Fedora 24

Networking Guide

Configuration and Administration of Networking for Fedora 24

Stephen Wadeley

Fedora 24 Networking Guide
Configuration and Administration of Networking for Fedora 24
Edition 1.0

Author Stephen Wadeley `swadeley@redhat.com`

The *Networking Guide* documents relevant information regarding the configuration and administration of network interfaces, networks and network services in Fedora 24. It is oriented towards system administrators with a basic understanding of Linux and networking.

This book is based on the *Deployment Guide* from Red Hat Enterprise Linux 6. The chapters related to networking were taken from the Deployment Guide to form the foundation for this book.

Preface

The *Networking Guide* contains information on how to use the networking related features of Fedora 24.

This manual discusses many intermediate topics such as the following:

- Setting up a network (from establishing an Ethernet connection using **NetworkManager** to configuring channel bonding interfaces).
- Configuring `DHCP`, **BIND**, and `DNS`.

Note

Some of the graphical procedures and menu locations are specific to GNOME, but most command line instructions will be universally applicable.

1. Target Audience

The *Networking Guide* assumes you have a basic understanding of the Fedora operating system. If you need help with the installation of this system, see the *Fedora Installation Guide*[1].

This guide is not aimed exclusively at experienced Linux system administrators. The authors of this book have attempted to cater for a wider audience as more and more organizations and users become subscribers to Red Hat, Inc. At the same time we are aware of the need not to allow seemingly trivial information to get in the way of the tasks. Your feedback on how well we have met this goal is welcome.

2. About This Book

The *Networking Guide* is based on the networking material in the *Red Hat Enterprise Linux 6 Deployment Guide*[2]. It also retains the information on `DHCP` and `DNS` servers from the *Part II, "Servers"* section. Most of the non-networking related material from the *Red Hat Enterprise Linux 6 Deployment Guide* guide can now be found in the *Fedora 24 System Administrator's Guide*[3] except for reference material, such as that found in the appendices of the Deployment Guide.

3. How to Read this Book

This manual is divided into the following main categories:

Part I, "Networking"

This part describes how to configure the network on Fedora.

Chapter 1, Introduction to Fedora Networking explains the default networking service, **NetworkManager**, and the various graphical and command-line tools that can be used to interact with **NetworkManager**. These include, an associated command-line configuration tool, **nmcli**, and

[1] http://docs.fedoraproject.org/install-guide
[2] https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/index.html
[3] https://docs.fedoraproject.org/sysadmin-guide

two graphical user interface tools, **control-center** and **nm-connection-editor**. Read this chapter to learn more about the many ways the **NetworkManager** daemon can be used.

Chapter 2, Configure Networking covers static and dynamic interface settings, selecting network configuration methods, using **NetworkManager** with graphical and command-line user interfaces. Read this chapter to learn about configuring network connections.

Chapter 3, Configure Host Names covers static, pretty, and transient host names and their configuration using **hostnamectl**. Read this chapter to learn more about configuring host names on local and remote systems.

Chapter 4, Configure Network Bonding covers the configuring and use of bonded network connections. Read this chapter to learn about the configuring of network bonds using graphical and command-line user interfaces.

Chapter 5, Configure Network Teaming covers the configuring and use of teamed network connections. Read this chapter to learn about the configuring of network teams using graphical and command-line user interfaces.

Chapter 6, Configure Network Bridging covers the configuring and use of network bridges. Read this chapter to learn about the configuring of network bridges using graphical and command-line user interfaces.

Chapter 7, Configure 802.1Q VLAN tagging covers the configuring and use of virtual private networks, VLANs, according to the 802.1Q standard. Read this chapter to learn about the configuring of VLANs using graphical and command-line user interfaces.

Chapter 8, Consistent Network Device Naming covers consistent network device naming for network interfaces, a feature that changes the name of network interfaces on a system in order to make locating and differentiating the interfaces easier. Read this chapter to learn about this feature and how to enable or disable it.

Part II, “Servers”

This part discusses how to set up servers normally required for networking.

Chapter 9, DHCP Servers covers the installation of a Dynamic Host Configuration Protocol (`DHCP`) server and client. Read this chapter if you need to configure `DHCP` on your system.

Chapter 10, DNS Servers covers the Domain Name System (`DNS`), explains how to install, configure, run, and administer the **BIND** `DNS` server. Read this chapter if you need to configure a `DNS` server on your system.

For topics related to network configuration but not listed above, such as configuring GRUB to enable serial links and the use of virtual console terminals, see the *Fedora 24 System Administrator's Guide*[4].

For topics related to servers but not listed above, such as configuring Network Time Protocol (`NTP`) and Precision Time Protocol (`PTP`), see the *Fedora 24 System Administrator's Guide*[5].

4. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

[4] https://docs.fedoraproject.org/sysadmin-guide
[5] https://docs.fedoraproject.org/sysadmin-guide

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[6] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

4.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to the first virtual terminal. Press **`Ctrl`**+**`Alt`**+**`F1`** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-

[6] https://fedorahosted.org/liberation-fonts/

> click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

`Mono-spaced Bold Italic` or ***`Proportional Bold Italic`***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **`ssh`** ***`username@domain.name`*** at a shell prompt. If the remote machine is **`example.com`** and your username on that machine is john, type **`ssh john@example.com`**.
>
> The **`mount -o remount`** ***`file-system`*** command remounts the named file system. For example, to remount the **`/home`** file system, the command is **`mount -o remount /home`**.
>
> To see the version of a currently installed package, use the **`rpm -q`** ***`package`*** command. It will return a result as follows: ***`package-version-release`***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

4.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **`mono-spaced roman`** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **`mono-spaced roman`** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();
```

```
        System.out.println("Created Echo");

        System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
    }
}
```

4.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

5. Feedback

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in *Bugzilla*[7] against the product **Fedora Documentation**.

When submitting a bug report, be sure to provide the following information:

- Manual's identifier: **`networking-guide`**
- Version number: **24**

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

[7] http://bugzilla.redhat.com/

6. Acknowledgments

Certain portions of this text first appeared in the *Red Hat Enterprise Linux 6 Deployment Guide*, copyright © 2010—YEAR Red Hat, Inc., available at *https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/index.html*.

Part I. Networking

This part describes how to configure the network on Fedora.

Introduction to Fedora Networking

1.1. How this Book is Structured

All new material in this book has been written and arranged in such a way as to clearly separate introductory material, such as explanations of concepts and use cases, from configuration tasks. The authors hope that you can quickly find configuration instructions you need, while still providing some relevant explanations and conceptual material to help you understand and decide on the appropriate tasks relevant to your needs. Where material has been reused from the *Red Hat Enterprise Linux 6 Deployment Guide*[1], it has been reviewed and changed, where possible, to fit this idea of separating concepts from tasks.

The material is grouped according to the goal rather than the method. Instructions on how to achieve a specific task using different methods are grouped together. This is intended to make it easier for you to find the information on how to achieve a particular task or goal, and at the same time allow you to quickly see the different methods available.

In each chapter, the configuration methods will be presented in the following order: A graphical user interface (GUI) method, such as the use of **nm-connection-editor** or **control-network** to direct `NetworkManager`, then **NetworkManager**'s command-line tool **nmcli**, and then finally methods using the command line and configuration files. The command line can be used to issue commands and to compose or edit configuration files, therefore the use of the **ip** commands and configuration files will be documented together.

1.2. Introduction to NetworkManager

As of Fedora 20, the default networking service is provided by `NetworkManager`, which is a dynamic network control and configuration daemon that attempts to keep network devices and connections up and active when they are available. The traditional **`ifcfg`** type configuration files are still supported. See *Section 1.6, “NetworkManager and the Network Scripts”* for more information.

Table 1.1. A Summary of Networking Tools and Applications

Application or Tool	Description
NetworkManager	The default networking daemon
nmtui	A simple curses-based text user interface (TUI) for **NetworkManager**
nmcli	A command-line tool provided to allow users and scripts to interact with **NetworkManager**
control-center	A graphical user interface tool provided by the GNOME Shell
nm-connection-editor	A GTK+ 3 application available for certain tasks not yet handled by **control-center**

`NetworkManager` can be used with the following types of connections: Ethernet, VLANs, Bridges, Bonds, Teams, Wi-Fi, mobile broadband (such as cellular 3G), and IP-over-InfiniBand. For these connection types, `NetworkManager` can configure network aliases, `IP` addresses, static routes, `DNS` information, and VPN connections, as well as many connection-specific parameters. Finally, `NetworkManager` provides an API via D-Bus which allows applications to query and control network configuration and state.

[1] https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/index.html

1.3. Installing NetworkManager

NetworkManager is installed by default on Fedora. If necessary, to ensure that it is, run the following command as the `root` user:

```
~]# dnf install NetworkManager
```

For information on user privileges and gaining privileges, see the *Fedora 24 System Administrator's Guide*[2].

1.3.1. The NetworkManager Daemon

The **NetworkManager** daemon runs with root privileges and is, by default, configured to start up at boot time. You can determine whether the **NetworkManager** daemon is running by entering this command:

```
~]$ 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
```

The **`systemctl status`** command will report **NetworkManager** as **`Active: inactive (dead)`** if the **NetworkManager** service is not running. To start it for the current session run the following command as the root user:

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

Run the **`systemctl enable`** command to ensure that **NetworkManager** starts up every time the system boots:

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

For more information on starting, stopping and managing services, see the *Fedora 24 System Administrator's Guide*[3].

1.3.2. Interacting with NetworkManager

Users do not interact with the **NetworkManager** system service directly. Instead, users perform network configuration tasks via graphical and command-line user interface tools. The following tools are available in Fedora:

1. A command-line tool, **nmcli**, is provided to allow users and scripts to interact with **NetworkManager**. Note that **nmcli** can be used on GUI-less systems like servers to control all aspects of **NetworkManager**. It is on an equal footing with the GUI tools.

2. A graphical user interface tool called **control-center**, provided by the GNOME Shell, is available for desktop users. It incorporates a **Network** settings tool. To start it, press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **`Super`** key appears in a variety of guises, depending on the keyboard and other hardware, but often as either the Windows or Command key, and typically to the left of the Spacebar.

[2] https://docs.fedoraproject.org/sysadmin-guide
[3] https://docs.fedoraproject.org/sysadmin-guide

3. The GNOME Shell also provides a network icon in its Notification Area 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. On other desktops the network icon is provided by **NetworkManager Applet**.

4. A graphical user interface tool, **nm-connection-editor**, is available for certain specialized tasks not handled by **control-center**. To start it using the GNOME shell, press the **`Super`** key to enter the Activities Overview, type **`network connections`** or **`nm-connection-editor`** and then press **`Enter`**. Most desktop environments follow the convention of using the key combination **`Alt`**+**`F2`** for opening new applications. For all desktops, press the **`Alt`**+**`F2`** key and enter **nm-connection-editor**.

1.4. Network Configuration Using the Command Line Interface (CLI)

The commands for the **ip** utility, sometimes referred to as *iproute2* after the upstream package name, are documented in the **`man ip(8)`** page. The package name in Fedora is *iproute*. If necessary, you can check that the **ip** utility is installed by checking its version number as follows:

```
~]$ ip -V
ip utility, iproute2-ss130716
```

The **ip** commands can be used to add and remove addresses and routes to interfaces in parallel with **NetworkManager**, which will preserve them and recognize them in **nmcli**, **nmtui**, **control-center**, and the D-Bus API.

Note

Note that **ip** commands given on the command line will not persist after a system restart.

Examples of using the command line and configuration files for each task are included after explaining the use of one of the graphical user interfaces to **NetworkManager**, namely, **control-center** and **nm-connection-editor**.

1.5. Network Configuration Using NetworkManager's CLI (nmcli)

The **NetworkManager** command-line tool, **nmcli**, provides a command line way to configure networking by controlling **NetworkManager**. It is installed, along with **NetworkManager**, by default. If required, for details on how to verify that **NetworkManager** is running, see *Section 1.3.1, “The NetworkManager Daemon”*.

Examples of using the **nmcli** tool for each task will be included where possible, after explaining the use of graphical user interfaces and other command line methods. See *Section 2.4, “Using the NetworkManager Command Line Tool, nmcli”* for an introduction to **nmcli**.

1.6. NetworkManager and the Network Scripts

In previous Red Hat Enterprise Linux releases, the default way to configure networking was using *network scripts*. The term *network scripts* is commonly used for the script **`/etc/init.d/network`** and any other installed scripts it calls. The user supplied files are typically viewed as configuration, but can also be interpreted as an amendment to the scripts.

Although **NetworkManager** provides the default networking service, Red Hat developers have worked hard to ensure that scripts and **NetworkManager** cooperate with each other. Administrators who are used to the scripts can certainly continue to use them. We expect both systems to be able to run in parallel and work well together. It is expected that most user shell scripts from previous releases will still work. Red Hat recommends that you test them first.

Running Network Script

Run the script **only** with the **systemctl** utility which will clear any existing environment variables and ensure clean execution. The command takes the following form:

```
systemctl start|stop|restart|status network
```

Do not run any service by calling **`/etc/init.d/`*`servicename`*` start|stop|restart|status`** directly.

Note that in Red Hat Enterprise Linux 7, **NetworkManager** is started first, and **`/etc/init.d/network`** checks with **NetworkManager** to avoid tampering with **NetworkManager**'s connections. **NetworkManager** is intended to be the primary application using sysconfig configuration files and **`/etc/init.d/network`** is intended to be secondary, playing a fallback role.

The **`/etc/init.d/network`** script is not event-driven, it runs either:

1. manually (by one of the **`systemctl`** commands **`start|stop|restart network`**),
2. on boot and shutdown if the network service is enabled (as a result of the command **`systemctl enable network`**).

It is a manual process and does not react to events that happen after boot. Users can also call the scripts **`ifup`** and **`ifdown`** manually.

Custom Commands and the Network Scripts

Custom commands in the scripts **`/sbin/ifup-local`**, **`ifdown-pre-local`**, and **`ifdown-local`** are only executed when those devices are controlled by the `/etc/init.d/network` service. If you modified the initscripts themselves (for example, **`/etc/sysconfig/network-scripts/ifup-eth`**) then those changes would be overwritten by an *initscripts* package update. Therefore it is recommend that you avoid modifying the initscripts directly and make use of the **`/sbin/if*local`** scripts, so that your custom changes will survive package updates. The initscripts just check for the presence of the relevant **`/sbin/if*local`** and run them if they exit. The initscripts do not place anything in the **`/sbin/if*local`** scripts, nor does the *initscripts* RPM (or any package) own or modify those files.

There are ways to perform custom tasks when network connections go up and down, both with the old network scripts and with **NetworkManager**. When **NetworkManager** is enabled, the **`ifup`** and **`ifdown`** script will ask **NetworkManager** whether **NetworkManager** manages the interface in question, which is found from the “DEVICE=” line in the **`ifcfg`** file. If **NetworkManager** does manage that device, and the device is not already connected, then **`ifup`** will ask **NetworkManager** to start the connection.

- If the device is managed by **NetworkManager** and it **is** already connected, nothing is done.

- If the device is not managed by **NetworkManager**, then the scripts will start the connection using the older, non-**NetworkManager** mechanisms that they have used since the time before **NetworkManager** existed.

If you are calling "**`ifdown`**" and the device is managed by **NetworkManager**, then **`ifdown`** will ask **NetworkManager** to terminate the connection.

The scripts dynamically check **NetworkManager**, so if **NetworkManager** is not running, the scripts will fall back to the old, pre-**NetworkManager** script-based mechanisms.

1.7. Network Configuration Using 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 **`/etc/NetworkManager/`** subdirectories. Interface specific information for example, is stored in **`ifcfg`** files in the **`/etc/sysconfig/network-scripts/`** directory.

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

In Fedora, when 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, then **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 be told to read the configuration files again. To do that, issue the following command as `root`:

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

The above command reads all connection profiles. Alternatively, to reload only one changed file, **`ifcfg-`*`ifname`***, issue a command as follows:

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

The command accepts multiple file names. These commands require `root` privileges. For more information on user privileges and gaining privileges, see the *Fedora 24 System Administrator's Guide*[4] and the **`su(1)`** and **`sudo(8)`** man pages.

Changes made using tools such as **nmcli** do not require a reload but do require the associated interface to be put down and then up again. That can be done by using commands in the following format:

```
nmcli dev disconnect interface-name
```

Followed by:

