   ```

2. Optionally, display the meter created in the previous step:

   ```
   # nft list meter ip example_table example_meter
   table ip example_table {
     meter example_meter {
       type ipv4_addr
       size 65535
       elements = { 192.0.2.1 : ct count over 2 , 192.0.2.2 : ct count over 2 } 
     }
   }
   ```

   The `elements` entry displays addresses that currently match the rule. In this example, `elements` lists IP addresses that have active connections to the SSH port. Note that the output does not display the number of active connections or if connections were rejected.

16.13. DEBUGGING NFTABLES RULES

The `nftables` framework provides different options for administrators to debug rules and if packets match them. This section describes these options.

16.13.1. Creating a rule with a counter

To identify if a rule is matched, you can use a counter. This section describes how to create a new rule with a counter.
For a procedure that adds a counter to an existing rule, see Section 16.13.2, “Adding a counter to an existing rule”.

Prerequisites
- The chain to which you want to add the rule exists.

Procedure
1. Add a new rule with the `counter` parameter to the chain. The following example adds a rule with a counter that allows TCP traffic on port 22 and counts the packets and traffic that match this rule:
   ```
   # nft add rule inet example_table example_chain tcp dport 22 counter accept
   ```

2. To display the counter values:
   ```
   # nft list ruleset
   table inet example_table {
   chain example_chain {
   type filter hook input priority 0; policy accept;
   tcp dport ssh counter packets 6872 bytes 10544565 accept
   }
   }
   ```

16.13.2. Adding a counter to an existing rule
To identify if a rule is matched, you can use a counter. This section describes how to add a counter to an existing rule.

For a procedure to add a new rule with a counter, see Section 16.13.1, “Creating a rule with a counter”.

Prerequisites
- The rule to which you want to add the counter exists.

Procedure
1. Display the rules in the chain including their handles:
   ```
   # nft --handle list chain inet example_table example_chain
   table inet example_table {
   chain example_chain {
   type filter hook input priority 0; policy accept;
   tcp dport ssh accept # handle 4
   }
   }
   ```

2. Add the counter by replacing the rule but with the `counter` parameter. The following example replaces the rule displayed in the previous step and adds a counter:
   ```
   # nft replace rule inet example_table example_chain handle 4 tcp dport 22 counter accept
   ```

3. To display the counter values:
# nft list ruleset
table inet example_table {
  chain example_chain {
    type filter hook input priority 0; policy accept;
    tcp dport ssh counter packets 6872 bytes 105448565 accept
  }
}

16.13.3. Monitoring packets that match an existing rule

The tracing feature in nftables in combination with the nft monitor command enables administrators to display packets that match a rule. The procedure describes how to enable tracing for a rule as well as monitoring packets that match this rule.

Prerequisites

- The rule to which you want to add the counter exists.

Procedure

1. Display the rules in the chain including their handles:

   ```
   # nft --handle list chain inet example_table example_chain
   table inet example_table {
     chain example_chain { # handle 1
       type filter hook input priority 0; policy accept;
       tcp dport ssh accept # handle 4
     }
   }
   ```

2. Add the tracing feature by replacing the rule but with the meta nftrace set 1 parameters. The following example replaces the rule displayed in the previous step and enables tracing:

   ```
   # nft replace rule inet example_table example_chain handle 4 tcp dport 22 meta nftrace set 1 accept
   ```

3. Use the nft monitor command to display the tracing. The following example filters the output of the command to display only entries that contain inet example_table example_chain:

   ```
   # nft monitor | grep "inet example_table example_chain"
   trace id 3c5eb15e inet example_table example_chain packet: iif "enp1s0" ether saddr 52:54:00:17:ff:e4 ether daddr 52:54:00:72:2f:6e ip saddr 192.0.2.1 ip daddr 192.0.2.2 ip dscp cs0 ip ecn not-ect ip ttl 64 ip id 49710 ip protocol tcp ip length 60 tcp sport 56728 tcp dport ssh tcp flags == syn tcp window 64240
   trace id 3c5eb15e inet example_table example_chain rule tcp dport ssh nftrace set 1 accept (verdict accept)
   ```
   ...
   
   Red Hat Enterprise Linux 8 Configuring and managing networking
WARNING

Depending on the number of rules with tracing enabled and the amount of matching traffic, the nft monitor command can display a lot of output. Use grep or other utilities to filter the output.

16.14. BACKING UP AND RESTORING NFTABLES RULE SETS

This section describes how to backup nftables rules to a file, as well as restoring rules from a file. Administrators can use a file with the rules to, for example, transfer the rules to a different server.

16.14.1. Backing up nftables rule sets to a file

This section describes how to back up nftables rule sets to a file.

Procedure

1. To backup nftables rules:
   - In nft list ruleset format:
     # nft list ruleset > file.nft
   - In JSON format:
     # nft -j list ruleset > file.json

16.14.2. Restoring nftables rule sets from a file

This section describes how to restore nftables rule sets.

Procedure

1. To restore nftables rules:
   - If the file to restore is in nft list ruleset format or contains nft commands:
     # nft -f file.nft
   - If the file to restore is in JSON format:
     # nft -j -f file.json

16.15. RELATED INFORMATION

- The Using nftables in Red Hat Enterprise Linux 8 blog post provides an overview about using nftables features.
• 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 back end for firewalld.
CHAPTER 17. GETTING STARTED WITH DPDK

The Data Plane Development Kit (DPDK) provides libraries and network drivers to accelerate package processing in user space.

Administrators use DPDK, for example, in virtual machines to use Single Root I/O Virtualization (SR-IOV) to reduce latencies and increase I/O throughput.

NOTE

Red Hat does not support experimental DPDK APIs.

17.1. INSTALLING THE DPDK PACKAGE

This section describes how to install the dpdk package.

Prerequisites

- Red Hat Enterprise Linux is installed.
- A valid subscription is assigned to the host.

Procedure

1. Use the yum utility to install the dpdk package:

   ```
   # yum install dpdk
   ```

Related information

- For a list of network adapters that support SR-IOV on Red Hat Enterprise Linux 8, see Network Adapter Fast Datapath Feature Support Matrix.