```
nmcli con up interface-name
```

NetworkManager does not trigger any of the network scripts, though the network scripts will try to trigger **NetworkManager** if it is running when **`ifup`** commands are used. See *Section 1.6, "NetworkManager and the Network Scripts"* for an explanation of the network scripts.

[4] https://docs.fedoraproject.org/sysadmin-guide

The **`ifup`** script is a generic script which does a few things and then calls interface-specific scripts like **`ifup-ethX`**, **`ifup-wireless`**, **`ifup-ppp`**, and so on. When a user runs **`ifup eth0`** manually, the following occurs:

1. **`ifup`** looks for a file called **`/etc/sysconfig/network-scripts/ifcfg-eth0`**;
2. if the **`ifcfg`** file exists, **`ifup`** looks for the **`TYPE`** key in that file to determine which type-specific script to call;
3. **`ifup`** calls **`ifup-wireless`** or **`ifup-eth`** or **`ifup-XXX`** based on **`TYPE`**;
4. the type-specific scripts do type-specific setup;
5. and then the type-specific scripts let common functions perform `IP`-related tasks like `DHCP` or static setup.

On bootup, **`/etc/init.d/network`** reads through all the **`ifcfg`** files and for each one that has **`ONBOOT=yes`**, it checks whether **NetworkManager** is already starting the DEVICE from that **`ifcfg`** file. If **NetworkManager** is starting that device or has already started it, nothing more is done for that file, and the next **`ONBOOT=yes`** file is checked. If **NetworkManager** is not yet starting that device, the initscripts will continue with their traditional behavior and call **`ifup`** for that **`ifcfg`** file.

The end result is that any **`ifcfg`** file that has **`ONBOOT=yes`** is expected to be started on system bootup, either by **NetworkManager** or by the initscripts. This ensures that some legacy network types which **NetworkManager** does not handle (such as ISDN or analog dialup modems) as well as any new application not yet supported by **NetworkManager** are still correctly started by the initscripts even though **NetworkManager** is unable to handle them.

Note

It is recommended not to store backup **`ifcfg`** files in the same location as the live ones. The script literally does **`ifcfg-*`** with an exclude only for these extensions: **`.old`**, **`.orig`**, **`.rpmnew`**, **`.rpmorig`**, and **`.rpmsave`**. The best way is not to store backup files anywhere within the **`/etc/`** directory.

1.8. Additional Resources

The following sources of information provide additional resources regarding networking for Fedora.

1.8.1. Installed Documentation

- **`man(1)`** man page — Describes man pages and how to find them.
- **`NetworkManager(8)`** man page — Describes the network management daemon.
- **`NetworkManager.conf(5)`** man page — Describes the `NetworkManager` configuration file.
- **`/usr/share/doc/initscripts-`*`version`*`/sysconfig.txt`** — Describes configuration files and their directives.

Configure Networking

2.1. Static and Dynamic Interface Settings

When to use static addressing and when to use dynamic addressing? These decisions are subjective, they depend on your accessed needs, your specific requirements. Having a policy, documenting it, and applying it consistently are usually more important than the specific decisions you make. In a traditional company LAN, this is an easier decision to make as you typically have fewer servers than other hosts. Provisioning and installation tools make providing static configurations to new hosts easy and using such tools will change your work flow and requirements. The following two sections are intended to provide guidance to those who have not already been through this decision-making process. For more information on automated configuration and management, see the **OpenLMI** section in the *Fedora 24 System Administrator's Guide*[1]. The *Fedora 24 System Installation Guide*[2] documents the use of **kickstart** which can also be used for automating the assignment of network settings.

2.1.1. When to Use Static Network Interface Settings

Use static `IP` addressing on those servers and devices whose network availability you want to ensure when automatic assignment methods, such as `DHCP`, fail. `DHCP`, `DNS`, and authentication servers are typical examples. Interfaces for out-of-band management devices are also worth configuring with static settings as these devices are supposed to work, as far as is possible, independently of other network infrastructure.

For hosts which are not considered vital, but for which static `IP` addressing is still considered desirable, use an automated provisioning method when possible. For example, `DHCP` servers can be configured to provide the same `IP` address to the same host every time. This method could be used for communal printers for example.

All the configuration tools listed in *Section 2.1.3, “Selecting Network Configuration Methods”* allow assigning static `IP` addresses manually. The **nmcli** tool is also suitable for use with scripted assignment of network configuration.

2.1.2. When to Use Dynamic Interface Settings

Enable and use dynamic assignment of `IP` addresses and other network information whenever there is no compelling reason not to. The time saved in planning and documenting manual settings can be better spent elsewhere. The *dynamic host control protocol* (DHCP) is a traditional method of dynamically assigning network configurations to hosts. See *Section 9.1, “Why Use DHCP?”* for more information on this subject.

NetworkManager will by default call the `DHCP` client, **dhclient**, when a profile has been set to obtain addresses automatically, or when an interface configuration file has BOOTPROTO set to **dhcp**. Where `DHCP` is required, an instance of `dhclient` is started for every Internet protocol, `IPv4` and `IPv6`, on an interface. Where **NetworkManager** is not running, or not managing an interface, then the legacy network service will call instances of `dhclient` as required.

[1] https://docs.fedoraproject.org/sysadmin-guide
[2] https://docs.fedoraproject.org/install-guide

2.1.3. Selecting Network Configuration Methods

- To configure a network using graphical user interface tools, proceed to *Section 2.2, “Using NetworkManager with the GNOME Graphical User Interface”*
- To configure a network interface manually, see *Section 2.3, “Using the Command Line Interface (CLI)”*.
- To configure an interface using **NetworkManager**'s command-line tool, **nmcli**, proceed to *Section 2.4, “Using the NetworkManager Command Line Tool, nmcli”*

2.2. Using NetworkManager with the GNOME Graphical User Interface

As of **Fedora 20**, **NetworkManager**'s own graphical user interface (GUI) is not used by default. The **Network** settings configuration tool is provided as part of the new GNOME **control-center** GUI. The old **nm-connection-editor** GUI is still available for certain tasks.

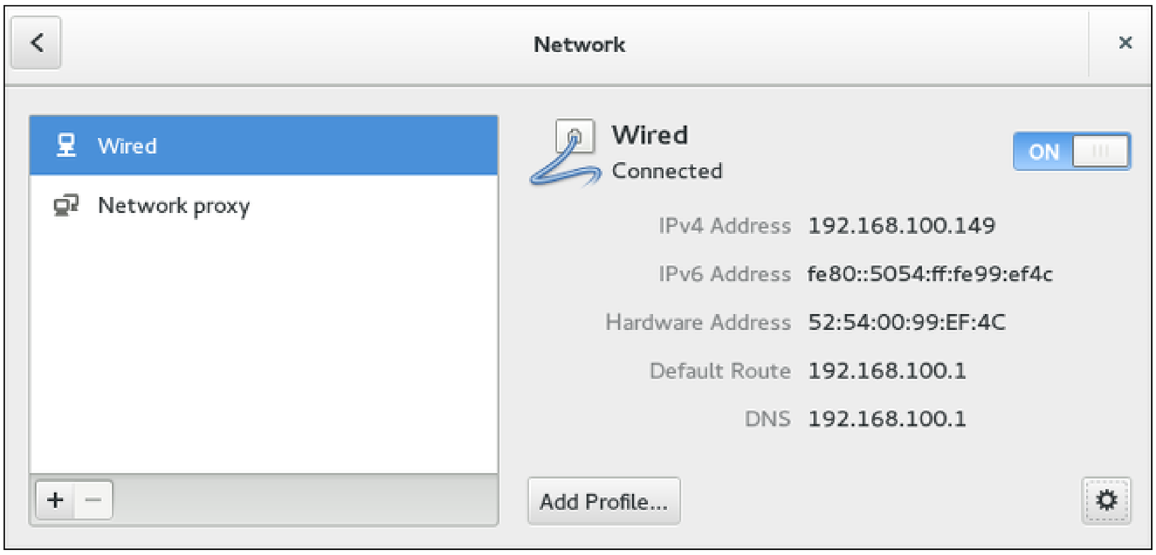
2.2.1. Connecting to a Network Using a GUI

Access the **Network** settings window of the **control-center** application as follows:

- Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears. Proceed to *Section 2.2.2, “Configuring New and Editing Existing Connections”*

2.2.2. Configuring New and Editing Existing Connections

The **Network** settings window shows the connection status, its type and interface, its `IP` address and routing details, and so on.



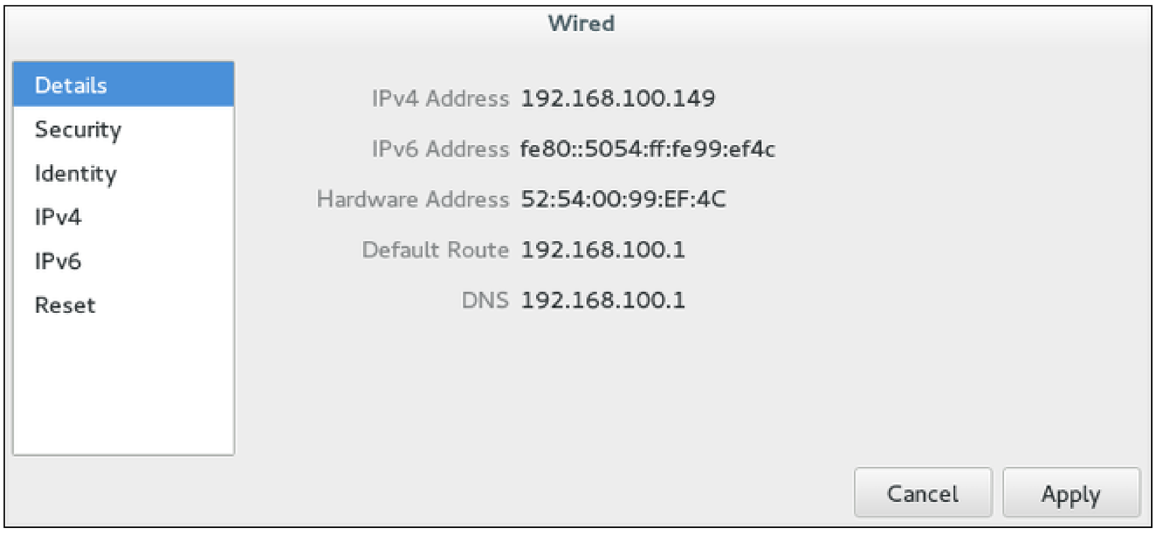
IPv4 Address 192.168.100.149
IPv6 Address fe80::5054:ff:fe99:ef4c
Hardware Address 52:54:00:99:EF:4C
Default Route 192.168.100.1
DNS 192.168.100.1

Figure 2.1. Configure Networks Using the Network Settings Window

The **Network** settings window has a menu on the left-hand side showing the available network devices or interfaces. This includes software interfaces such as for VLANs, bridges, bonds, and teams. On the right-hand side, the *connection profiles* are shown for the selected network device or interface. A profile is a named collection of settings that can be applied to an interface. Below that is a plus and a minus button for adding and deleting new network connections, and on the right a gear wheel icon will appear for editing the connection details of the selected network device or VPN connection. To add a new connection, click the plus symbol to open the **Add Network Connection** window and proceed to *Section 2.2.2.1, “Configuring a New Connection”*.

Editing an Existing Connection

Clicking on the gear wheel icon of an existing connection profile in the **Network** settings window opens the **Network** details window, from where you can perform most network configuration tasks such as `IP` addressing, `DNS`, and routing configuration.



IPv4 Address 192.168.100.149
IPv6 Address fe80::5054:ff:fe99:ef4c
Hardware Address 52:54:00:99:EF:4C
Default Route 192.168.100.1
DNS 192.168.100.1

Figure 2.2. Configure Networks Using the Network Connection Details Window

2.2.2.1. Configuring a New Connection

In the **Network** settings window, click the plus sign below the menu to open the **Add Network Connection** window. This displays a list of connection types that can be added.

Then, to configure:

- VPN connections, click the **VPN** entry and proceed to *Section 2.2.7, “Establishing a VPN Connection”*;
- Bond connections, click the **Bond** entry and proceed to *Section 4.2.1, “Establishing a Bond Connection”*;
- Bridge connections, click the **Bridge** entry and proceed to *Section 6.1.1, “Establishing a Bridge Connection”*;
- VLAN connections, click the **VLAN** entry and proceed to *Section 7.2.1, “Establishing a VLAN Connection”*;or,
- Team connections, click the **Team** entry and proceed to *Section 5.9, “Creating a Network Team Using a GUI”*.

2.2.3. Connecting to a Network Automatically

For any connection type you add or configure, you can choose whether you want **NetworkManager** to try to connect to that network automatically when it is available.

Procedure 2.1. Configuring NetworkManager to Connect to a Network Automatically When Detected

1. Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.

2. Select the network interface from the left-hand-side menu.

3. Click on the gear wheel icon of a connection profile on the right-hand side menu. If you have only one profile associated with the selected interface the gear wheel icon will be in the lower right-hand-side corner. The **Network** details window appears.

4. Select the **Identity** menu entry on the left. The **Network** window changes to the identity view.

5. Select **Connect automatically** to cause **NetworkManager** to auto-connect to the connection whenever **NetworkManager** detects that it is available. Clear the check box if you do not want **NetworkManager** to connect automatically. If the check box is clear, you will have to select that connection manually in the network settings tool to cause it to connect.

2.2.4. System-wide and Private Connection Profiles

NetworkManager stores all *connection profiles*. A profile is a named collection of settings that can be applied to an interface. **NetworkManager** stores these connection profiles for system-wide use (*system connections*), as well as all *user connection* profiles. Access to the connection profiles is controlled by permissions which are stored by **NetworkManager**. See the **`nm-settings(5)`** man page for more information on the **`connection`** settings **`permissions`** property. 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. As an example, the following command in an **`ifcfg`** file will make the connection available only to the users listed:

```
USERS="joe bob alice"
```

This can also be set using graphical user interface tools. In **nm-connection-editor**, there is the corresponding **All users may connect to this network** check box on the **General** tab, and in the GNOME **control-center** Network settings Identity window, there is the **Make available to other users** check box.

NetworkManager's default policy is to allow all users to create and modify system-wide connections. Profiles that should be available at boot time cannot be private because they will not be visible until the user logs in. For example, if user `user` creates a connection profile **`user-em2`** with the **Connect Automatically** check box selected but with the **Make available to other users** not selected, then the connection will not be available at boot time.

To restrict connections and networking, there are two options which can be used alone or in combination:

- Clear the **Make available to other users** check box, which changes the connection to be modifiable and usable only by the user doing the changing.

- Use the **polkit** framework to restrict permissions of general network operations on a per-user basis.

The combination of these two options provides fine-grained security and control over networking. See the **`polkit(8)`** man page for more information on **polkit**.

Note that VPN connections are **always** created as private-per-user, since they are assumed to be more private than a Wi-Fi or Ethernet connection.

Procedure 2.2. Changing a Connection to Be User-specific Instead of System-Wide, or Vice Versa

Depending on the system's policy, you may need root privileges on the system in order to change whether a connection is user-specific or system-wide.

1. Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.

2. Select the network interface from the left-hand-side menu.
3. Click on the gear wheel icon of a connection profile on the right-hand side menu. If you have only one profile associated with the selected interface the gear wheel icon will be in the lower right-hand-side corner. The **Network** details window appears.
4. Select the **Identity** menu entry on the left. The **Network** window changes to the identity view.
5. Select the **Make available to other users** check box to cause **NetworkManager** to make the connection available system-wide. Depending on system policy, you may then be prompted for an administrator or `root` password by the **polkit** application. If so, enter the appropriate password to finalize the changes.

   Conversely, clear the **Make available to other users** check box to make the connection user-specific.

2.2.5. Configuring a Wired (Ethernet) Connection

To configure a wired network connection, press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.

Select the **Wired** network interface from the left-hand-side menu if it is not already highlighted.

The system creates and configures a single wired *connection profile* called **Wired** by default. A profile is a named collection of settings that can be applied to an interface. 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. You can edit the default **Wired** profile by clicking the gear wheel icon. You can create a new wired connection profile by clicking the **Add Profile** button. Connection profiles associated with a selected interface are shown on the right-hand side menu.

When you add a new connection by clicking the **Add Profile** 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** and **Reset** menu entry. In effect, you are always editing a connection profile; the difference only lies in whether that connection previously existed or was just created by **NetworkManager** when you clicked **Add Profile**.

2.2.5.1. Configuring the Connection Name, Auto-Connect Behavior, and Availability Settings

Many settings in the **Editing** dialog are common to all connection types, see the **Identity** view if using the GNOME tool or the **General** tab if using **nm-connection-editor**:

- **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.

[3] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/

- **Firewall Zone** — If required, select a different firewall zone to apply. See the *Red Hat Enterprise Linux 7 Security Guide*[3] for more information on firewall zones.
- **Connect Automatically** — Select this box if you want **NetworkManager** to auto-connect to this connection when it is available. See *Section 2.2.3, "Connecting to a Network Automatically"* for more information.
- **Make available to other users** — Select this box to create a connection available to all users on the system. Changing this setting may require root privileges. See *Section 2.2.4, "System-wide and Private Connection Profiles"* for details.
- **Automatically connect to VPN when using this connection** — Select this box if you want **NetworkManager** to auto-connect to the selected VPN connection when this connection profile is connected. Select the VPN from the drop-down menu.

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your wired connection, click the **Apply** button to save your customized configuration. If the profile was in use while being edited, power cycle the connection to make **NetworkManager** apply the changes. If the profile is OFF, set it to ON. See *Section 2.2.1, "Connecting to a Network Using a GUI"* for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network** window and clicking the gear wheel icon to return to the editing dialog.

Then, to configure:

- port-based Network Access Control (PNAC), click the **802.1X Security** tab and proceed to *Section 2.2.10.1, "Configuring 802.1X Security"*;
- `IPv4` settings for the connection, click the **IPv4 Settings** tab and proceed to *Section 2.2.10.4, "Configuring IPv4 Settings"*; or,
- `IPv6` settings for the connection, click the **IPv6 Settings** tab and proceed to *Section 2.2.10.5, "Configuring IPv6 Settings"*.

2.2.6. Configuring a Wi-Fi Connection

This section explains how to use **NetworkManager** to configure a Wi-Fi (also known as wireless or 802.11*a/b/g/n*) connection to an Access Point.

To configure a mobile broadband (such as 3G) connection, see *Section 2.2.8, "Establishing a Mobile Broadband Connection"*.

Quickly Connecting to an Available Access Point

The easiest way to connect to an available access point is to click on the network connection icon to activate the network connection, locate the *Service Set Identifier* (SSID) of the access point in the list of **Wi-Fi** networks, and click on it. A padlock symbol indicates the access point requires authentication. If the access point is secured, a dialog prompts you for an authentication key or password.

NetworkManager tries to auto-detect the type of security used by the access point. If there are multiple possibilities, **NetworkManager** guesses the security type and presents it in the **Wi-Fi security** drop-down menu. For WPA-PSK security (WPA with a passphrase) no choice is necessary. For WPA Enterprise (802.1X) you have to specifically select the security, because that cannot be auto-detected. If you are unsure, try connecting to each type in turn. Finally, enter the key or passphrase in the **Password** field. Certain password types, such as a 40-bit WEP or 128-bit WPA key, are invalid unless they are of a requisite length. The **Connect** button will remain inactive until you enter a key of the length required for the selected security type. To learn more about wireless security, see *Section 2.2.10.2, “Configuring Wi-Fi Security”*.

If **NetworkManager** connects to the access point successfully, the network connection icon will change into a graphical indicator of the wireless connection's signal strength.

You can also edit the settings for one of these auto-created access point connections just as if you had added it yourself. The **Wi-Fi** page of the **Network** window has a **History** button. Clicking it reveals a list of all the connections you have ever tried to connect to. See *Section 2.2.6.2, “Editing a Connection, or Creating a Completely New One”*

2.2.6.1. Connecting to a Hidden Wi-Fi Network

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. You can still connect to a wireless access point that is hiding its SSID as long as you know its SSID, authentication method, and secrets.

To connect to a hidden wireless network, press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** window appears. Select **Wi-Fi** from the menu and then select **Connect to Hidden Network** to cause a dialog to appear. If you have connected to the hidden network before, use the **Connection** drop-down to select it, and click **Connect**. If you have not, leave the **Connection** drop-down as **New**, enter the SSID of the hidden network, select its **Wi-Fi security** method, enter the correct authentication secrets, and click **Connect**.

For more information on wireless security settings, see *Section 2.2.10.2, “Configuring Wi-Fi Security”*.

2.2.6.2. Editing a Connection, or Creating a Completely New One

You can edit an existing connection that you have tried or succeeded in connecting to in the past by opening the **Wi-Fi** page of the **Network** dialog and selecting the gear wheel icon to the right of the Wi-Fi connection name. If the network is not currently in range, click **History** to display past connections. When you click the gear wheel icon the editing connection dialog appears. The **Details** window shows the connection details.

To configure a new connection whose SSID is in range, first attempt to connect to it by opening the **Network** window, selecting the **Wi-Fi** menu entry, and clicking the connection name (by default, the same as the SSID). If the SSID is not in range, see *Section 2.2.6.1, “Connecting to a Hidden Wi-Fi Network”*. If the SSID is in range, the procedure is as follows:

1. Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.
2. Select the **Wi-Fi** interface from the left-hand-side menu entry.
3. Click the Wi-Fi connection profile on the right-hand side menu you want to connect to. A padlock symbol indicates a key or password is required.
4. If requested, enter the authentication details.

Configuring the SSID, Auto-Connect Behavior, and Availability Settings

To edit a Wi-Fi connection's settings, select **Wi-Fi** in the **Network** page and then select the gear wheel icon to the right of the Wi-Fi connection name. Select **Identity**. The following settings are available:

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

Select the MAC address, also known as a *hardware address*, of the Wi-Fi interface to use.

A single system could have one or more wireless network adapters connected to it. The **MAC address** field therefore 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.

The following settings are common to all connection profiles:

- **Connect automatically** — Select this box if you want **NetworkManager** to auto-connect to this connection when it is available. See *Section 2.2.3, "Connecting to a Network Automatically"* for more information.
- **Make available to other users** — Select this box to create a connection available to all users on the system. Changing this setting may require root privileges. See *Section 2.2.4, "System-wide and Private Connection Profiles"* for details.

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing the wireless connection, click the **Apply** button to save your configuration. Given a correct configuration, you can connect to your modified connection by selecting it from the GNOME Shell notification area menu in the top right-hand corner of the screen. Click in the top right-hand side corner to open the menu. Select **Wi-Fi**. See *Section 2.2.1, "Connecting to a Network Using a GUI"* for details on selecting and connecting to a network.

You can further configure an existing connection by selecting it in the **Network** window and clicking the gear wheel icon to reveal the connection details.

Then, to configure:

- security authentication for the wireless connection, click **Security** and proceed to *Section 2.2.10.2, "Configuring Wi-Fi Security"*;
- `IPv4` settings for the connection, click **IPv4** and proceed to *Section 2.2.10.4, "Configuring IPv4 Settings"*; or,
- `IPv6` settings for the connection, click **IPv6** and proceed to *Section 2.2.10.5, "Configuring IPv6 Settings"*.

2.2.7. Establishing a VPN Connection

Establishing 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. After successfully establishing a VPN connection using a secure tunnel, a VPN router or gateway performs the following actions upon the packets you transmit:

1. it adds an *Authentication Header* for routing and authentication purposes;
2. it encrypts the packet data; and,
3. it encloses the data in packets according to the Encapsulating Security Payload (ESP) protocol, which constitutes the decryption and handling instructions.

The receiving VPN router strips the header information, decrypts the data, and routes it to its intended destination (either a workstation or other node on a network). Using a network-to-network connection, the receiving node on the local network receives the packets already decrypted and ready for processing. The encryption and decryption process in a network-to-network VPN connection is therefore transparent to clients.

Because they employ several layers of authentication and encryption, VPNs are a secure and effective means of connecting multiple remote nodes to act as a unified intranet.

Procedure 2.3. Adding a New VPN Connection

You can configure a new VPN connection by opening the **Network** window and selecting the plus symbol below the menu.

1. Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.
2. Select the plus symbol below the menu. The **Add Network Connection** window appears.
3. Select the **VPN** menu entry. The view now changes to offer configuring a VPN manually, or importing a VPN configuration file.

   The appropriate **NetworkManager** VPN plug-in for the VPN type you want to configure must be installed. (see *Fedora 24 System Administrator's Guide*[4] for more information on how to install new packages in Fedora 24).
4. Click the **Add** button to open the **Choose a VPN Connection Type** assistant.
5. Select the VPN protocol for the gateway you are connecting to from the menu. The VPN protocols available for selection in the menu correspond to the **NetworkManager** VPN plug-ins installed. For example, if the *NetworkManager-openswan-gnome* package is installed then the IPsec based VPN will be selectable from the menu.
6. The **Add Network Connection** window changes to present the settings customized for the type of VPN connection you selected in the previous step.

Procedure 2.4. Editing an Existing VPN Connection

You can configure an existing VPN connection by opening the **Network** window and selecting the name of the connection from the list. Then click the **Edit** button.

1. Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.
2. Select the **VPN** connection you want to edit from the left hand menu.

[4] https://docs.fedoraproject.org/sysadmin-guide

3. Click the **Configure** button.

Configuring the Connection Name, Auto-Connect Behavior, and Availability Settings

Five settings in the **Editing** dialog are common to all connection types, see the **General** tab:

- **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.
- **Automatically connect to this network when it is available** — Select this box if you want **NetworkManager** to auto-connect to this connection when it is available. See *Section 2.2.3, "Connecting to a Network Automatically"* for more information.
- **All users may connect to this network** — Select this box to create a connection available to all users on the system. Changing this setting may require root privileges. See *Section 2.2.4, "System-wide and Private Connection Profiles"* for details.
- **Automatically connect to VPN when using this connection** — Select this box if you want **NetworkManager** to auto-connect to a VPN connection when it is available. Select the VPN from the drop-down menu.
- **Firewall Zone** — Select the Firewall Zone from the dropdown menu.

Configuring the VPN Tab

Gateway
: The name or `IP` address of the remote VPN gateway.

Group name
: The name of a VPN group configured on the remote gateway.

User password
: If required, enter the password used to authenticate with the VPN.

Group password
: If required, enter the password used to authenticate with the VPN.

User name
: If required, enter the user name used to authenticate with the VPN.

Phase1 Algorithms
: If required, enter the algorithms to be used to authenticate and set up an encrypted channel.

Phase2 Algorithms
: If required, enter the algorithms to be used for the IPsec negotiations.

Domain
: If required, enter the Domain Name.

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your new VPN connection, click the **Save** button to save your customized configuration. If the profile was in use while being edited, power cycle the connection to make **NetworkManager** apply the changes. If the profile is OFF, set it to ON. See *Section 2.2.1, “Connecting to a Network Using a GUI”* for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network** window and clicking **Configure** to return to the **Editing** dialog.

Then, to configure:

- `IPv4` settings for the connection, click the **IPv4 Settings** tab and proceed to *Section 2.2.10.4, “Configuring IPv4 Settings”*.

2.2.8. Establishing a Mobile Broadband Connection

You can use **NetworkManager**'s mobile broadband connection abilities to connect to the following *2G* and *3G* services:

- 2G — *GPRS* (*General Packet Radio Service*), *EDGE* (*Enhanced Data Rates for GSM Evolution*), or CDMA (Code Division Multiple Access).
- 3G — *UMTS* (*Universal Mobile Telecommunications System*), *HSPA* (*High Speed Packet Access*), or EVDO (EVolution Data-Only).

Your computer must have a mobile broadband device (modem), which the system has discovered and recognized, in order to create the connection. Such a device may be built into your computer (as is the case on many notebooks and netbooks), or may be provided separately as internal or external hardware. Examples include PC card, USB Modem or Dongle, mobile or cellular telephone capable of acting as a modem.

Procedure 2.5. Adding a New Mobile Broadband Connection

You can configure a mobile broadband connection by opening the **Network Connections** tool and selecting the **Mobile Broadband** tab.

1. Press the **`Super`** key to enter the Activities Overview, type **`nm-connection-editor`** and then press **`Enter`**. The **Network Connections** tool appears.
2. Click the **Add** button. The **Choose a Connection Type** menu opens.
3. Select the **Mobile Broadband** menu entry.
4. Click **Create** to open the **Set up a Mobile Broadband Connection** assistant.
5. Under **Create a connection for this mobile broadband device**, choose the 2G- or 3G-capable device you want to use with the connection. If the drop-down menu is inactive, this indicates that the system was unable to detect a device capable of mobile broadband. In this case, click **Cancel**, ensure that you do have a mobile broadband-capable device attached and recognized by the computer and then retry this procedure. Click the **Continue** button.
6. Select the country where your service provider is located from the list and click the **Continue** button.
7. Select your provider from the list or enter it manually. Click the **Continue** button.
8. Select your payment plan from the drop-down menu and confirm the *Access Point Name* (APN) is correct. Click the **Continue** button.
9. Review and confirm the settings and then click the **Apply** button.

10. Edit the mobile broadband-specific settings by referring to *Section 2.2.8.1, “Configuring the Mobile Broadband Tab”*.

Procedure 2.6. Editing an Existing Mobile Broadband Connection

Follow these steps to edit an existing mobile broadband connection.

1. Press the **`Super`** key to enter the Activities Overview, type **`nm-connection-editor`** and then press **`Enter`**. The **Network Connections** tool appears.

2. Select the **Mobile Broadband** tab.

3. Select the connection you want to edit and click the **Edit** button.

4. Configure the connection name, auto-connect behavior, and availability settings.

   Five settings in the **Editing** dialog are common to all connection types, see the **General** tab:

   - **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.

   - **Automatically connect to this network when it is available** — Select this box if you want **NetworkManager** to auto-connect to this connection when it is available. See *Section 2.2.3, “Connecting to a Network Automatically”* for more information.

   - **All users may connect to this network** — Select this box to create a connection available to all users on the system. Changing this setting may require root privileges. See *Section 2.2.4, “System-wide and Private Connection Profiles”* for details.

   - **Automatically connect to VPN when using this connection** — Select this box if you want **NetworkManager** to auto-connect to a VPN connection when it is available. Select the VPN from the drop-down menu.

   - **Firewall Zone** — Select the Firewall Zone from the drop-down menu.

5. Edit the mobile broadband-specific settings by referring to *Section 2.2.8.1, “Configuring the Mobile Broadband Tab”*.

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your mobile broadband connection, click the **Apply** button to save your customized configuration. If the profile was in use while being edited, power cycle the connection to make **NetworkManager** apply the changes. If the profile is OFF, set it to ON. See *Section 2.2.1, “Connecting to a Network Using a GUI”* for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network Connections** window and clicking **Edit** to return to the **Editing** dialog.

Then, to configure:

- **Point-to-point** settings for the connection, click the **PPP Settings** tab and proceed to *Section 2.2.10.3, “Configuring PPP (Point-to-Point) Settings”*;

- `IPv4` settings for the connection, click the **IPv4 Settings** tab and proceed to *Section 2.2.10.4, “Configuring IPv4 Settings”*; or,

- `IPv6` settings for the connection, click the **IPv6 Settings** tab and proceed to *Section 2.2.10.5, “Configuring IPv6 Settings”*.

2.2.8.1. Configuring the Mobile Broadband Tab

If you have already added a new mobile broadband connection using the assistant (see *Procedure 2.5, “Adding a New Mobile Broadband Connection”* for instructions), you can edit the **Mobile Broadband** tab to disable roaming if home network is not available, assign a network ID, or instruct **NetworkManager** to prefer a certain technology (such as 3G or 2G) when using the connection.

Number
: The number that is dialed to establish a PPP connection with the GSM-based mobile broadband network. This field may be automatically populated during the initial installation of the broadband device. You can usually leave this field blank and enter the **APN** instead.

Username
: Enter the user name used to authenticate with the network. Some providers do not provide a user name, or accept any user name when connecting to the network.

Password
: Enter the password used to authenticate with the network. Some providers do not provide a password, or accept any password.

APN
: Enter the *Access Point Name* (APN) used to establish a connection with the GSM-based network. Entering the correct APN for a connection is important because it often determines:

- how the user is billed for their network usage; and/or
- whether the user has access to the Internet, an intranet, or a subnetwork.

Network ID
: Entering a **Network ID** causes **NetworkManager** to force the device to register only to a specific network. This can be used to ensure the connection does not roam when it is not possible to control roaming directly.

Type
: **Any** — The default value of **Any** leaves the modem to select the fastest network.

3G (UMTS/HSPA) — Force the connection to use only 3G network technologies.

2G (GPRS/EDGE) — Force the connection to use only 2G network technologies.

Prefer 3G (UMTS/HSPA) — First attempt to connect using a 3G technology such as HSPA or UMTS, and fall back to GPRS or EDGE only upon failure.

Prefer 2G (GPRS/EDGE) — First attempt to connect using a 2G technology such as GPRS or EDGE, and fall back to HSPA or UMTS only upon failure.

Allow roaming if home network is not available
: Uncheck this box if you want **NetworkManager** to terminate the connection rather than transition from the home network to a roaming one, thereby avoiding possible roaming charges. If the box is checked, **NetworkManager** will attempt to maintain a good connection by transitioning from the home network to a roaming one, and vice versa.

PIN
: If your device's *SIM* (*Subscriber Identity Module*) is locked with a *PIN* (*Personal Identification Number*), enter the PIN so that **NetworkManager** can unlock the device. **NetworkManager** must unlock the SIM if a PIN is required in order to use the device for any purpose.

CDMA and EVDO have fewer options. They do not have the **`APN`**, **`Network ID`**, or **`Type`** options.

2.2.9. Establishing a DSL Connection

This section is intended for those installations which have a DSL card fitted within a host rather than the external combined DSL modem router combinations typical of private consumer or SOHO installations.

Procedure 2.7. Adding a New DSL Connection

You can configure a new DSL connection by opening the **Network Connections** window, clicking the **Add** button and selecting **DSL** from the **Hardware** section of the new connection list.

1. Press the **`Super`** key to enter the Activities Overview, type **`nm-connection-editor`** and then press **`Enter`**. The **Network Connections** tool appears.
2. Click the **Add** button.
3. The **Choose a Connection Type** list appears.
4. Select **DSL** and press the **Create** button.
5. The **Editing DSL Connection *1*** window appears.

Procedure 2.8. Editing an Existing DSL Connection

You can configure an existing DSL connection by opening the **Network Connections** window and selecting the name of the connection from the list. Then click the **Edit** button.

1. Press the **`Super`** key to enter the Activities Overview, type **`nm-connection-editor`** and then press **`Enter`**. The **Network Connections** tool appears.
2. Select the connection you want to edit and click the **Edit** button.

Configuring the Connection Name, Auto-Connect Behavior, and Availability Settings

Five settings in the **Editing** dialog are common to all connection types, see the **General** tab:

- **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.
- **Automatically connect to this network when it is available** — Select this box if you want **NetworkManager** to auto-connect to this connection when it is available. See *Section 2.2.3, “Connecting to a Network Automatically”* for more information.
- **All users may connect to this network** — Select this box to create a connection available to all users on the system. Changing this setting may require root privileges. See *Section 2.2.4, “System-wide and Private Connection Profiles”* for details.
- **Automatically connect to VPN when using this connection** — Select this box if you want **NetworkManager** to auto-connect to a VPN connection when it is available. Select the VPN from the drop-down menu.
- **Firewall Zone** — Select the Firewall Zone from the drop-down menu.

Configuring the DSL Tab

Username
: Enter the user name used to authenticate with the service provider.

Service
: Leave blank unless otherwise directed by your service provider.

Password
: Enter the password supplied by the service provider.

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your DSL connection, click the **Apply** button to save your customized configuration. If the profile was in use while being edited, power cycle the connection to make **NetworkManager** apply the changes. If the profile is OFF, set it to ON. See *Section 2.2.1, “Connecting to a Network Using a GUI”* for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network Connections** window and clicking **Edit** to return to the **Editing** dialog.

Then, to configure:

- **The MAC address and MTU** settings, click the **Wired** tab and proceed to *Section 2.2.5.1, “Configuring the Connection Name, Auto-Connect Behavior, and Availability Settings”*;
- **Point-to-point** settings for the connection, click the **PPP Settings** tab and proceed to *Section 2.2.10.3, “Configuring PPP (Point-to-Point) Settings”*;
- `IPv4` settings for the connection, click the **IPv4 Settings** tab and proceed to *Section 2.2.10.4, “Configuring IPv4 Settings”*.

2.2.10. Configuring Connection Settings

2.2.10.1. Configuring 802.1X Security

802.1X security is the name of the IEEE standard for *port-based Network Access Control* (PNAC). It is also called *WPA Enterprise*. Simply put, 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.

802.1X security is most often associated with securing wireless networks (WLANs), but can also be used to prevent intruders with physical access to the network (LAN) from gaining entry. In the past, `DHCP` servers were configured not to lease `IP` addresses to unauthorized users, but for various reasons this practice is both impractical and insecure, and thus is no longer recommended. Instead, 802.1X security is used to ensure a logically-secure network through port-based authentication.

802.1X provides a framework for WLAN and LAN access control and serves as an envelope for carrying one of the Extensible Authentication Protocol (EAP) types. An EAP type is a protocol that defines how security is achieved on the network.

You can configure 802.1X security for a wired or wireless connection type by opening the **Network** window (see *Section 2.2.1, “Connecting to a Network Using a GUI”*) and following the applicable procedure below. Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears. Proceed to *Procedure 2.9, “For a Wired Connection”* or *Procedure 2.10, “For a Wireless Connection”*:

Procedure 2.9. For a Wired Connection

1. Select a **Wired** network interface from the left-hand-side menu.

2. Either click on **Add Profile** to add a new network connection profile for which you want to configure 802.1X security, or select an existing connection profile and click the gear wheel icon.

3. Then select **Security** and set the symbolic power button to **ON** to enable settings configuration.

4. Proceed to *Section 2.2.10.1.1, “Configuring TLS (Transport Layer Security) Settings”*

Procedure 2.10. For a Wireless Connection

1. Select a **Wireless** network interface from the left-hand-side menu. If necessary, set the symbolic power button to **ON** and check that your hardware switch is on.

2. Either select the connection name of a new connection, or click the gear wheel icon of an existing connection profile, for which you want to configure 802.1X security. In the case of a new connection, complete any authentication steps to complete the connection and then click the gear wheel icon.

3. Select **Security**.

4. From the drop-down menu select one of the following security methods: **LEAP**, **Dynamic WEP (802.1X)**, or **WPA & WPA2 Enterprise**.

5. Refer to *Section 2.2.10.1.1, “Configuring TLS (Transport Layer Security) Settings”* for descriptions of which *extensible authentication protocol* (EAP) types correspond to your selection in the **Security** drop-down menu.

2.2.10.1.1. Configuring TLS (Transport Layer Security) Settings

With Transport Layer Security, the client and server mutually authenticate using the TLS protocol. The server demonstrates that it holds a digital certificate, the client proves its own identity using its client-side certificate, and key information is exchanged. Once authentication is complete, the TLS tunnel is no longer used. Instead, the client and server use the exchanged keys to encrypt data using AES, TKIP or WEP.

The fact that certificates must be distributed to all clients who want to authenticate means that the EAP-TLS authentication method is very strong, but also more complicated to set up. 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.

Selecting an Authentication Method

Select from one of following authentication methods:

- Select **TLS** for *Transport Layer Security* and proceed to *Section 2.2.10.1.2, “Configuring TLS Settings”*;

- Select **FAST** for *Flexible Authentication via Secure Tunneling* and proceed to *Section 2.2.10.1.4, “Configuring Tunneled TLS Settings”*;

- Select **Tunneled TLS** for *Tunneled Transport Layer Security*, otherwise known as TTLS, or EAP-TTLS and proceed to *Section 2.2.10.1.4, “Configuring Tunneled TLS Settings”*;

- Select **Protected EAP (PEAP)** for *Protected Extensible Authentication Protocol* and proceed to *Section 2.2.10.1.5, “Configuring Protected EAP (PEAP) Settings”*.

2.2.10.1.2. Configuring TLS Settings

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.

2.2.10.1.3. Configuring FAST Settings

Anonymous Identity
: Provide the identity of this server.

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.

2.2.10.1.4. Configuring Tunneled TLS Settings

Anonymous identity
: This value is used as the unencrypted identity.

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

Inner authentication

PAP — Password Authentication Protocol.

MSCHAP — Challenge Handshake Authentication Protocol.

MSCHAPv2 — Microsoft Challenge Handshake Authentication Protocol version 2.

CHAP — Challenge Handshake Authentication Protocol.

Username

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

Password

Enter the password to be used in the authentication process.

2.2.10.1.5. Configuring Protected EAP (PEAP) Settings

Anonymous Identity

This value is used as the unencrypted identity.

CA certificate

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

PEAP version

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

Inner authentication

MSCHAPv2 — Microsoft Challenge Handshake Authentication Protocol version 2.

MD5 — Message Digest 5, a cryptographic hash function.

GTC — Generic Token Card.

Username

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

Password

Enter the password to be used in the authentication process.

2.2.10.2. Configuring Wi-Fi Security

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 will be used to derive a WEP key.

LEAP — Lightweight Extensible Authentication Protocol, from Cisco Systems.

Dynamic WEP (802.1X) — WEP keys are changed dynamically. Use with *Section 2.2.10.1.1, “Configuring TLS (Transport Layer Security) Settings”*

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. Use with *Section 2.2.10.1.1, “Configuring TLS (Transport Layer Security) Settings”*

Password
: Enter the password to be used in the authentication process.

2.2.10.3. Configuring PPP (Point-to-Point) Settings

Configure Methods

Use point-to-point encryption (MPPE)
: Microsoft Point-To-Point Encryption protocol (*RFC 3078*[5]).

Allow BSD data compression
: PPP BSD Compression Protocol (*RFC 1977*[6]).

Allow Deflate data compression
: PPP Deflate Protocol (*RFC 1979*[7]).

Use TCP header compression
: Compressing TCP/IP Headers for Low-Speed Serial Links (*RFC 1144*[8]).

Send PPP echo packets
: LCP Echo-Request and Echo-Reply Codes for loopback tests (*RFC 1661*[9]).

2.2.10.4. Configuring IPv4 Settings

The **IPv4 Settings** tab allows you to configure the method used to connect to a network, to enter `IP` address, route, and `DNS` information as required. The **IPv4 Settings** tab is available when you create and modify one of the following connection types: wired, wireless, mobile broadband, VPN or DSL. If you need to configure `IPv6` addresses, see *Section 2.2.10.5, “Configuring IPv6 Settings”*. If you need to configure static routes, click the **Routes** button and proceed to *Section 2.2.10.6, “Configuring Routes”*.

If you are using `DHCP` to obtain a dynamic `IP` address from a `DHCP` server, you can simply set **Method** to **Automatic (DHCP)**.

Setting the Method

Available IPv4 Methods by Connection Type

When you click the **Method** drop-down menu, depending on the type of connection you are configuring, you are able to select one of the following `IPv4` connection methods. All of the methods are listed here according to which connection type, or types, they are associated with:

Method
: **Automatic (DHCP)** — Choose this option if the network you are connecting to uses a `DHCP` server to assign `IP` addresses. You do not need to fill in the **DHCP client ID** field.

[5] http://www.rfc-editor.org/info/rfc3078
[6] http://www.rfc-editor.org/info/rfc1977
[7] http://www.rfc-editor.org/info/rfc1979
[8] http://www.rfc-editor.org/info/rfc1144
[9] http://www.rfc-editor.org/info/rfc1661

Automatic (DHCP) addresses only — Choose this option if the network you are connecting to uses a `DHCP` server to assign `IP` addresses but you want to assign `DNS` servers manually.

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*[10] with prefix `169.254/16`.

Shared to other computers — Choose this option if the interface you are configuring is for sharing an Internet or WAN connection. The interface is assigned an address in the `10.42.x.1/24` range, a `DHCP` server and `DNS` server are started, and the interface is connected to the default network connection on the system with *network address translation* (NAT).

Disabled — `IPv4` is disabled for this connection.

Wired, Wireless and DSL Connection Methods

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

Mobile Broadband Connection Methods

Automatic (PPP) — Choose this option if the network you are connecting to assigns your `IP` address and `DNS` servers automatically.

Automatic (PPP) addresses only — Choose this option if the network you are connecting to assigns your `IP` address automatically, but you want to manually specify `DNS` servers.

VPN Connection Methods

Automatic (VPN) — Choose this option if the network you are connecting to assigns your `IP` address and `DNS` servers automatically.

Automatic (VPN) addresses only — Choose this option if the network you are connecting to assigns your `IP` address automatically, but you want to manually specify `DNS` servers.

DSL Connection Methods

Automatic (PPPoE) — Choose this option if the network you are connecting to assigns your `IP` address and `DNS` servers automatically.

Automatic (PPPoE) addresses only — Choose this option if the network you are connecting to assigns your `IP` address automatically, but you want to manually specify `DNS` servers.

For information on configuring static routes for the network connection, go to *Section 2.2.10.6, "Configuring Routes"*.

2.2.10.5. Configuring IPv6 Settings

Method

Ignore — Choose this option if you want to ignore `IPv6` settings for this connection.

Automatic — Choose this option to use *SLAAC* to create an automatic, stateless configuration based on the hardware address and *router advertisements* (RA).

Automatic, addresses only — Choose this option if the network you are connecting to uses *router advertisements* (RA) to create an automatic, stateless configuration, but you want to assign `DNS` servers manually.

[10] http://www.rfc-editor.org/info/rfc3927

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

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

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*[11] with prefix `FE80::0`.

Addresses

DNS servers — Enter a comma separated list of `DNS` servers.

Search domains — Enter a comma separated list of domain controllers.

For information on configuring static routes for the network connection, go to *Section 2.2.10.6, "Configuring Routes"*.

2.2.10.6. Configuring Routes

A host's routing table will be automatically populated with routes to directly connected networks. The routes are learned by examining the network interfaces when they are "up". This section describes entering static routes to networks or hosts which can be reached by traversing an intermediate network or connection, such as a VPN tunnel or leased line. In order to reach a remote network or host, the system is given the address of a gateway to which traffic should be sent.

When a host's interface is configured by `DHCP`, an address of a gateway that leads to an upstream network or the Internet is usually assigned. This gateway is usually referred to as the default gateway as it is the gateway to use if no better route is known to the system (and present in the routing table). Network administrators often use the first or last host `IP` address in the network as the gateway address; for example, `192.168.10.1` or `192.168.10.254`. Not to be confused by the address which represents the network itself; in this example, `192.168.10.0`, or the subnet's broadcast address; in this example `192.168.10.255`.

Configuring Static Routes

To set a static route, open the **IPv4** or **IPv6** settings window for the connection you want to configure. See *Section 2.2.1, "Connecting to a Network Using a GUI"* for instructions on how to do that.

Routes

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.

Automatic

When Automatic is **ON**, routes from `RA` or `DHCP` are used, but you can also add additional static routes. When **OFF**, only static routes you define are used.

[11] http://www.rfc-editor.org/info/rfc4862

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.

2.3. Using the Command Line Interface (CLI)

2.3.1. Configuring a Network Interface Using ifcfg Files

Interface configuration 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.

Static Network Settings

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

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

Optionally specify the hardware or MAC address using the **`HWADDR`** directive. Note that this may influence the device naming procedure as explained in *Chapter 8, Consistent Network Device Naming*. You do not need to specify the network or broadcast address as this is calculated automatically by **ipcalc**. To enable a normal user to set the interface up or down, add **`USERCTL=yes`**.

Dynamic Network Settings

To configure an interface with dynamic network settings using **`ifcfg`** files, for an interface with name em1, create a file with name **`ifcfg-em1`** in the **`/etc/sysconfig/network-scripts/`** directory as follows:

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

Optionally specify the hardware or MAC address using the **`HWADDR`** directive. Note that this may influence the device naming procedure as explained in *Chapter 8, Consistent Network Device Naming*. You do not need to specify the network or broadcast address as this is calculated automatically by **ipcalc**.

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
```

To configure an interface to ignore routes sent by a DHCP server, add the following line to the **ifcfg** file.

```
PEERDNS=no
```

This will prevent network service from updating **/etc/resolv.conf** with the DNS servers received from a DHCP server.

To configure an interface to use particular DNS servers, set **PEERDNS=no** as described above and add lines as follows to the **ifcfg** file:

```
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 DNS servers specified.

NetworkManager will by default call the DHCP client, **dhclient**, when a profile has been set to obtain addresses automatically, or when an interface configuration file has BOOTPROTO set to **dhcp**. Where DHCP is required, an instance of `dhclient` is started for every Internet protocol, IPv4 and IPv6, on an interface. Where **NetworkManager** is not running, or not managing an interface, then the legacy network service will call instances of `dhclient` as required.

2.3.2. Configuring a Network Interface Using ip Commands

The **ip** utility can be used to assign IP addresses to an interface. The command takes the following form:

```
ip addr [ add | del ] address dev ifname
```

Assigning a Static Address Using ip Commands

To assign an IP address to an interface, issue a command as `root` as follows:

```
~]# ip address add 10.0.0.3/24 dev eth0
The address assignment of a specific device can be viewed as follows:
~]# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether f0:de:f1:7b:6e:5f brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.3/24 brd 10.0.0.255 scope global global eth0
       valid_lft 58682sec preferred_lft 58682sec
    inet6 fe80::f2de:f1ff:fe7b:6e5f/64 scope link
       valid_lft forever preferred_lft forever
```

Further examples and command options can be found in the **ip-address(8)** manual page.

Configuring Multiple Addresses Using ip Commands

As the **ip** utility supports assigning multiple addresses to the same interface it is no longer necessary to use the alias interface method of binding multiple addresses to the same interface. The **ip** command to assign an address can be repeated multiple times in order to assign multiple address. For example:

```
~]# ip address add 192.168.2.223/24 dev eth1
~]# ip address add 192.168.4.223/24 dev eth1
~]# ip addr
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:fb:77:9e brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.223/24 scope global eth1
    inet 192.168.4.223/24 scope global eth1
```

The commands for the **ip** utility are documented in the **`ip(8)`** manual page.

Note

Note that **ip** commands given on the command line will not persist after a system restart.

2.3.3. Static Routes and the Default Gateway

Static routes are for traffic that must not, or should not, go through the default gateway. Routing is often handled by devices on the network dedicated to routing (although any device can be configured to perform routing). Therefore, it is often not necessary to configure static routes on Fedora servers or clients. Exceptions include traffic that must pass through an encrypted VPN tunnel or traffic that should take a specific route for reasons of cost or security. The default gateway is for any and all traffic which is not destined for the local network and for which no preferred route is specified in the routing table. The default gateway is traditionally a dedicated network router.

Configuring Static Routes Using the Command Line

If static routes are required, they can be added to the routing table by means of the **`ip route add`** command and removed using the **`ip route del`** command. The more frequently used **`ip route`** commands take the following form:

```
ip route [ add | del | change | append | replace ] destination-address
```

See the **`ip-route(8)`** man page for more details on the options and formats.

Use the **`ip route`** command without options to display the `IP` routing table. For example:

```
~]$ ip route
default via 192.168.122.1 dev eth1  proto static  metric 1024
192.168.122.0/24 dev eth1  proto kernel  scope link  src 192.168.122.107
192.168.122.0/24 dev eth0  proto kernel  scope link  src 192.168.122.126
```

To add a static route to a host address, in other words to a single `IP` address, issue a command as `root`:

```
ip route add 192.0.2.1 via 10.0.0.1 [dev ifname]
```

Where *`192.0.2.1`* is the `IP` address of the host in dotted decimal notation, *`10.0.0.1`* is the next hop address and *`ifname`* is the exit interface leading to the next hop.

To add a static route to a network, in other words to an `IP` address representing a range of `IP` addresses, issue the following command as `root`:

```
ip route add 192.0.2.0/24 via 10.0.0.1 [dev ifname]
```

where *`192.0.2.0`* is the `IP` address of the destination network in dotted decimal notation and *`/24`* is the network prefix. The network prefix is the number of enabled bits in the subnet mask. This format of network address slash network prefix length is sometimes referred to as *classless inter-domain routing* (CIDR) notation.

Static route configuration can be stored per-interface in a **`/etc/sysconfig/network-scripts/route-`*`interface`*** file. For example, static routes for the eth0 interface would be stored in the **`/etc/sysconfig/network-scripts/route-eth0`** file. The **`route-`*`interface`*** file has two formats: **ip** command arguments and network/netmask directives. These are described below.

See the **`ip-route(8)`** man page for more information on the **`ip route`** command.

Configuring The Default Gateway

The default gateway is determined by the network scripts which parse the **`/etc/sysconfig/network`** file first and then the network interface **`ifcfg`** files for interfaces that are "up". The **`ifcfg`** files are parsed in numerically ascending order, and the last GATEWAY directive to be read is used to compose a default route in the routing table.

The default route can thus be indicated by means of the GATEWAY directive, either globally or in interface-specific configuration files. However, in Fedora the use of the global **`/etc/sysconfig/network`** file is deprecated, and specifying the gateway should now only be done in per-interface configuration files.

In dynamic network environments, where mobile hosts are managed by **NetworkManager**, gateway information is likely to be interface specific and is best left to be assigned by `DHCP`. In special cases where it is necessary to influence **NetworkManager**'s selection of the exit interface to be used to reach a gateway, make use of the **`DEFROUTE=no`** command in the **`ifcfg`** files for those interfaces which do not lead to the default gateway.

2.3.4. Configuring Static Routes in ifcfg files

Static routes set using **ip** commands on the command line will be lost if the system is shutdown or restarted. To configure static routes to be persistent after a system restart, they must be placed in per-interface configuration files in the **`/etc/sysconfig/network-scripts/`** directory. The file name should be of the format **`route-`*`ifname`***. There are two types of commands to use in the configuration files; **ip** commands as explained in *Section 2.3.4.1, "Static Routes Using the IP Command Arguments Format"* and the *Network/Netmask* format as explained in *Section 2.3.4.2, "Network/Netmask Directives Format"*.

2.3.4.1. Static Routes Using the IP Command Arguments Format

If required in a per-interface configuration file, for example **`/etc/sysconfig/network-scripts/route-eth0`**, define a route to a default gateway on the first line. This is only required if the gateway is not set via `DHCP` and is not set globally in the **`/etc/sysconfig/network`** file:

```
default via 192.168.1.1 dev interface
```

where *`192.168.1.1`* is the `IP` address of the default gateway. The *`interface`* is the interface that is connected to, or can reach, the default gateway. The **dev** option can be omitted, it is optional. Note that this setting takes precedence over a setting in the **`/etc/sysconfig/network`** file.

If a route to a remote network is required, a static route can be specified as follows. Each line is parsed as an individual route:

```
10.10.10.0/24 via 192.168.1.1 [dev interface]
```

where *10.10.10.0/24* is the network address and prefix length of the remote or destination network. The address *192.168.1.1* is the IP address leading to the remote network. It is preferably the *next hop address* but the address of the exit interface will work. The “next hop” means the remote end of a link, for example a gateway or router. The **dev** option can be used to specify the exit interface *interface* but it is not required. Add as many static routes as required.

The following is an example of a **route-*interface*** file using the **ip** command arguments format. The default gateway is 192.168.0.1, interface eth0 and a leased line or WAN connection is available at 192.168.0.10. The two static routes are for reaching the 10.10.10.0/24 network and the 172.16.1.10/32 host:

```
default via 192.168.0.1 dev eth0
10.10.10.0/24 via 192.168.0.10 dev eth0
172.16.1.10/32 via 192.168.0.10 dev eth0
```

In the above example, packets going to the local 192.168.0.0/24 network will be directed out the interface attached to that network. Packets going to the 10.10.10.0/24 network and 172.16.1.10/32 host will be directed to 192.168.0.10. Packets to unknown, remote, networks will use the default gateway therefore static routes should only be configured for remote networks or hosts if the default route is not suitable. Remote in this context means any networks or hosts that are not directly attached to the system.

Specifying an exit interface is optional. It can be useful if you want to force traffic out of a specific interface. For example, in the case of a VPN, you can force traffic to a remote network to pass through a tun0 interface even when the interface is in a different subnet to the destination network.

Duplicate default gateways

If the default gateway is already assigned by DHCP and if the same gateway with the same metric is specified in a configuration file, an error during start-up, or when bringing up an interface, will occur. The follow error message may be shown: "RTNETLINK answers: File exists". This error may be ignored.

2.3.4.2. Network/Netmask Directives Format

You can also use the network/netmask directives format for **route-*interface*** files. The following is a template for the network/netmask format, with instructions following afterwards:

```
ADDRESS0=10.10.10.0 NETMASK0=255.255.255.0 GATEWAY0=192.168.1.1
```

- **ADDRESS0=*10.10.10.0*** is the network address of the remote network or host to be reached.
- **NETMASK0=*255.255.255.0*** is the netmask for the network address defined with **ADDRESS0=*10.10.10.0***.
- **GATEWAY0=*192.168.1.1*** is the default gateway, or an IP address that can be used to reach **ADDRESS0=*10.10.10.0***

The following is an example of a **route-*interface*** file using the network/netmask directives format. The default gateway is 192.168.0.1 but a leased line or WAN connection is available at

`192.168.0.10`. The two static routes are for reaching the `10.10.10.0/24` and `172.16.1.0/24` networks:

```
ADDRESS0=10.10.10.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.0.10
ADDRESS1=172.16.1.10
NETMASK1=255.255.255.0
GATEWAY1=192.168.0.10
```

Subsequent static routes must be numbered sequentially, and must not skip any values. For example, **`ADDRESS0`**, **`ADDRESS1`**, **`ADDRESS2`**, and so on.

2.3.5. Configuring IPv6 Tokenized Interface Identifiers

In a network, servers are generally given static addresses and these are usually configured manually to avoid relying on a `DHCP` server which may fail or run out of addresses. The `IPv6` protocol introduced *Stateless Address Autoconfiguration* (SLAAC) which enables clients to assign themselves an address without relying on a `DHCPv6` server. SLAAC derives the `IPv6` address based on the interface hardware, therefore it should not be used for servers in case the hardware is changed and the associated SLAAC generated address changes with it. In an `IPv6` environment, if the network prefix is changed, or the system is moved to a new location, any manually configured static addresses would have to be edited due to the changed prefix.

To address these problems, the IETF draft *Tokenised IPv6 Identifiers*[12] has been implemented in the kernel together with corresponding additions to the **`ip`** utility. This enables the lower 64 bit interface identifier part of the `IPv6` address to be based on a token, supplied by the administrator, leaving the network prefix, the higher 64 bits, to be obtained from *router advertisements* (RA). This means that if the network interface hardware is changed, the lower 64 bits of the address will not change, and if the system is moved to another network, the network prefix will be obtained from router advertisements automatically, thus no manual editing is required.

To configure an interface to use a tokenized `IPv6` identifier, issue a command in the following format as `root` user:

```
~]# ip token set ::1a:2b:3c:4d/64 dev eth4
```

Where **`::1a:2b:3c:4d/64`** is the token to be used. This setting is not persistent. To make it persistent, add the command to an init script. See *Section 1.7, “Network Configuration Using sysconfig Files”* for information on network scripts.

Using a memorable token is possible, but is limited to the range of valid hexadecimal digits. For example, for a `DNS` server, which traditionally uses port **`53`**, a token of **`::53/64`** could be used.

To view all the configured `IPv6` tokens, issue the following command:

```
~]$ ip token
     token :: dev eth0
     token :: dev eth1
     token :: dev eth2
     token :: dev eth3
     token ::1a:2b:3c:4d dev eth4
```

[12] https://tools.ietf.org/id/draft-chown-6man-tokenised-ipv6-identifiers-02.txt

To view the configured `IPv6` token for a specific interface, issue the following command:

```
~]$ ip token get dev eth4
     token ::1a:2b:3c:4d dev eth4
```

Note that adding a token to an interface will replace a previously allocated token, and in turn invalidate the address derived from it. Supplying a new token causes a new address to be generated and applied, but this process will leave any other addresses unchanged. In other words, a new tokenized identifier only replaces a previously existing tokenized identifier, not any other `IP` address.

Note

Take care not to add the same token to more than one system or interface. The duplicate address detection (DAD) mechanism will remove the duplicate address from the interface but it cannot change the configuration. Once a token is set, it cannot be cleared or reset, except by rebooting the machine.

2.4. Using the NetworkManager Command Line Tool, nmcli

The command-line tool **nmcli** can be used by both users and scripts for controlling **NetworkManager**. The basic format of a command is as follows:

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

where OBJECT can be one of **`general`**, **`networking`**, **`radio`**, **`connection`**, or **`device`**. The most used options are: **`-t, --terse`** for use in scripts, the **`-p, --pretty`** option for users, and the **`-h, --help`** option. Command completion has been implemented for **nmcli**, so remember to press **Tab** whenever you are unsure of the command options available. See the **`nmcli(1)`** man page for a complete list of the options and commands.

The **nmcli** tool has some built-in context-sensitive help. For example, issue the following two commands and notice the difference:

```
~]$ nmcli help
Usage: nmcli [OPTIONS] OBJECT { COMMAND | help }

OPTIONS
  -t[erse]                                    terse output
  -p[retty]                                   pretty output
  -m[ode] tabular|multiline                   output mode
  -f[ields] <field1,field2,...>|all|common    specify fields to output
  -e[scape] yes|no                            escape columns separators in values
  -n[ocheck]                                  don't check nmcli and NetworkManager versions
  -a[sk]                                      ask for missing parameters
  -w[ait] <seconds>                           set timeout waiting for finishing operations
  -v[ersion]                                  show program version
  -h[elp]                                     print this help

OBJECT
  g[eneral]       NetworkManager's general status and operations
  n[etworking]    overall networking control
  r[adio]         NetworkManager radio switches
  c[onnection]    NetworkManager's connections
  d[evice]        devices managed by NetworkManager
```

```
~]$ nmcli general help
Usage: nmcli general { COMMAND | help }

  COMMAND := { status | hostname | permissions | logging }

  status

  hostname [<hostname>]

  permissions

  logging [level <log level>] [domains <log domains>]
```

In the second example above the help is related to the object **`general`**.

The **`nmcli-examples(5)`** man page has many useful examples. A brief selection is shown here:

To show the overall status of **NetworkManager**:

```
nmcli general status
```

To control **NetworkManager** logging:

```
nmcli general logging
```

To show all connections:

```
nmcli connection show
```

To show only currently active connections, add the **`-a, --active`** option as follows:

```
nmcli connection show --active
```

To show devices recognized by **NetworkManager** and their state:

```
nmcli device status
```

Commands can be shortened and some options omitted. For example the command:

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

Can be reduced to the following command:

```
nmcli con mod MyCafe 802-11-wireless.mtu 1350
```

The **`id`** option can be omitted because the connection ID (name) is unambiguous for **nmcli** in this case. As you become familiar with the commands, further abbreviations can be made. For example:

```
nmcli connection add type ethernet
```

can be reduced to:

```
nmcli c a type eth
```

Note

Remember to use tab completion when in doubt.

Starting and Stopping an Interface Using nmcli

The **nmcli** tool can be used to start and stop any network interface, including masters. For example:

```
nmcli con up id bond0
nmcli con up id port0
nmcli dev disconnect bond0
nmcli dev disconnect eth0
```

Note

It is recommended to use **`nmcli dev disconnect`** ***`iface-name`*** rather than **`nmcli con down id`** ***`id-string`*** because disconnection places the interface into a “manual” mode, in which no automatic connection will be started until the user tells **NetworkManager** to start a connection or until an external event like a carrier change, hibernate, or sleep, occurs.

The nmcli Interactive Connection Editor

The **nmcli** tool has an interactive connection editor. To use it, enter the following command:

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

You will be prompted to enter a valid connection type from the list displayed. After entering a connection type you will be placed at the **nmcli** prompt. If you are familiar with the connection types you can add a valid connection **`type`** option to the **`nmcli con edit`** command and be taken straight to the **nmcli** prompt. The format is as follows for editing an existing connection profile:

```
nmcli con edit [id | uuid | path] ID
```

For adding and editing a new connection profile, the following format applies:

```
nmcli con edit [type new-connection-type] [con-name new-connection-name]
```

Type **`help`** at the **nmcli** prompt to see a list of valid commands. Use the **`describe`** command to get a description of settings and their properties. The format is as follows:

```
describe setting.property
```

For example:

```
nmcli> describe team.config
```

2.4.1. Understanding the nmcli Options

Many of the **nmcli** commands are self-explanatory, however a few command options are worth a moments study:

`type` — The connection type.

Allowed values are: **`adsl`**, **`bond`**, **`bond-slave`**, **`bridge`**, **`bridge-slave`**, **`bluetooth`**, **`cdma`**, **`ethernet`**, **`generic`**, **`gsm`**, **`infiniband`**, **`olpc-mesh`**, **`pppoe`**, **`team`**, **`team-slave`**, **`vlan`**, **`vpn`**, **`wifi`**, **`wimax`**.

Each connection type has type-specific command options. Press **Tab** to see a list of them or see the **`TYPE_SPECIFIC_OPTIONS`** list in the **`nmcli(1)`** man page. The **`type`** option is applicable after the following: **`nmcli connection add`** and **`nmcli connection edit`**.

`con-name` — The name assigned to a connection profile.

If you do not specify a connection name, one will be generated as follows:

```
type-ifname[-number]
```

The connection name is the name of a *connection profile* and should not be confused with the interface name that denotes a device (wlan0, eth0, em1, and so on). Users can however 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.

`id` — An identification string assigned by the user to a connection profile.

The ID can be used in **`nmcli connection`** commands to identify a connection. The NAME field in the output always denotes the connection ID (name). 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.

2.4.2. Connecting to a Network Using nmcli

To list the currently available network connections, issue a command as follows:

```
~]$ nmcli con show
NAME  UUID                                  TYPE           DEVICE
eth0  96a5deb0-5eb0-41e1-a7ed-38fea413f9c8  802-3-ethernet eth0
MyWiFi             91451385-4eb8-4080-8b82  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 example above **`eth0`** is the connection ID given by the user to the profile applied to the interface eth0. In the second line 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 dev status
DEVICE    TYPE       STATE         CONNECTION
wlan0     wifi          connected     my-ssid
eth0      ethernet   unavailable   --
```

```
lo        loopback   unmanaged     --
```

Adding a Dynamic Ethernet Connection

To add an Ethernet configuration profile with dynamic `IP` configuration, allowing `DHCP` to assign the network configuration, a command in the following format can be used:

```
nmcli connection add type ethernet con-name connection-name ifname interface-name
```

For example, to create a dynamic connection profile named *my-office*, issue a command as follows:

```
~]$ nmcli con add type ethernet con-name my-office ifname eth0
Connection 'my-office' (0a053110-5361-412c-a4fb-6ff20877e9e4) successfully added.
```

NetworkManager will set its internal parameter **connection.autoconnect** to **yes**. **NetworkManager** will also write out settings to **/etc/sysconfig/network-scripts/ifcfg-my-office** where the ONBOOT directive will be set to **yes**.

Note that manual changes to the ifcfg file will not be noticed by **NetworkManager** until the interface is next brought up. See *Section 1.7, "Network Configuration Using sysconfig Files"* for more information on using configuration files.

To bring up the Ethernet connection, issue a command as follows:

```
~]$ nmcli con up my-office
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/
ActiveConnection/1)
```

Review the status of the devices and connections:

```
~]$ nmcli device status
DEVICE  TYPE      STATE         CONNECTION
eth0    ethernet  connected     my-office
lo      loopback  unmanaged     --
```

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

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

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

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

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

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

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

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

Example 2.1. Configuring a Dynamic Ethernet Connection Using the Interactive Editor

To configure a dynamic Ethernet connection using the interactive editor, issue commands as follows:

```
~]$ nmcli con edit type ethernet con-name eth0

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

Adding a new '802-3-ethernet' connection

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

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

=== [method] ===
[NM property description]
IPv4 configuration method.  If 'auto' is specified then the appropriate automatic method
 (DHCP, PPP, etc) is used for the interface and most other properties can be left unset.
  If 'link-local' is specified, then a link-local address in the 169.254/16 range will be
 assigned to the interface.  If 'manual' is specified, static IP addressing is used and at
 least one IP address must be given in the 'addresses' property.  If 'shared' is specified
 (indicating that this connection will provide network access to other computers) then
 the interface is assigned an address in the 10.42.x.1/24 range and a DHCP and forwarding
 DNS server are started, and the interface is NAT-ed to the current default network
 connection.  'disabled' means IPv4 will not be used on this connection.  This property
 must be set.

nmcli> set ipv4.method auto
nmcli> save
Saving the connection with 'autoconnect=yes'. That might result in an immediate activation
 of the connection.
Do you still want to save? [yes] yes
Connection 'eth0' (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.

Adding a Static Ethernet Connection

To add an Ethernet connection with static `IPv4` configuration, a command in the following format can be used:

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

`IPv6` address and gateway information can be added using the **`ip6`** and **`gw6`** options.

For example, a command to create a static Ethernet connection with only `IPv4` address and gateway is as follows:

```
~]$ nmcli con add type ethernet con-name test-lab ifname eth1 ip4 10.10.10.10/24 \
gw4 10.10.10.254
```

Optionally, at the same time specify `IPv6` address and gateway for the device as follows:

```
~]$ nmcli con add type ethernet con-name test-lab ifname eth1 ip4 10.10.10.10/24 \
gw4 10.10.10.254 ip6 abbe::cafe gw6 2001:db8::1
Connection 'test-lab' (05abfd5e-324e-4461-844e-8501ba704773) successfully added.
```

NetworkManager will set its internal parameter **`ipv4.method`** to **`manual`** and **`connection.autoconnect`** to **`yes`**. **NetworkManager** will also write out settings to **`/etc/sysconfig/network-scripts/ifcfg-my-office`** where the corresponding BOOTPROTO will be set to **`none`** and ONBOOT will be set to **`yes`**.

Note that manual changes to the ifcfg file will not be noticed by **NetworkManager** until the interface is next brought up. See *Section 1.7, “Network Configuration Using sysconfig Files”* for more information on using configuration files.

To set 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.

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.

Alternatively, to add additional `DNS` servers to any previously set, use the **+** prefix as follows:

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

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

To bring up the new Ethernet connection, issue a command as follows:

```
~]$ nmcli con up test-lab ifname eth1
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/
ActiveConnection/2
```

Review the status of the devices and connections:

```
~]$ nmcli device status
DEVICE  TYPE      STATE      CONNECTION
eth0    ethernet  connected  my-office
eth1    ethernet  connected  test-lab
lo      loopback  unmanaged  --
```

To view detailed information about the newly configured connection, issue a command as follows:

```
~]$ 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:              eth1
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.

Modifying a Static Ethernet Connection

To replace existing `IP` addresses, enter a command as follows:

```
~]$ nmcli con mod test-lab ipv4.addresses "192.168.112.112"
```

To add additional `IP` addresses, use the **+** prefix as follows:

```
~]$ nmcli con mod test-lab +ipv4.addresses "192.168.114.114"
```

Example 2.2. Configuring a Static Ethernet Connection Using the Interactive Editor

To configure a static Ethernet connection using the interactive editor, issue commands as follows:

```
~]$ nmcli con edit type ethernet con-name eth0

===| 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>
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] no
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 'eth0' (704a5666-8cbd-4d89-b5f9-fa65a3dbc916) successfully saved.
nmcli> quit
~]$
```

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

Locking a Profile to a Specific Device

To lock a profile to a specific interface device, the commands used in the examples above include the interface name. For example:

```
nmcli connection add type ethernet con-name connection-name ifname interface-name
```

To make a profile usable for all compatible Ethernet interfaces, issue a command as follows:

```
nmcli connection add type ethernet con-name connection-name ifname "*"
```

Note that you have to use the **`ifname`** argument even if you do not want to set a specific interface. Use the wildcard character **`*`** to specify that the profile can be used with any compatible device.

To lock a profile to a specific MAC address, use a command in the following format:

```
nmcli connection add type ethernet con-name "connection-name" ifname "*" mac
 00:00:5E:00:53:00
```

Adding a Wi-Fi Connection

To view the available Wi-Fi access points, issue a command as follows:

```
~]$ nmcli dev wifi list
  SSID            MODE   CHAN  RATE     SIGNAL  BARS  SECURITY
  FedoraTest     Infra  11    54 MB/s  98      #▄#█  WPA1
  Red Hat Guest  Infra  6     54 MB/s  97      #▄#█  WPA2
  Red Hat        Infra  6     54 MB/s  77      #▄#_  WPA2 802.1X
* Red Hat        Infra  40    54 MB/s  66      #▄#_  WPA2 802.1X
  VoIP           Infra  1     54 MB/s  32      #▄__  WEP
  MyCafe         Infra  11    54 MB/s  39      #▄__  WPA2
```

To create a Wi-Fi connection profile with static `IP` configuration, but allowing automatic `DNS` address assignment, issue a command as follows:

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

To set a WPA2 password, for example “caffeine”, issue commands as follows:

```
~]$ nmcli con modify MyCafe wifi-sec.key-mgmt wpa-psk
~]$ nmcli con modify MyCafe wifi-sec.psk caffeine
```

To change Wi-Fi state, issue a command in the following format:

```
~]$ nmcli radio wifi [on | off ]
```

Changing a Specific Property

To check a specific property, for example **`mtu`**, issue a command as follows:

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

To change the property of a setting, issue a command as follows:

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

To verify the change, issue a command as follows:

```
~]$ nmcli connection show id 'MyCafe' | grep mtu
```

```
802-11-wireless.mtu:                     1350
```

Note that **NetworkManager** refers to parameters such as **`802-3-ethernet`** and **`802-11-wireless`** as the setting, and **`mtu`** as a property of the setting. See the **`nm-settings(5)`** man page for more information on properties and their settings.

2.4.3. Configuring Static Routes Using nmcli

To configure static routes using the **nmcli** tool, the command line or the interactive editor mode can be used.

Example 2.3. Configuring Static Routes Using nmcli

To configure a static route for an existing Ethernet connection using the command line, enter a command as follows:

```
~]# nmcli connection modify eth0 +ipv4.routes "192.168.122.0/24 10.10.10.1"
```

This will direct traffic for the `192.168.122.0/24` subnet to the gateway at `10.10.10.1`.

Example 2.4. Configuring Static Routes Using nmcli Editor

To configure a static route for an Ethernet connection using the interactive editor, issue commands as follows:

```
~]$ nmcli con edit type ethernet con-name eth0

===| 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.routes 192.168.122.0/24 10.10.10.1
nmcli>
nmcli> save persistent
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 'eth0' (704a5666-8cbd-4d89-b5f9-fa65a3dbc916) successfully saved.
nmcli> quit
~]$
```

2.5. Additional Resources

The following sources of information provide additional resources relevant to this chapter.

2.5.1. Installed Documentation

- **`ip(8)`** man page — Describes the **ip** utility's command syntax.
- **`nmcli(1)`** man page — Describes **NetworkManager**'s command-line tool.
- **`nmcli-examples(5)`** man page — Gives examples of **nmcli** commands.
- **`nm-settings(5)`** man page — Describes **NetworkManager** properties and their settings.

2.5.2. Online Documentation

RFC 1518[13] — Classless Inter-Domain Routing (CIDR)

Describes the CIDR Address Assignment and Aggregation Strategy, including variable-length subnetting.

RFC 1918[14] — Address Allocation for Private Internets

Describes the range of `IPv4` addresses reserved for private use.

RFC 3330[15] — Special-Use IPv4 Addresses

Describes the global and other specialized `IPv4` address blocks that have been assigned by the Internet Assigned Numbers Authority (IANA).

[13] http://www.rfc-editor.org/info/rfc1518
[14] http://www.rfc-editor.org/info/rfc1918
[15] http://www.rfc-editor.org/info/rfc3330

Configure Host Names

3.1. Understanding Host Names

There are three classes of `hostname`: static, pretty, and transient.

The “static” host name is the traditional `hostname`, which can be chosen by the user, and is stored in the **`/etc/hostname`** file. The “transient” `hostname` is a dynamic host name maintained by the kernel. It is initialized to the static host name by default, whose value defaults to “localhost”. It can be changed by `DHCP` or `mDNS` at runtime. The “pretty” `hostname` is a free-form UTF8 host name for presentation to the user.

Note

A host name can be a free-form string up to 64 characters in length. However, Red Hat recommends that both static and transient names match the *fully-qualified domain name* (FQDN) used for the machine in `DNS`, such as `host.example.com`. It is also recommended that the static and transient names consists only of 7 bit ASCII lower-case characters, no spaces or dots, and limits itself to the format allowed for `DNS` domain name labels, even though this is not a strict requirement. Older specifications do not permit the underscore, and so their use is not recommended.

The **hostnamectl** tool will enforce the following: Static and transient host names to consist of **`a-z`**, **`A-Z`**, **`0-9`**, “**`-`**”, “_” and “**`.`**” only, to not begin or end in a dot, and to not have two dots immediately following each other. The size limit of 64 characters is enforced.

3.1.1. Recommended Naming Practices

The Internet Corporation for Assigned Names and Numbers (ICANN) sometimes adds previously unregistered Top-Level Domains (such as `.yourcompany`) to the public register. Therefore, Red Hat strongly recommends that you do not use a domain name that is not delegated to you, even on a private network, as this can result in a domain name that resolves differently depending on network configuration. As a result, network resources can become unavailable. Using domain names that are not delegated to you also makes DNSSEC more difficult to deploy and maintain, as domain name collisions require manual configuration to enable DNSSEC validation. See the *ICANN FAQ on domain name collision*[1] for more information on this issue.

3.2. Configuring Host Names Using hostnamectl

The **hostnamectl** tool is provided for administering the three separate classes of host names in use on a given system.

3.2.1. View All the Host Names

To view all the current host names, enter the following command:

```
~]$ hostnamectl status
```

[1] http://www.icann.org/en/help/name-collision/faqs

The **`status`** option is implied by default if no option is given.

3.2.2. Set All the Host Names

To set all the host names on a system, enter the following command as `root`:

```
~]# hostnamectl set-hostname name
```

This will alter the pretty, static, and transient host names alike. The static and transient host names will be simplified forms of the pretty host name. Spaces will be replaced with “**`-`**” and special characters will be removed.

3.2.3. Set a Particular Host Name

To set a particular host name, enter the following command as `root` with the relevant option:

```
~]# hostnamectl set-hostname name [option...]
```

Where *`option`* is one or more of: **`--pretty`**, **`--static`**, and **`--transient`**.

If the **`--static`** or **`--transient`** options are used together with the **`--pretty`** option, the static and transient host names will be simplified forms of the pretty host name. Spaces will be replaced with “**`-`**” and special characters will be removed. If the **`--pretty`** option is not given, no simplification takes place.

When setting a pretty host name, remember to use the appropriate quotation marks if the host name contains spaces or a single quotation mark. For example:

```
~]# hostnamectl set-hostname "Stephen's notebook" --pretty
```

3.2.4. Clear a Particular Host Name

To clear a particular host name and allow it to revert to the default, enter the following command as `root` with the relevant option:

```
~]# hostnamectl set-hostname "" [option...]
```

Where *`""`* is a quoted empty string and where *`option`* is one or more of: **`--pretty`**, **`--static`**, and **`--transient`**.

3.2.5. Changing Host Names Remotely

To execute a **`hostnamectl`** command on a remote system, use the **`-H, --host`** option as follows:

```
~]# hostnamectl set-hostname -H [username]@hostname
```

Where *`hostname`* is the remote host you want to configure. The *`username`* is optional. The **hostnamectl** tool will use SSH to connect to the remote system.

3.3. Configuring Host Names Using nmcli

The **NetworkManager** tool **nmcli** can be used to query and set the static host name in the **`/etc/hostname`** file. Note that at time of writing, changing the host name in this way will not be noticed by **hostnamectl**.

To query the static host name, issue the following command:

```
~]$ nmcli general hostname
```

To set the static host name to *my-server*, issue the following command as `root`:

```
~]# nmcli general hostname my-server
```

To force **hostnamectl** to notice the change in the static host name, restart `hostnamed` as `root`:

```
~]# systemctl restart systemd-hostnamed
```

3.4. Additional Resources

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

3.4.1. Installed Documentation

- **`hostnamectl(1)`** man page — Describes **hostnamectl** including the commands and command options.
- **`hostname(1)`** man page — Contains an explanation of the **`hostname`** and **`domainname`** commands.
- **`hostname(5)`** man page — Contains an explanation of the host name file, its contents, and use.
- **`hostname(7)`** man page — Contains an explanation of host name resolution.
- **`machine-info(5)`** man page — Describes the local machine information file and the environment variables it contains.
- **`machine-id(5)`** man page — Describes the local machine ID configuration file.
- **`systemd-hostnamed.service(8)`** man page — Describes the `systemd-hostnamed` system service used by **hostnamectl**.

3.4.2. Online Documentation

http://www.freedesktop.org/wiki/Software/systemd/hostnamed
    Information on `systemd-hostnamed`.

Configure Network Bonding

Fedora allows administrators to bind multiple network interfaces together into a single, bonded, channel. Channel bonding enables two or more network interfaces to act as one, simultaneously increasing the bandwidth and providing redundancy.

Warning

The use of direct cable connections without network switches is not supported for bonding. The failover mechanisms described here will not work as expected without the presence of network switches.

Note

The active-backup, balance-tlb and balance-alb modes do not require any specific configuration of the switch. Other bonding modes require configuring the switch to aggregate the links. For example, a Cisco switch requires EtherChannel for Modes 0, 2, and 3, but for Mode 4 LACP and EtherChannel are required. See the documentation supplied with your switch and see *https://www.kernel.org/doc/Documentation/networking/bonding.txt*.

4.1. Understanding the Default Behavior of Master and Slave Interfaces

When controlling bonded slave interfaces using the `NetworkManager` daemon, and especially when fault finding, keep the following in mind:

1. Starting the master interface does not automatically start the slave interfaces.
2. Starting a slave interface always starts the master interface.
3. Stopping the master interface also stops the slave interfaces.
4. A master without slaves can start static `IP` connections.
5. A master without slaves waits for slaves when starting `DHCP` connections.
6. A master with a `DHCP` connection waiting for slaves completes when a slave with a carrier is added.
7. A master with a `DHCP` connection waiting for slaves continues waiting when a slave without a carrier is added.

4.2. Creating a Bond Connection Using a GUI

You can use the GNOME **control-center** utility to direct **NetworkManager** to create a Bond from two or more Wired or InfiniBand connections. It is not necessary to create the connections to be bonded first. They can be configured as part of the process to configure the bond. You must have the MAC addresses of the interfaces available in order to complete the configuration process.

4.2.1. Establishing a Bond Connection

Procedure 4.1. Adding a New Bond Connection

You can configure a Bond connection by opening the **Network** window, clicking the plus symbol, and selecting **Bond** from the list.

1. Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.

2. Click the plus symbol to open the selection list. Select **Bond**. The **Editing Bond connection *1*** window appears.

3. On the **Bond** tab, click **Add** and select the type of interface you want to use with the bond connection. Click the **Create** button. Note that the dialog to select the slave type only comes up when you create the first slave; after that, it will automatically use that same type for all further slaves.

4. The **Editing bond0 slave 1** window appears. Use the **Device MAC address** drop-down menu to select the MAC address of the interface to be bonded. The first slave's MAC address will be used as the MAC address for the bond interface. If required, enter a clone MAC address to be used as the bond's MAC address. Click the **Save** button.

5. The name of the bonded slave appears in the **Bonded connections** window. Click the **Add** button to add further slave connections.

6. Review and confirm the settings and then click the **Save** button.

7. Edit the bond-specific settings by referring to *Section 4.2.1.1, “Configuring the Bond Tab”* below.

Procedure 4.2. Editing an Existing Bond Connection

Follow these steps to edit an existing bond connection.

1. Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.

2. Select the connection you want to edit and click the **Options** button.

3. Select the **General** tab.

4. Configure the connection name, auto-connect behavior, and availability settings.

   Five settings in the **Editing** dialog are common to all connection types, see the **General** tab:

   - **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.

   - **Automatically connect to this network when it is available** — Select this box if you want **NetworkManager** to auto-connect to this connection when it is available. See *Section 2.2.3, “Connecting to a Network Automatically”* for more information.

   - **All users may connect to this network** — Select this box to create a connection available to all users on the system. Changing this setting may require root privileges. See *Section 2.2.4, “System-wide and Private Connection Profiles”* for details.

   - **Automatically connect to VPN when using this connection** — Select this box if you want **NetworkManager** to auto-connect to a VPN connection when it is available. Select the VPN from the drop-down menu.

- **Firewall Zone** — Select the firewall zone from the drop-down menu.

5. Edit the bond-specific settings by referring to *Section 4.2.1.1, “Configuring the Bond Tab”* below.

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your bond connection, click the **Save** button to save your customized configuration. If the profile was in use while being edited, power cycle the connection to make **NetworkManager** apply the changes. If the profile is OFF, set it to ON. See *Section 2.2.1, “Connecting to a Network Using a GUI”* for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network** window and clicking **Options** to return to the **Editing** dialog.

Then, to configure:

- `IPv4` settings for the connection, click the **IPv4 Settings** tab and proceed to *Section 2.2.10.4, “Configuring IPv4 Settings”*; or,
- `IPv6` settings for the connection, click the **IPv6 Settings** tab and proceed to *Section 2.2.10.5, “Configuring IPv6 Settings”*.

4.2.1.1. Configuring the Bond Tab

If you have already added a new bond connection (refer to *Procedure 4.1, “Adding a New Bond Connection”* for instructions), you can edit the **Bond** tab to set the load sharing mode and the type of link monitoring to use to detect failures of a slave connection.

Mode

The mode that is used to share traffic over the slave connections which make up the bond. The default is **Round-robin**. Other load sharing modes, such as `802.3ad`, can be selected by means of the drop-down list.

Link Monitoring

The method of monitoring the slaves ability to carry network traffic.

The following modes of load sharing are selectable from the **Mode** drop-down list:

Round-robin

Sets a round-robin policy for fault tolerance and load balancing. Transmissions are received and sent out sequentially on each bonded slave interface beginning with the first one available. This mode might not work behind a bridge with virtual machines without additional switch configuration.

Active backup

Sets an active-backup policy for fault tolerance. Transmissions are received and sent out via the first available bonded slave interface. Another bonded slave interface is only used if the active bonded slave interface fails. Note that this is the only mode available for bonds of InfiniBand devices.

XOR

Sets an XOR (exclusive-or) policy. Transmissions are based on the selected hash policy. The default is to derive a hash by XOR of the source and destination MAC addresses multiplied by the modulo of the number of slave interfaces. In this mode traffic destined for specific peers will always be sent over the same interface. As the destination is determined by the MAC addresses this method works best for traffic to peers on the same link or local network. If traffic has to pass through a single router then this mode of traffic balancing will be suboptimal.

Broadcast
: Sets a broadcast policy for fault tolerance. All transmissions are sent on all slave interfaces. This mode might not work behind a bridge with virtual machines without additional switch configuration.

802.3ad
: Sets an IEEE `802.3ad` dynamic link aggregation policy. Creates aggregation groups that share the same speed and duplex settings. Transmits and receives on all slaves in the active aggregator. Requires a network switch that is `802.3ad` compliant.

Adaptive transmit load balancing
: Sets an adaptive Transmit Load Balancing (TLB) policy for fault tolerance and load balancing. The outgoing traffic is distributed according to the current load on each slave interface. Incoming traffic is received by the current slave. If the receiving slave fails, another slave takes over the MAC address of the failed slave. This mode is only suitable for local addresses known to the kernel bonding module and therefore cannot be used behind a bridge with virtual machines.

Adaptive load balancing
: Sets an Adaptive Load Balancing (ALB) policy for fault tolerance and load balancing. Includes transmit and receive load balancing for `IPv4` traffic. Receive load balancing is achieved through ARP negotiation. This mode is only suitable for local addresses known to the kernel bonding module and therefore cannot be used behind a bridge with virtual machines.

The following types of link monitoring can be selected from the **Link Monitoring** drop-down list. It is a good idea to test which channel bonding module parameters work best for your bonded interfaces.

MII (Media Independent Interface)
: The state of the carrier wave of the interface is monitored. This can be done by querying the driver, by querying MII registers directly, or by using **ethtool** to query the device. Three options are available:

  Monitoring Frequency
  : The time interval, in milliseconds, between querying the driver or MII registers.

  Link up delay
  : The time in milliseconds to wait before attempting to use a link that has been reported as up. This delay can be used if some gratuitous ARP requests are lost in the period immediately following the link being reported as “up”. This can happen during switch initialization for example.

  Link down delay
  : The time in milliseconds to wait before changing to another link when a previously active link has been reported as “down”. This delay can be used if an attached switch takes a relatively long time to change to backup mode.

ARP
: The address resolution protocol (ARP) is used to probe one or more peers to determine how well the link-layer connections are working. It is dependent on the device driver providing the transmit start time and the last receive time.

  Two options are available:

  Monitoring Frequency
  : The time interval, in milliseconds, between sending ARP requests.

ARP targets
: A comma separated list of `IP` addresses to send ARP requests to.

4.3. Using the Command Line Interface (CLI)

A bond is created using the **`bonding`** kernel module and a special network interface called a *channel bonding interface*.

4.3.1. Check if Bonding Kernel Module is Installed

In Fedora, the bonding module is not loaded by default. You can load the module by issuing the following command as `root`:

```
~]# modprobe --first-time bonding
```

This activation will not persist across system restarts. See the *Fedora 24 System Administrator's Guide*[1] for an explanation of persistent module loading. Note that given a correct configuration file using the **`BONDING_OPTS`** directive, the bonding module will be loaded as required and therefore does not need to be loaded separately.

To display information about the module, issue the following command:

```
~]$ modinfo bonding
```

See the **`modprobe(8)`** man page for more command options.

4.3.2. Create a Channel Bonding Interface

To create a channel bonding interface, create a file in the **`/etc/sysconfig/network-scripts/`** directory called **`ifcfg-bondN`**, replacing *N* with the number for the interface, such as **`0`**.

The contents of the file can be based on a configuration file for whatever type of interface is getting bonded, such as an Ethernet interface. The essential differences are that the **`DEVICE`** directive is **`bondN`**, replacing *N* with the number for the interface, and **`TYPE=Bond`**. In addition, set **`BONDING_MASTER=yes`**.

Example 4.1. Example ifcfg-bond0 Interface Configuration File

An example of a channel bonding interface.

```
DEVICE=bond0
NAME=bond0
TYPE=Bond
BONDING_MASTER=yes
IPADDR=192.168.1.1
PREFIX=24
ONBOOT=yes
BOOTPROTO=none
BONDING_OPTS="bonding parameters separated by spaces"
```

The NAME directive is useful for naming the connection profile in **NetworkManager**. ONBOOT says whether the profile should be started when booting (or more generally, when auto-connecting a device).

[1] https://docs.fedoraproject.org/sysadmin-guide

Put all Bonding Module Parameters in ifcfg-bondN Files

Parameters for the bonding kernel module must be specified as a space-separated list in the **`BONDING_OPTS="`*`bonding parameters`*`"`** directive in the **`ifcfg-bond`*`N`*** interface file. Do *not* specify options for the bonding device in **`/etc/modprobe.d/`*`bonding`*`.conf`**, or in the deprecated **`/etc/modprobe.conf`** file.

The **`max_bonds`** parameter is not interface specific and should not be set when using **`ifcfg-bond`*`N`*** files with the **`BONDING_OPTS`** directive as this directive will cause the network scripts to create the bond interfaces as required.

For further instructions and advice on configuring the bonding module and to view the list of bonding parameters, see *Section 4.4, "Using Channel Bonding"*.

4.3.3. Creating SLAVE Interfaces

The channel bonding interface is the "master" and the interfaces to be bonded are referred to as the "slaves". After the channel bonding interface is created, the network interfaces to be bound together must be configured by adding the **`MASTER`** and **`SLAVE`** directives to the configuration files of the slaves. The configuration files for each of the slave interfaces can be nearly identical.

Example 4.2. Example Slave Interface Configuration File

For example, if two Ethernet interfaces are being channel bonded, **`eth0`** and **`eth1`**, they can both look like the following example:

```
DEVICE=ethN
NAME=bond0-slave
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

In this example, replace *N* with the numerical value for the interface. Note that if more than one profile or configuration file exists with ONBOOT=yes for an interface, they may race with each other and a plain TYPE=Ethernet profile may be activated instead of a bond slave.

4.3.4. Activating a Channel Bond

To activate a bond, bring up all the slaves. As `root`, issue the following commands:

```
~]# ifup ifcfg-eth0
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/
ActiveConnection/7)
```

```
~]# ifup ifcfg-eth1
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/
ActiveConnection/8)
```

Note that if editing interface files for interfaces which are currently "up", set them down first as follows:

```
ifdown ethN
```

Then when complete, bring up all the slaves, which will bring up the bond (provided it was not set "down").

To make **NetworkManager** aware of the changes, issue a command for every changed interface as `root`:

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

Alternatively, to reload all interfaces:

```
~]# nmcli con reload
```

The default behavior is for **NetworkManager** not to be aware of the changes and to continue using the old configuration data. This is set by the **`monitor-connection-files`** option in the **`NetworkManager.conf`** file. See the **`NetworkManager.conf(5)`** manual page for more information.

To view the status of the bond interface, issue the following command:

```
~]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master bond0 state
 UP mode DEFAULT qlen 1000
    link/ether 52:54:00:e9:ce:d2 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master bond0 state
 UP mode DEFAULT qlen 1000
    link/ether 52:54:00:38:a6:4c brd ff:ff:ff:ff:ff:ff
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode
 DEFAULT
    link/ether 52:54:00:38:a6:4c brd ff:ff:ff:ff:ff:ff
```

4.3.5. Creating Multiple Bonds

In Fedora 24, for each bond a channel bonding interface is created including the **`BONDING_OPTS`** directive. This configuration method is used so that multiple bonding devices can have different configurations. To create multiple channel bonding interfaces, proceed as follows:

- Create multiple **`ifcfg-bond`*`N`*** files with the **`BONDING_OPTS`** directive; this directive will cause the network scripts to create the bond interfaces as required.
- Create, or edit existing, interface configuration files to be bonded and include the **`SLAVE`** directive.
- Assign the interfaces to be bonded, the slave interfaces, to the channel bonding interfaces by means of the **`MASTER`** directive.

Example 4.3. Example multiple ifcfg-bondN interface configuration files

The following is an example of a channel bonding interface configuration file:

```
DEVICE=bondN
NAME=bondN
TYPE=Bond
BONDING_MASTER=yes
IPADDR=192.168.1.1
PREFIX=24
ONBOOT=yes
BOOTPROTO=none
BONDING_OPTS="bonding parameters separated by spaces"
```

> In this example, replace *N* with the number for the bond interface. For example, to create two bonds create two configuration files, **`ifcfg-bond0`** and **`ifcfg-bond1`**, with appropriate `IP` addresses.

Create the interfaces to be bonded as per *Example 4.2, "Example Slave Interface Configuration File"* and assign them to the bond interfaces as required using the **`MASTER=bond`*`N`*** directive. For example, continuing on from the example above, if two interfaces per bond are required, then for two bonds create four interface configuration files and assign the first two using **`MASTER=bond`*`0`*** and the next two using **`MASTER=bond`*`1`***.

4.3.6. Configuring a VLAN over a Bond

This section will show configuring a VLAN over a bond consisting of two Ethernet links between a server and an Ethernet switch. The switch has a second bond to another server. Only the configuration for the first server will be shown as the other is essentially the same apart from the `IP` addresses.

Warning

The use of direct cable connections without network switches is not supported for bonding. The failover mechanisms described here will not work as expected without the presence of network switches.

Note

The active-backup, balance-tlb and balance-alb modes do not require any specific configuration of the switch. Other bonding modes require configuring the switch to aggregate the links. For example, a Cisco switch requires EtherChannel for Modes 0, 2, and 3, but for Mode 4 LACP and EtherChannel are required. See the documentation supplied with your switch and see *https://www.kernel.org/doc/Documentation/networking/bonding.txt*.

Check the available interfaces on the server:

```
~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether 52:54:00:f6:63:9a brd ff:ff:ff:ff:ff:ff
```

Procedure 4.3. Configuring the Interfaces on the Server

1. Configure a slave interface using **`eth0`**:

```
~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
NAME=bond0-slave0
DEVICE=eth0
TYPE=Ethernet
```

```
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

The use of the NAME directive is optional. It is for display by a GUI interface, such as **nm-connection-editor**.

2. Configure a slave interface using **`eth1`**:

```
~]# vi /etc/sysconfig/network-scripts/ifcfg-eth1
NAME=bond0-slave1
DEVICE=eth1
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

The use of the NAME directive is optional. It is for display by a GUI interface, such as **nm-connection-editor**.

3. Configure a channel bonding interface **`ifcfg-bond0`**:

```
~]# vi /etc/sysconfig/network-scripts/ifcfg-bond0
NAME=bond0
DEVICE=bond0
BONDING_MASTER=yes
TYPE=Bond
IPADDR=192.168.100.100
NETMASK=255.255.255.0
ONBOOT=yes
BOOTPROTO=none
BONDING_OPTS="mode=active-backup miimon=100"
```

The use of the NAME directive is optional. It is for display by a GUI interface, such as **nm-connection-editor**. In this example MII is used for link monitoring, see the *Section 4.4.1, "Bonding Module Directives"* section for more information on link monitoring.

4. Check the status of the interfaces on the server:

```
~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
    inet6 fe80::5054:ff:fe19:28fe/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:f6:63:9a brd ff:ff:ff:ff:ff:ff
    inet6 fe80::5054:ff:fef6:639a/64 scope link
       valid_lft forever preferred_lft forever
```

Procedure 4.4. Resolving Conflicts with Interfaces

The interfaces configured as slaves should not have `IP` addresses assigned to them apart from the `IPv6` link-local addresses (starting **`fe80`**). If you have an unexpected `IP` address, then there may be another configuration file with ONBOOT set to **`yes`**.

1. If this occurs, issue the following command to list all **`ifcfg`** files that may be causing a conflict:

```
~]$ grep -r "ONBOOT=yes" /etc/sysconfig/network-scripts/ | cut -f1 -d":" | xargs grep -E
 "IPADDR|SLAVE"
/etc/sysconfig/network-scripts/ifcfg-lo:IPADDR=127.0.0.1
```

   The above shows the expected result on a new installation. Any file having both the ONBOOT directive as well as the IPADDR or SLAVE directive will be displayed. For example, if the **`ifcfg-eth1`** file was incorrectly configured, the display might look similar to the following:

```
~]# grep -r "ONBOOT=yes" /etc/sysconfig/network-scripts/ | cut -f1 -d":" | xargs grep -E
 "IPADDR|SLAVE"
/etc/sysconfig/network-scripts/ifcfg-lo:IPADDR=127.0.0.1
/etc/sysconfig/network-scripts/ifcfg-eth1:SLAVE=yes
/etc/sysconfig/network-scripts/ifcfg-eth1:IPADDR=192.168.55.55
```

2. Any other configuration files found should be moved to a different directory for backup, or assigned to a different interface by means of the HWADDR directive. After resolving any conflict set the interfaces “down” and “up” again or restart the network service as `root`:

```
~]# systemctl restart network
Shutting down interface bond0:                             [  OK  ]
Shutting down loopback interface:                          [  OK  ]
Bringing up loopback interface:                            [  OK  ]
Bringing up interface bond0:  Determining if ip address 192.168.100.100 is already in
 use for device bond0...
                                                           [  OK  ]
```

   If you are using **NetworkManager**, you might need to restart it at this point to make it forget the unwanted `IP` address. As `root`:

```
~]# systemctl restart NetworkManager
```

Procedure 4.5. Checking the bond on the Server

1. Bring up the bond on the server as `root`:

```
~]# ifup /etc/sysconfig/network-scripts/ifcfg-bond0
Determining if ip address 192.168.100.100 is already in use for device bond0...
```

2. Check the status of the interfaces on the server:

```
~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master bond0
 state UP qlen 1000
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master bond0
 state UP qlen 1000
    link/ether 52:54:00:f6:63:9a brd ff:ff:ff:ff:ff:ff
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.100/24 brd 192.168.100.255 scope global bond0
    inet6 fe80::5054:ff:fe19:28fe/64 scope link
```

```
       valid_lft forever preferred_lft forever
```

Notice that **eth0** and **eth1** have **master bond0 state UP** and **bond0** has status of **MASTER,UP**.

3. View the bond configuration details:

```
~]$ cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.6.0 (September 26, 2009)

Bonding Mode: transmit load balancing
Primary Slave: None
Currently Active Slave: eth0
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth0
MII Status: up
Speed: 100 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 52:54:00:19:28:fe
Slave queue ID: 0

Slave Interface: eth1
MII Status: up
Speed: 100 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 52:54:00:f6:63:9a
Slave queue ID: 0
```

4. Check the routes on the server:

```
~]$ ip route
192.168.100.0/24 dev bond0  proto kernel  scope link  src 192.168.100.100
169.254.0.0/16 dev bond0  scope link  metric 1004
```

Procedure 4.6. Configuring the VLAN on the Server

Note

A VLAN slave cannot be configured on a bond with the **fail_over_mac=follow** option, because the VLAN virtual device cannot change its MAC address to match the parent's new MAC address. In such a case, traffic would still be sent with the now incorrect source MAC address.

Some older network interface cards, loopback interfaces, Wimax cards, and some Infiniband devices, are said to be *VLAN challenged*, meaning they cannot support VLANs. This is usually because the devices cannot cope with VLAN headers and the larger MTU size associated with VLANs.

1. Create a VLAN interface file **bond0.192**:

```
~]# vi /etc/sysconfig/network-scripts/ifcfg-bond0.192
```

```
DEVICE=bond0.192
NAME=bond0.192
BOOTPROTO=none
ONPARENT=yes
IPADDR=192.168.10.1
NETMASK=255.255.255.0
VLAN=yes
```

2. Bring up the VLAN interface as `root`:

```
~]# ifup /etc/sysconfig/network-scripts/ifcfg-bond0.192
Determining if ip address 192.168.10.1 is already in use for device bond0.192...
```

3. Enabling VLAN tagging on the network switch. Consult the documentation for the switch to see what configuration is required.

4. Check the status of the interfaces on the server:

```
~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master bond0
 state UP qlen 1000
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master bond0
 state UP qlen 1000
    link/ether 52:54:00:f6:63:9a brd ff:ff:ff:ff:ff:ff
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.100/24 brd 192.168.100.255 scope global bond0
    inet6 fe80::5054:ff:fe19:28fe/64 scope link
       valid_lft forever preferred_lft forever
5: bond0.192@bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue
 state UP
    link/ether 52:54:00:19:28:fe brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.1/24 brd 192.168.10.255 scope global bond0.192
    inet6 fe80::5054:ff:fe19:28fe/64 scope link
       valid_lft forever preferred_lft forever
```

Notice there is now **`bond0.192@bond0`** in the list of interfaces and the status is **`MASTER,UP`**.

5. Check the route on the server:

```
~]$ ip route
192.168.100.0/24 dev bond0  proto kernel  scope link  src 192.168.100.100
192.168.10.0/24 dev bond0.192  proto kernel  scope link  src 192.168.10.1
169.254.0.0/16 dev bond0  scope link  metric 1004
169.254.0.0/16 dev bond0.192  scope link  metric 1005
```

Notice there is now a route for the `192.168.10.0/24` network pointing to the VLAN interface **`bond0.192`**.

Configuring the Second Server

Repeat the configuration steps for the second server, using different `IP` addresses but from the same subnets respectively.

Test the bond is up and the network switch is working as expected:

```
~]$ ping -c4 192.168.100.100
PING 192.168.100.100 (192.168.100.100) 56(84) bytes of data.
64 bytes from 192.168.100.100: icmp_seq=1 ttl=64 time=1.35 ms
64 bytes from 192.168.100.100: icmp_seq=2 ttl=64 time=0.214 ms
64 bytes from 192.168.100.100: icmp_seq=3 ttl=64 time=0.383 ms
64 bytes from 192.168.100.100: icmp_seq=4 ttl=64 time=0.396 ms

--- 192.168.100.100 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.214/0.586/1.353/0.448 ms
```

Testing the VLAN

To test that the network switch is configured for the VLAN, try to ping the first servers' VLAN interface:

```
~]# ping -c2 192.168.10.1
PING 192.168.10.1 (192.168.10.1) 56(84) bytes of data.
64 bytes from 192.168.10.1: icmp_seq=1 ttl=64 time=0.781 ms
64 bytes from 192.168.10.1: icmp_seq=2 ttl=64 time=0.977 ms
--- 192.168.10.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.781/0.879/0.977/0.098 ms
```

No packet loss suggests everything is configured correctly and that the VLAN and underlying interfaces are “up”.

Optional Steps

- If required, perform further tests by removing and replacing network cables one at a time to verify that failover works as expected. Make use of the **ethtool** utility to verify which interface is connected to which cable. For example:

  ```
  ethtool --identify ifname integer
  ```

  Where *integer* is the number of times to flash the LED on the network interface.

- The bonding module does not support STP, therefore consider disabling the sending of BPDU packets from the network switch.

- If the system is not linked to the network except over the connection just configured, consider enabling the switch port to transition directly to sending and receiving. For example on a Cisco switch, by means of the **portfast** command.

4.4. Using Channel Bonding

To enhance performance, adjust available module options to ascertain what combination works best. Pay particular attention to the **miimon** or **arp_interval** and the **arp_ip_target** parameters. See *Section 4.4.1, “Bonding Module Directives”* for a list of available options and how to quickly determine the best ones for your bonded interface.

4.4.1. Bonding Module Directives

It is a good idea to test which channel bonding module parameters work best for your bonded interfaces before adding them to the **BONDING_OPTS="*bonding parameters*"** directive in your bonding interface configuration file (**ifcfg-bond0** for example). Parameters to bonded interfaces can be configured without unloading (and reloading) the bonding module by manipulating files in the `sysfs` file system.

`sysfs` is a virtual file system that represents kernel objects as directories, files and symbolic links. `sysfs` can be used to query for information about kernel objects, and can also manipulate those objects through the use of normal file system commands. The `sysfs` virtual file system is mounted under the **/sys/** directory. All bonding interfaces can be configured dynamically by interacting with and manipulating files under the **/sys/class/net/** directory.

In order to determine the best parameters for your bonding interface, create a channel bonding interface file such as **ifcfg-bond0** by following the instructions in *Section 4.3.2, "Create a Channel Bonding Interface"*. Insert the **SLAVE=yes** and **MASTER=bond0** directives in the configuration files for each interface bonded to **bond0**. Once this is completed, you can proceed to testing the parameters.

First, bring up the bond you created by running **ifup bond*N*** as `root`:

```
~]# ifup bond0
```

If you have correctly created the **ifcfg-bond0** bonding interface file, you will be able to see **bond0** listed in the output of running **ip link show** as `root`:

```
~]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master bond0 state
 UP mode DEFAULT qlen 1000
    link/ether 52:54:00:e9:ce:d2 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master bond0 state
 UP mode DEFAULT qlen 1000
    link/ether 52:54:00:38:a6:4c brd ff:ff:ff:ff:ff:ff
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode
 DEFAULT
    link/ether 52:54:00:38:a6:4c brd ff:ff:ff:ff:ff:ff
```

To view all existing bonds, even if they are not up, run:

```
~]$ cat /sys/class/net/bonding_masters
bond0
```

You can configure each bond individually by manipulating the files located in the **/sys/class/net/bond*N*/bonding/** directory. First, the bond you are configuring must be taken down:

```
~]# ifdown bond0
```

As an example, to enable MII monitoring on bond0 with a 1 second interval, run as `root`:

```
~]# echo 1000 > /sys/class/net/bond0/bonding/miimon
```

To configure bond0 for *`balance-alb`* mode, run either:

```
~]# echo 6 > /sys/class/net/bond0/bonding/mode
```

...or, using the name of the mode:

```
~]# echo balance-alb > /sys/class/net/bond0/bonding/mode
```

After configuring options for the bond in question, you can bring it up and test it by running **`ifup bond`*`N`***. If you decide to change the options, take the interface down, modify its parameters using `sysfs`, bring it back up, and re-test.

Once you have determined the best set of parameters for your bond, add those parameters as a space-separated list to the *`BONDING_OPTS=`* directive of the **`/etc/sysconfig/network-scripts/ifcfg-bond`*`N`*** file for the bonding interface you are configuring. Whenever that bond is brought up (for example, by the system during the boot sequence if the *`ONBOOT=yes`* directive is set), the bonding options specified in the *`BONDING_OPTS`* will take effect for that bond.

The following list provides the names of many of the more common channel bonding parameters, along with a description of what they do. For more information, see the brief descriptions for each **`parm`** in **`modinfo bonding`** output, or for more detailed information, see *https://www.kernel.org/doc/Documentation/networking/bonding.txt*.

Bonding Interface Parameters

`ad_select=`*`value`*

Specifies the 802.3ad aggregation selection logic to use. Possible values are:

- **`stable`** or **`0`** — Default setting. The active aggregator is chosen by largest aggregate bandwidth. Reselection of the active aggregator occurs only when all slaves of the active aggregator are down or if the active aggregator has no slaves.
- **`bandwidth`** or **`1`** — The active aggregator is chosen by largest aggregate bandwidth. Reselection occurs if:
  - A slave is added to or removed from the bond;
  - Any slave's link state changes;
  - Any slave's 802.3ad association state changes;
  - The bond's administrative state changes to up.
- **`count`** or **`2`** — The active aggregator is chosen by the largest number of slaves. Reselection occurs as described for the **`bandwidth`** setting above.

The **`bandwidth`** and **`count`** selection policies permit failover of 802.3ad aggregations when partial failure of the active aggregator occurs. This keeps the aggregator with the highest availability, either in bandwidth or in number of slaves, active at all times.

arp_all_targets=*`value`*

Specifies the quantity of **`arp_ip_target`**s that must be reachable in order for the ARP monitor to consider a slave as being up. This option affects only active-backup mode for slaves with **`arp_validation`** enabled. Possible values are:

- **`any`** or **`0`** — Default setting. Consider the slave up when any of the **`arp_ip_target`**s is reachable.
- **`all`** or **`1`** — Consider the slave up only when all of the **`arp_ip_target`**s are reachable.

`arp_interval=`*`time_in_milliseconds`*

Specifies, in milliseconds, how often ARP monitoring occurs.

Important

It is essential that both **`arp_interval`** and **`arp_ip_target`** parameters are specified, or, alternatively, the **`miimon`** parameter is specified. Failure to do so can cause degradation of network performance in the event that a link fails.

If using this setting while in **`mode=0`** or **`mode=2`** (the two load-balancing modes), the network switch must be configured to distribute packets evenly across the NICs. For more information on how to accomplish this, see *https://www.kernel.org/doc/Documentation/networking/bonding.txt*.

The value is set to **`0`** by default, which disables it.

`arp_ip_target=`*`ip_address`*`[,`*`ip_address_2`*`,…`*`ip_address_16`*`]`
Specifies the target `IP` address of ARP requests when the **`arp_interval`** parameter is enabled. Up to 16 `IP` addresses can be specified in a comma separated list.

`arp_validate=`*`value`*
Validate source/distribution of ARP probes; default is **none**. Other valid values are **`active`**, **`backup`**, and **`all`**.

`downdelay=`*`time_in_milliseconds`*
Specifies (in milliseconds) how long to wait after link failure before disabling the link. The value must be a multiple of the value specified in the **`miimon`** parameter. The value is set to **`0`** by default, which disables it.

`fail_over_mac=`*`value`*
Specifies whether active-backup mode should set all slaves to the same MAC address at enslavement (the traditional behavior), or, when enabled, perform special handling of the bond's MAC address in accordance with the selected policy. Possible values are:

- **none** or **`0`** — Default setting. This setting disables **`fail_over_mac`**, and causes bonding to set all slaves of an active-backup bond to the same MAC address at enslavement time.

- **`active`** or **`1`** — The "active" **`fail_over_mac`** policy indicates that the MAC address of the bond should always be the MAC address of the currently active slave. The MAC address of the slaves is not changed; instead, the MAC address of the bond changes during a failover.

  This policy is useful for devices that cannot ever alter their MAC address, or for devices that refuse incoming broadcasts with their own source MAC (which interferes with the ARP monitor). The disadvantage of this policy is that every device on the network must be updated via gratuitous ARP, as opposed to the normal method of switches snooping incoming traffic to update their ARP tables. If the gratuitous ARP is lost, communication may be disrupted.

  When this policy is used in conjunction with the MII monitor, devices which assert link up prior to being able to actually transmit and receive are particularly susceptible to loss of the gratuitous ARP, and an appropriate updelay setting may be required.

- **`follow`** or **`2`** — The "follow" **`fail_over_mac`** policy causes the MAC address of the bond to be selected normally (normally the MAC address of the first slave added to the bond). However, the second and subsequent slaves are not set to this MAC address while they are in a backup role; a slave is programmed with the bond's MAC address at failover time (and the formerly active slave receives the newly active slave's MAC address).

This policy is useful for multiport devices that either become confused or incur a performance penalty when multiple ports are programmed with the same MAC address.

lacp_rate=*`value`*
: Specifies the rate at which link partners should transmit LACPDU packets in 802.3ad mode. Possible values are:

- **`slow`** or **`0`** — Default setting. This specifies that partners should transmit LACPDUs every 30 seconds.
- **`fast`** or **`1`** — Specifies that partners should transmit LACPDUs every 1 second.

`miimon=`*`time_in_milliseconds`*
: Specifies (in milliseconds) how often MII link monitoring occurs. This is useful if high availability is required because MII is used to verify that the NIC is active. To verify that the driver for a particular NIC supports the MII tool, type the following command as root:

```
~]# ethtool interface_name | grep "Link detected:"
```

In this command, replace *`interface_name`*> with the name of the device interface, such as **`eth0`**, not the bond interface. If MII is supported, the command returns:

```
Link detected: yes
```

If using a bonded interface for high availability, the module for each NIC must support MII. Setting the value to **`0`** (the default), turns this feature off. When configuring this setting, a good starting point for this parameter is **`100`**.

Important

It is essential that both **`arp_interval`** and **`arp_ip_target`** parameters are specified, or, alternatively, the **`miimon`** parameter is specified. Failure to do so can cause degradation of network performance in the event that a link fails.

`mode=`*`value`*
: Allows you to specify the bonding policy. The *`value`* can be one of:

- **`balance-rr`** or **`0`** — Sets a round-robin policy for fault tolerance and load balancing. Transmissions are received and sent out sequentially on each bonded slave interface beginning with the first one available.
- **`active-backup`** or **`1`** — Sets an active-backup policy for fault tolerance. Transmissions are received and sent out via the first available bonded slave interface. Another bonded slave interface is only used if the active bonded slave interface fails.
- **`balance-xor`** or **`2`** — Transmissions are based on the selected hash policy. The default is to derive a hash by XOR of the source and destination MAC addresses multiplied by the modulo of the number of slave interfaces. In this mode traffic destined for specific peers will always be sent over the same interface. As the destination is determined by the MAC addresses this method works best for traffic to peers on the same link or local network. If traffic has to pass through a single router then this mode of traffic balancing will be suboptimal.

- **`broadcast`** or **`3`** — Sets a broadcast policy for fault tolerance. All transmissions are sent on all slave interfaces.
- **`802.3ad`** or **`4`** — Sets an IEEE 802.3ad dynamic link aggregation policy. Creates aggregation groups that share the same speed and duplex settings. Transmits and receives on all slaves in the active aggregator. Requires a switch that is 802.3ad compliant.
- **`balance-tlb`** or **`5`** — Sets a Transmit Load Balancing (TLB) policy for fault tolerance and load balancing. The outgoing traffic is distributed according to the current load on each slave interface. Incoming traffic is received by the current slave. If the receiving slave fails, another slave takes over the MAC address of the failed slave. This mode is only suitable for local addresses known to the kernel bonding module and therefore cannot be used behind a bridge with virtual machines.
- **`balance-alb`** or **`6`** — Sets an Adaptive Load Balancing (ALB) policy for fault tolerance and load balancing. Includes transmit and receive load balancing for `IPv4` traffic. Receive load balancing is achieved through ARP negotiation. This mode is only suitable for local addresses known to the kernel bonding module and therefore cannot be used behind a bridge with virtual machines.

`packets_per_slave=`*`range`*

Specify the number of packets to transmit through a slave before moving to the next one. When set to **`0`** then a slave is chosen at random.

The valid range is **`0`** to **`65535`**; the default value is **`1`**. This option has effect only in **`balance-rr`** mode.

`primary=`*`interface_name`*

Specifies the interface name, such as **`eth0`**, of the primary device. The **`primary`** device is the first of the bonding interfaces to be used and is not abandoned unless it fails. This setting is particularly useful when one NIC in the bonding interface is faster and, therefore, able to handle a bigger load.

This setting is only valid when the bonding interface is in **`active-backup`** mode. See *https://www.kernel.org/doc/Documentation/networking/bonding.txt* for more information.

`primary_reselect=`*`value`*

Specifies the reselection policy for the primary slave. This affects how the primary slave is chosen to become the active slave when failure of the active slave or recovery of the primary slave occurs. This parameter is designed to prevent flip-flopping between the primary slave and other slaves. Possible values are:

- **`always`** or **`0`** (default) — The primary slave becomes the active slave whenever it comes back up.
- **`better`** or **`1`** — The primary slave becomes the active slave when it comes back up, if the speed and duplex of the primary slave is better than the speed and duplex of the current active slave.
- **`failure`** or **`2`** — The primary slave becomes the active slave only if the current active slave fails and the primary slave is up.

The **`primary_reselect`** setting is ignored in two cases:

- If no slaves are active, the first slave to recover is made the active slave.
- When initially enslaved, the primary slave is always made the active slave.

Changing the **`primary_reselect`** policy via `sysfs` will cause an immediate selection of the best active slave according to the new policy. This may or may not result in a change of the active slave, depending upon the circumstances

`resend_igmp=`*`range`*

Specifies the number of IGMP membership reports to be issued after a failover event. One membership report is issued immediately after the failover, subsequent packets are sent in each 200ms interval.

The valid range is **`0`** to **`255`**; the default value is **`1`**. A value of **`0`** prevents the IGMP membership report from being issued in response to the failover event.

This option is useful for bonding modes **`balance-rr`** (mode 0), **`active-backup`** (mode 1), **`balance-tlb`** (mode 5) and **`balance-alb`** (mode 6), in which a failover can switch the IGMP traffic from one slave to another. Therefore a fresh IGMP report must be issued to cause the switch to forward the incoming IGMP traffic over the newly selected slave.

`updelay=`*`time_in_milliseconds`*

Specifies (in milliseconds) how long to wait before enabling a link. The value must be a multiple of the value specified in the **`miimon`** parameter. The value is set to **`0`** by default, which disables it.

`use_carrier=`*`number`*

Specifies whether or not **`miimon`** should use MII/ETHTOOL ioctls or `netif_carrier_ok()` to determine the link state. The `netif_carrier_ok()` function relies on the device driver to maintain its state with **`netif_carrier_`*`on/off`***; most device drivers support this function.

The MII/ETHTOOL ioctls tools utilize a deprecated calling sequence within the kernel. However, this is still configurable in case your device driver does not support **`netif_carrier_`*`on/off`***.

Valid values are:

- **`1`** — Default setting. Enables the use of `netif_carrier_ok()`.
- **`0`** — Enables the use of MII/ETHTOOL ioctls.

Note

If the bonding interface insists that the link is up when it should not be, it is possible that your network device driver does not support **`netif_carrier_`*`on/off`***.

`xmit_hash_policy=`*`value`*

Selects the transmit hash policy used for slave selection in **`balance-xor`** and **`802.3ad`** modes. Possible values are:

- **`0`** or **`layer2`** — Default setting. This parameter uses the XOR of hardware MAC addresses to generate the hash. The formula used is:

  ```
  (source_MAC_address XOR destination_MAC) MODULO slave_count
  ```

  This algorithm will place all traffic to a particular network peer on the same slave, and is 802.3ad compliant.

- **`1`** or **`layer3+4`** — Uses upper layer protocol information (when available) to generate the hash. This allows for traffic to a particular network peer to span multiple slaves, although a single connection will not span multiple slaves.

  The formula for unfragmented TCP and UDP packets used is:

  ```
  ((source_port XOR dest_port) XOR
    ((source_IP XOR dest_IP) AND 0xffff)
      MODULO slave_count
  ```

  For fragmented TCP or UDP packets and all other `IP` protocol traffic, the source and destination port information is omitted. For non-`IP` traffic, the formula is the same as the **`layer2`** transmit hash policy.

  This policy intends to mimic the behavior of certain switches; particularly, Cisco switches with PFC2 as well as some Foundry and IBM products.

  The algorithm used by this policy is not 802.3ad compliant.

- **`2`** or **`layer2+3`** — Uses a combination of layer2 and layer3 protocol information to generate the hash.

  Uses XOR of hardware MAC addresses and `IP` addresses to generate the hash. The formula is:

  ```
  (((source_IP XOR dest_IP) AND 0xffff) XOR
    ( source_MAC XOR destination_MAC ))
      MODULO slave_count
  ```

  This algorithm will place all traffic to a particular network peer on the same slave. For non-`IP` traffic, the formula is the same as for the layer2 transmit hash policy.

  This policy is intended to provide a more balanced distribution of traffic than layer2 alone, especially in environments where a layer3 gateway device is required to reach most destinations.

  This algorithm is 802.3ad compliant.

4.5. Using the NetworkManager Command Line Tool, nmcli

To create a bond, named *`mybond0`*, issue a command as follows:

```
~]$ nmcli con add type bond con-name mybond0 ifname mybond0 mode active-backup
Connection 'mybond0' (9301ff97-abbc-4432-aad1-246d7faea7fb) successfully added.
```

To add a slave interface, issue a command in the following form:

```
~]$ nmcli con add type bond-slave ifname ens7 master mybond0
```

To add additional slaves, repeat the previous command with a new interface, for example:

```
~]$ nmcli con add type bond-slave ifname ens3 master mybond0
Connection 'bond-slave-ens3-1' (50c59350-1531-45f4-ba04-33431c16e386) successfully added.
```

Note that as no **`con-name`** was given for the slaves, the name was derived from the interface name by prepending the type. At time of writing, **nmcli** only supports Ethernet slaves.

In order to bring up a bond, the slaves must be brought up first as follows:

```
~]$ nmcli con up bond-slave-ens7
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/
ActiveConnection/14)
```

```
~]$ nmcli con up bond-slave-ens3
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/
ActiveConnection/15)
```

Now bring up the bond as follows:

```
~]$ nmcli con up bond-mybond0
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/
ActiveConnection/16)
```

See *Section 2.4, “Using the NetworkManager Command Line Tool, nmcli”* for an introduction to **nmcli**

4.6. Additional Resources

The following sources of information provide additional resources regarding network bonding.

4.6.1. Installed Documentation

- **`nmcli(1)`** man page — Describes **NetworkManager**'s command-line tool.
- **`nmcli-examples(5)`** man page — Gives examples of **nmcli** commands.
- **`nm-settings(5)`** man page — Description of settings and parameters of **NetworkManager** connections.

4.6.2. Online Documentation

Fedora 24 System Administrator's Guide[2]

Explains the use of kernel module capabilities.

[2] https://docs.fedoraproject.org/sysadmin-guide

Configure Network Teaming

Warning

The use of direct cable connections without network switches is not supported for teaming. The failover mechanisms described here will not work as expected without the presence of network switches.

Note

The active-backup, balance-tlb and balance-alb modes do not require any specific configuration of the switch. Other bonding modes require configuring the switch to aggregate the links. For example, a Cisco switch requires EtherChannel for Modes 0, 2, and 3, but for Mode 4 LACP and EtherChannel are required. See the documentation supplied with your switch and see *https://www.kernel.org/doc/Documentation/networking/bonding.txt*.

Unlike bonding, if a team interface is configured to use link aggregation but LACP is not correctly configured and working on the network switch, no routable connection will be formed.

5.1. Understanding Network Teaming

The combining or aggregating together of network links in order to provide a logical link with higher throughput, or to provide redundancy, is known by many names such as “channel bonding”, “Ethernet bonding”, “port trunking”, “channel teaming”, “NIC teaming”, “link aggregation”, and so on. This concept as originally implemented in the Linux kernel is widely referred to as “bonding”. The term Network Teaming has been chosen to refer to this new implementation of the concept. The existing bonding driver is unaffected, Network Teaming is offered as an alternative and does not replace bonding in Fedora.

Network Teaming, or Team, is designed to implement the concept in a different way by providing a small kernel driver to implement the fast handling of packet flows, and various user-space applications to do everything else in user space. The driver has an *Application Programming Interface* (API), referred to as “Team Netlink API”, which implements Netlink communications. User-space applications can use this API to communicate with the driver. A library, referred to as “lib”, has been provided to do user space wrapping of Team Netlink communications and RT Netlink messages. An application daemon, `teamd`, which uses Libteam lib is also provided. One instance of `teamd` can control one instance of the Team driver. The daemon implements the load-balancing and active-backup logic, such as round-robin, by using additional code referred to as “runners”. By separating the code in this way, the Network Teaming implementation presents an easily extensible and scalable solution for load-balancing and redundancy requirements. For example, custom runners can be relatively easily written to implement new logic via `teamd`, and even `teamd` is optional, users can write their own application to use **libteam**.

A tool to control a running instance of `teamd` using D-bus is provided by **teamdctl**. It provides a D-Bus wrapper around the `teamd` D-Bus API. By default, `teamd` listens and communicates using Unix Domain Sockets but still monitors D-Bus. This is to ensure that `teamd` can be used in environments where D-Bus is not present or not yet loaded. For example, when booting over `teamd` links D-Bus would not yet be loaded. The **teamdctl** tool can be used during run time to read the configuration, the state of link-watchers, check and change the state of ports, add and remove ports, and to change ports between active and backup states.

Team Netlink API communicates with user-space applications using Netlink messages. The user-space library **libteam** does not directly interact with the API, but uses **libnl** or **teamnl** to interact with the driver API.

To sum up, the instances of Team driver, running in the kernel, do not get configured or controlled directly. All configuration is done with the aid of user space applications, such as the **teamd** application. The application then directs the kernel driver part accordingly.

5.2. Comparison of Network Teaming to Bonding

Table 5.1. A Comparison of Features in Bonding and Team

Feature	Bonding	Team
broadcast Tx policy	Yes	Yes
round-robin Tx policy	Yes	Yes
active-backup Tx policy	Yes	Yes
LACP (802.3ad) support	Yes (passive only)	Yes
Hash-based Tx policy	Yes	Yes
User can set hash function	No	Yes
Tx load-balancing support (TLB)	Yes	Yes
LACP hash port select	Yes	Yes
load-balancing for LACP support	No	Yes
Ethtool link monitoring	Yes	Yes
ARP link monitoring	Yes	Yes
NS/NA (IPv6) link monitoring	No	Yes
ports up/down delays	Yes	Yes
port priorities and stickiness (“primary” option enhancement)	No	Yes
separate per-port link monitoring setup	No	Yes
multiple link monitoring setup	Limited	Yes
lockless Tx/Rx path	Yes (RCU)	Yes (RCU)
VLAN support	Yes	Yes
user-space runtime control	Limited	Full
Logic in user-space	No	Yes
Extensibility	Hard	Easy
Modular design	No	Yes
Performance overhead	Low	Very Low
D-Bus interface	No	Yes

Feature	Bonding	Team
multiple device stacking	Yes	Yes
zero config using LLDP	No	(in planning)
NetworkManager support	Yes	Yes

5.3. Understanding the Default Behavior of Master and Slave Interfaces

When controlling teamed port interfaces using the `NetworkManager` daemon, and especially when fault finding, keep the following in mind:

1. Starting the master interface does not automatically start the port interfaces.
2. Starting a port interface always starts the master interface.
3. Stopping the master interface also stops the port interfaces.
4. A master without ports can start static `IP` connections.
5. A master without ports waits for ports when starting `DHCP` connections.
6. A master with a `DHCP` connection waiting for ports completes when a port with a carrier is added.
7. A master with a `DHCP` connection waiting for ports continues waiting when a port without a carrier is added.

Warning

The use of direct cable connections without network switches is not supported for teaming. The failover mechanisms described here will not work as expected without the presence of network switches.

5.4. Understanding the Network Teaming Daemon and the "Runners"

The Team daemon, `teamd`, uses **libteam** to control one instance of the team driver. This instance of the team driver adds instances of a hardware device driver to form a “team” of network links. The team driver presents a network interface, team0 for example, to the other parts of the kernel. The interfaces created by instances of the team driver are given names such as team0, team1, and so forth in the documentation. This is for ease of understanding and other names can be used. The logic common to all methods of teaming is implemented by `teamd`; those functions that are unique to the different load sharing and backup methods, such as round-robin, are implemented by separate units of code referred to as “runners”. Because words such as “module” and “mode” already have specific meanings in relation to the kernel, the word “runner” was chosen to refer to these units of code. The user specifies the runner in the JSON format configuration file and the code is then compiled into an instance of `teamd` when the instance is created. A runner is not a plug-in because the code for a runner is compiled into an instance of `teamd` as it is being created. Code could be created as a plug-in for `teamd` should the need arise.

The following runners are available at time of writing.

- broadcast (data is transmitted over all ports)

- round-robin (data is transmitted over all ports in turn)
- active-backup (one port or link is used while others are kept as a backup)
- loadbalance (with active Tx load balancing and BPF-based Tx port selectors)
- lacp (implements the 802.3ad Link Aggregation Control Protocol)

In addition, the following link-watchers are available:

- **ethtool** (Libteam lib uses **ethtool** to watch for link state changes). This is the default if no other link-watcher is specified in the configuration file.
- **arp_ping** (The **arp_ping** utility is used to monitor the presence of a far-end hardware address using ARP packets.)
- **nsna_ping** (Neighbor Advertisements and Neighbor Solicitation from the `IPv6` Neighbor Discovery protocol are used to monitor the presence of a neighbor's interface)

There are no restrictions in the code to prevent a particular link-watcher from being used with a particular runner, however when using the **lacp** runner, **ethtool** is the only recommended link-watcher.

5.5. Install the Network Teaming Daemon

The networking teaming daemon, `teamd`, is not installed by default. To install `teamd`, issue the following command as `root`:

```
~]# dnf install teamd
```

5.6. Converting a Bond to a Team

It is possible to convert existing bonding configuration files to team configuration files using the **bond2team** tool. It can convert bond configuration files in **`ifcfg`** format to team configuration files in either **`ifcfg`** or JSON format. Note that firewall rules, alias interfaces, and anything that might be tied to the original interface name can break after the renaming because the tool will only change the **`ifcfg`** file, nothing else.

To see some examples of the command format, issue the following command:

```
~]$ bond2team --examples
```

New files will be created in a directory whose name starts with **`/tmp/bond2team.XXXXXX/`**, where XXXXXX is a random string. After creating the new configuration files, move the old bonding files to a backup folder and then move the new files to the **`/etc/sysconfig/network-scripts/`** directory.

Example 5.1. Convert a Bond to a Team

To convert a current **`bond0`** configuration to team **`ifcfg`**, issue a command as `root`:

```
~]# /usr/bin/bond2team --master bond0
```

Note that this will retain the name **`bond0`**. To use a new name to save the configuration, use the **`--rename`** as follows:

```
~]# /usr/bin/bond2team --master bond0 --rename team0
```

add the **`--json`** option to output JSON format files instead of **`ifcfg`** files. See the **`teamd.conf(5)`** man page for examples of JSON format.

Example 5.2. Convert a Bond to a Team and Specify the File Path

To convert a current **`bond0`** configuration to team **`ifcfg`**, and to manually specify the path to the **`ifcfg`** file, issue a command as `root`:

```
~]# /usr/bin/bond2team --master bond0 --configdir /path/to/ifcfg-file
```

add the **`--json`** option to output JSON format files instead of **`ifcfg`** files.

Example 5.3. Create a Team Configuration Using Bond2team

It is also possible to create a team configuration by supplying the **bond2team** tool with a list of bonding parameters. For example:

```
~]# /usr/bin/bond2team --bonding_opts "mode=1 miimon=500"
```

Ports can also be supplied on the command line as follows:

```
~]# /usr/bin/bond2team --bonding_opts "mode=1 miimon=500 primary=eth1 \
  primary_reselect-0" --port eth1 --port eth2 --port eth3 --port eth4
```

See the **`bond2team(1)`** man page for further details. For an explanation of bonding parameters, see *Section 4.4, “Using Channel Bonding”*

5.7. Selecting Interfaces to Use as Ports for a Network Team

To view the available interfaces, issue the following command:

```
~]$ ip link show
1: lo:  <LOOPBACK,UP,LOWER_UP > mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: em1:  <BROADCAST,MULTICAST,UP,LOWER_UP > mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
 qlen 1000
    link/ether 52:54:00:6a:02:8a brd ff:ff:ff:ff:ff:ff
3: em2:  <BROADCAST,MULTICAST,UP,LOWER_UP > mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
 qlen 1000
link/ether 52:54:00:9b:6d:2a brd ff:ff:ff:ff:ff:ff
```

From the available interfaces, determine which are suitable for adding to your network team and then proceed to *Section 5.8, “Selecting Network Team Configuration Methods”*

Note

The Team developers prefer the term “port” rather than “slave”, however **NetworkManager** uses the term “team-slave” to refer to interfaces that make up a team.

5.8. Selecting Network Team Configuration Methods

To configure a network team using a graphical user interface, see *Section 5.9, “Creating a Network Team Using a GUI”*

To create a network team using the Team daemon, `teamd`, proceed to *Section 5.10.1, “Creating a Network Team Using teamd”*.

To create a network team using configuration files, proceed to *Section 5.10.2, “Creating a Network Team Using ifcfg Files”*.

To create a network team using the command-line tool, **nmcli**, proceed to *Section 5.13, “Configure Network Teaming Using nmcli”*.

5.9. Creating a Network Team Using a GUI

5.9.1. Establishing a Team Connection

You can use the GNOME **control-center** utility to direct **NetworkManager** to create a team from two or more Wired or InfiniBand connections. It is not necessary to create the connections to be teamed first. They can be configured as part of the process to configure the team. You must have the MAC addresses of the interfaces available in order to complete the configuration process.

Procedure 5.1. Adding a New Team Connection

You can configure a team connection by opening the **Network** window, clicking the plus symbol, and selecting **Team** from the list.

1. Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.

2. Click the plus symbol to open the selection list. Select **Team**. The **Editing Team Connection *1*** window appears.

3. On the **Team** tab, click **Add** and select the type of interface you want to use with the team connection. Click the **Create** button. Note that the dialog to select the port type only comes up when you create the first port; after that, it will automatically use that same type for all further ports.

4. The **Editing team0 port 1** window appears. Fill in the MAC address of the first interface to be added to the team.

5. If custom port settings are to be applied, click on the **Team Port** tab and enter a JSON configuration string or import it from a file.

6. Click the **Save** button.

7. The name of the teamed port appears in the **Teamed connections** window. Click the **Add** button to add further port connections.

8. Review and confirm the settings and then click the **Save** button.

9. Edit the team-specific settings by referring to *Section 5.9.1.1, “Configuring the Team Tab”* below.

Procedure 5.2. Editing an Existing Team Connection

Follow these steps to edit an existing bond connection.

1. Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.

2. Select the connection you want to edit and click the **Options** button.

3. Select the **General** tab.

4. Configure the connection name, auto-connect behavior, and availability settings.

   Five settings in the **Editing** dialog are common to all connection types, see the **General** tab:

   - **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.

   - **Automatically connect to this network when it is available** — Select this box if you want **NetworkManager** to auto-connect to this connection when it is available. See *Section 2.2.3, “Connecting to a Network Automatically”* for more information.

   - **All users may connect to this network** — Select this box to create a connection available to all users on the system. Changing this setting may require root privileges. See *Section 2.2.4, “System-wide and Private Connection Profiles”* for details.

   - **Automatically connect to VPN when using this connection** — Select this box if you want **NetworkManager** to auto-connect to a VPN connection when it is available. Select the VPN from the drop-down menu.

   - **Firewall Zone** — Select the firewall zone from the drop-down menu.

5. Edit the team-specific settings by referring to *Section 5.9.1.1, “Configuring the Team Tab”* below.

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your team connection, click the **Save** button to save your customized configuration. If the profile was in use while being edited, power cycle the connection to make **NetworkManager** apply the changes. If the profile is OFF, set it to ON. See *Section 2.2.1, “Connecting to a Network Using a GUI”* for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network** window and clicking **Options** to return to the **Editing** dialog.

Then, to configure:

- `IPv4` settings for the connection, click the **IPv4 Settings** tab and proceed to *Section 2.2.10.4, “Configuring IPv4 Settings”*; or,

- IPv6 settings for the connection, click the **IPv6 Settings** tab and proceed to *Section 2.2.10.5, “Configuring IPv6 Settings”*.

5.9.1.1. Configuring the Team Tab

If you have already added a new team connection you can enter a custom JSON configuration string in the text box or import a configuration file. Click **Save** to apply the JSON configuration to the team interface.

For examples of JSON strings, see *Section 5.12, “Configure teamd Runners”*

See *Procedure 5.1, “Adding a New Team Connection”* for instructions on how to add a new team.

5.10. Configure a Network Team Using the Command Line

5.10.1. Creating a Network Team Using teamd

To create a network team, a JSON format configuration file is required for the virtual interface that will serve as the interface to the team of ports or links. A quick way is to copy the example configuration files and then edit them using an editor running with `root` privileges. To list the available example configurations, enter the following command:

```
~]$ ls /usr/share/doc/teamd-*/example_configs/
activebackup_arp_ping_1.conf  activebackup_multi_lw_1.conf   loadbalance_2.conf
activebackup_arp_ping_2.conf  activebackup_nsna_ping_1.conf  loadbalance_3.conf
activebackup_ethtool_1.conf   broadcast.conf                 random.conf
activebackup_ethtool_2.conf   lacp_1.conf                    roundrobin_2.conf
activebackup_ethtool_3.conf   loadbalance_1.conf             roundrobin.conf
```

To view one of the included files, such as **`activebackup_ethtool_1.conf`**, enter the following command:

```
~]$ cat /usr/share/doc/teamd-*/example_configs/activebackup_ethtool_1.conf
{
 "device": "team0",
 "runner": {"name": "activebackup"},
 "link_watch": {"name": "ethtool"},
 "ports": {
  "eth1": {
   "prio": -10,
   "sticky": true
  },
  "eth2": {
   "prio": 100
  }
 }
}
```

Create a working configurations directory to store `teamd` configuration files. For example, as normal user, enter a command with the following format:

```
~]$ mkdir ~/teamd_working_configs
```

Copy the file you have chosen to your working directory and edit it as necessary. As an example, you could use a command with the following format:

```
~]$ cp /usr/share/doc/teamd-*/example_configs/activebackup_ethtool_1.conf \
 ~/teamd_working_configs/activebackup_ethtool_1.conf
```

To edit the file to suit your environment, for example to change the interfaces to be used as ports for the network team, open the file for editing as follows:

```
~]$ vi ~/teamd_working_configs/activebackup_ethtool_1.conf
```

Make any necessary changes and save the file. See the **`vi(1)`** man page for help on using the **vi** editor or use your preferred editor.

Note that it is essential that the interfaces to be used as ports within the team must not be active, that is to say, they must be “down”, when adding them into a team device. To check their status, issue the following command:

```
~]$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: em1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
 qlen 1000
    link/ether 52:54:00:d5:f7:d4 brd ff:ff:ff:ff:ff:ff
3: em2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
 qlen 1000
  link/ether 52:54:00:d8:04:70 brd ff:ff:ff:ff:ff:ff
```

In this example we see that both the interfaces we plan to use are “UP”.

To take down an interface, issue a command as `root` in the following format:

```
~]# ip link set down em1
```

Repeat for each interface as necessary.

To create a team interface based on the configuration file, as `root` user, change to the working configurations directory (*`teamd_working_configs`* in this example):

```
~]# cd /home/userteamd_working_configs
```

Then issue a command in the following format:

```
~]# teamd -g -f activebackup_ethtool_1.conf -d
Using team device "team0".
Using PID file "/var/run/teamd/team0.pid"
Using config file "/home/user/teamd_working_configs/activebackup_ethtool_1.conf"
```

The **`-g`** option is for debug messages, **`-f`** option is to specify the configuration file to load, and the **`-d`** option is to make the process run as a daemon after startup. See the **`teamd(8)`** man page for other options.

To check the status of the team, issue the following command as `root`:

```
~]# teamdctl team0 state
setup:
  runner: activebackup
ports:
  em1
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
  em2
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
runner:
  active port: em1
```

To apply an address to the network team interface, team0, issue a command as `root` in the following format:

```
~]# ip addr add 192.168.23.2/24 dev team0
```

To check the IP address of a team interface, issue a command as follows:

```
~]$ ip addr show team0
4: team0:  <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 16:38:57:60:20:6f brd ff:ff:ff:ff:ff:ff
    inet 192.168.23.2/24 scope global team0
       valid_lft forever preferred_lft forever
    inet6 2620:52:0:221d:1438:57ff:fe60:206f/64 scope global dynamic
       valid_lft 2591880sec preferred_lft 604680sec
    inet6 fe80::1438:57ff:fe60:206f/64 scope link
       valid_lft forever preferred_lft forever
```

To activate the team interface, or to bring it “up”, issue a command as `root` in the following format:

```
~]# ip link set dev team0 up
```

To temporarily deactivate the team interface, or to take it “down”, issue a command as `root` in the following format:

```
~]# ip link set dev team0 down
```

To terminate, or kill, an instance of the team daemon, as `root` user, issue a command in the following format:

```
~]# teamd -t team0 -k
```

The **`-k`** option is to specify that the instance of the daemon associated with the device team0 is to be killed. See the **`teamd(8)`** man page for other options.

For help on command-line options for `teamd`, issue the following command:

```
~]$ teamd -h
```

In addition, see the **`teamd(8)`** man page.

5.10.2. Creating a Network Team Using ifcfg Files

To create a networking team using **`ifcfg`** files, create a file in the **`/etc/sysconfig/network-scripts/`** directory as follows:

```
DEVICE=team0
DEVICETYPE=Team
ONBOOT=yes
BOOTPROTO=none
IPADDR=192.168.11.1
PREFIX=24
TEAM_CONFIG='{"runner": {"name": "activebackup"}, "link_watch": {"name": "ethtool"}}'
```

This creates the interface to the team, in other words, this is the master.

To create a port to be a member of team0, create one or more files in the **`/etc/sysconfig/network-scripts/`** directory as follows:

```
DEVICE=eth1
HWADDR=D4:85:64:01:46:9E
DEVICETYPE=TeamPort
ONBOOT=yes
```

```
TEAM_MASTER=team0
TEAM_PORT_CONFIG='{"prio": 100}'
```

Add additional port interfaces similar to the above as required, changing the DEVICE and HWADDR field to match the ports (the network devices) being added. If port priority is not specified by **`prio`** it defaults to **`0`**; it accepts negative and positive values in the range **`-32,767`** to **`+32,767`**.

Specifying the hardware or MAC address using the **`HWADDR`** directive will influence the device naming procedure. This is explained in *Chapter 8, Consistent Network Device Naming*.

To bring up the network team, issue the following command as `root`:

```
~]# ifup team0
```

To view the network team, issue the following command:

```
~]$ ip link show
```

5.10.3. Add a Port to a Network Team Using iputils

To add a port em1 to a network team team0, using the **ip** utility, issue the following commands as `root`:

```
~]# ip link set dev em1 down
~]# ip link set dev em1 master team0
```

Add additional ports as required. Team driver will bring ports up automatically.

5.10.4. Listing the ports of a Team Using teamnl

To view or list the ports in a network team, using the **teamnl** utility, issue the following command as `root`:

```
~]# teamnl team0 ports
em2: up 100 fullduplex
em1: up 100 fullduplex
```

5.10.5. Configuring Options of a Team Using teamnl

To view or list all currently available options, using the **teamnl** utility, issue the following command as `root`:

```
~]# teamnl team0 options
```

To configure a team to use active backup mode, issue the following command as `root`:

```
~]# teamnl team0 setoption mode activebackup
```

5.10.6. Add an Address to a Network Team Using iputils

To add an address to a team team0, using the **ip** utility, issue the following command as `root`:

```
~]# ip addr add 192.168.252.2/24 dev team0
```

5.10.7. Bring up an Interface to a Network Team Using iputils

To activate or “bring up” an interface to a network team, team0, using the **ip** utility, issue the following command as `root`:

```
~]# ip link set team0 up
```

5.10.8. Viewing the Active Port Options of a Team Using teamnl

To view or list the **`activeport`** option in a network team, using the **teamnl** utility, issue the following command as `root`:

```
~]# teamnl team0 getoption activeport
0
```

5.10.9. Setting the Active Port Options of a Team Using teamnl

To set the **`activeport`** option in a network team, using the **teamnl** utility, issue the following command as `root`:

```
~]# teamnl team0 setoption activeport 5
```

To check the change in team port options, issue the following command as `root`:

```
~]# teamnl team0 getoption activeport
5
```

5.11. Controlling teamd with teamdctl

In order to query a running instance of `teamd` for statistics or configuration information, or to make changes, the control tool **teamdctl** is used.

To view the current team state of a team team0, enter the following command as `root`:

```
~]# teamdctl team0 state view
```

For a more verbose output:

```
~]# teamdctl team0 state view -v
```

For a complete state dump in JSON format (useful for machine processing) of team0, use the following command:

```
~]# teamdctl team0 state dump
```

For a configuration dump in JSON format of team0, use the following command:

```
~]# teamdctl team0 config dump
```

To view the configuration of a port em1, that is part of a team team0, enter the following command:

```
~]# teamdctl team0 port config dump em1
```

5.11.1. Add a Port to a Network Team

To add a port em1 to a network team team0, issue the following command as `root`:

```
~]# teamdctl team0 port add em1
```

5.11.2. Remove a Port From a Network Team

To remove an interface em1 from a network team team0, issue the following command as `root`:

```
~]# teamdctl team0 port remove em1
```

5.11.3. Apply a Configuration to a Port in a Network Team

To apply a JSON format configuration to a port em1 in a network team team0, issue a command as `root` in the following format:

```
~]# teamdctl team0 port config update em1 JSON-config-string
```

Where *`JSON-config-string`* is the configuration as a string of text in JSON format. This will update the configuration of the port using the JSON format string supplied. An example of a valid JSON string for configuring a port would be the following:

```
{
  "prio": -10,
  "sticky": true
}
```

Use single quotes around the JSON configuration string and omit the line breaks.

Note that the old configuration will be overwritten and that any options omitted will be reset to the default values. See the **`teamdctl(8)`** man page for more team daemon control tool command examples.

5.11.4. View the Configuration of a Port in a Network Team

To copy the configuration of a port em1 in a network team team0, issue the following command as `root`:

```
~]# teamdctl team0 port config dump em1
```

This will dump the JSON format configuration of the port to standard output.

5.12. Configure teamd Runners

Runners are units of code which are compiled into the Team daemon when an instance of the daemon is created. For an introduction to the `teamd` runners, see *Section 5.4, “Understanding the Network Teaming Daemon and the "Runners"”*.

5.12.1. Configure the broadcast Runner

To configure the broadcast runner, using an editor as `root`, add the following to the team JSON format configuration file:

```
{
 "device": "team0",
 "runner": {"name": "broadcast"},
 "ports": {"em1": {}, "em2": {}}
}
```

Please see the **`teamd.conf(5)`** man page for more information.

5.12.2. Configure the random Runner

The random runner behaves similarly to the roundrobin runner.

To configure the random runner, using an editor as `root`, add the following to the team JSON format configuration file:

```
{
 "device": "team0",
 "runner": {"name": "random"},
 "ports": {"em1": {}, "em2": {}}
}
```

Please see the **`teamd.conf(5)`** man page for more information.

5.12.3. Configure the roundrobin Runner

To configure the roundrobin runner, using an editor as `root`, add the following to the team JSON format configuration file:

```
{
 "device": "team0",
 "runner": {"name": "roundrobin"},
 "ports": {"em1": {}, "em2": {}}
}
```

A very basic configuration for roundrobin.

Please see the **`teamd.conf(5)`** man page for more information.

5.12.4. Configure the activebackup Runner

The active backup runner can use all of the link-watchers to determine the status of links in a team. Any one of the following examples can be added to the team JSON format configuration file:

```
{
   "device": "team0",
   "runner": {
      "name": "activebackup"
   },
   "link_watch": {
      "name": "ethtool"
   },
   "ports": {
      "em1": {
         "prio": -10,
         "sticky": true
      },
      "em2": {
         "prio": 100
      }
   }
}
```

This example configuration uses the active-backup runner with **ethtool** as the link watcher. Port em2 has higher priority. The sticky flag ensures that if em1 becomes active, it stays active as long as the link remains up.

```
{
   "device": "team0",
   "runner": {
      "name": "activebackup"
   },
   "link_watch": {
      "name": "ethtool"
   },
   "ports": {
      "em1": {
         "prio": -10,
         "sticky": true,
         "queue_id": 4
      },
      "em2": {
         "prio": 100
      }
   }
}
```

This example configuration adds a queue ID of **4**. It uses active-backup runner with **ethtool** as the link watcher. Port em2 has higher priority. But the sticky flag ensures that if em1 becomes active, it will stay active as long as the link remains up.

To configure the activebackup runner using **ethtool** as the link watcher and applying a delay, using an editor as `root`, add the following to the team JSON format configuration file:

```
{
   "device": "team0",
   "runner": {
      "name": "activebackup"
   },
   "link_watch": {
      "name": "ethtool",
      "delay_up": 2500,
      "delay_down": 1000
   },
   "ports": {
      "em1": {
         "prio": -10,
         "sticky": true
      },
      "em2": {
         "prio": 100
      }
   }
}
```

This example configuration uses the active-backup runner with **ethtool** as the link watcher. Port em2 has higher priority. But the sticky flag ensures that if em1 becomes active, it stays active while the link remains up. Link changes are not propagated to the runner immediately, but delays are applied.

Please see the **`teamd.conf(5)`** man page for more information.

5.12.5. Configure the loadbalance Runner

This runner can be used for two types of load balancing, active and passive. In active mode, constant re-balancing of traffic is done by using statistics of recent traffic to share out traffic as evenly as possible. In static mode, streams of traffic are distributed randomly across the available links. This has a speed advantage due to lower processing overhead. In high volume traffic applications this is often preferred as traffic usually consists of multiple stream which will be distributed randomly between the available links, in this way load sharing is accomplished without intervention by `teamd`.

To configure the loadbalance runner for passive transmit (Tx) load balancing, using an editor as `root`, add the following to the team JSON format configuration file:

```
{
 "device": "team0",
 "runner": {
   "name": "loadbalance",
   "tx_hash": ["eth", "ipv4", "ipv6"]
 },
 "ports": {"em1": {}, "em2": {}}
}
```

Configuration for hash-based passive transmit (Tx) load balancing.

To configure the loadbalance runner for active transmit (Tx) load balancing, using an editor as `root`, add the following to the team JSON format configuration file:

```
{
   "device": "team0",
   "runner": {
     "name": "loadbalance",
     "tx_hash": ["eth", "ipv4", "ipv6"],
     "tx_balancer": {
       "name": "basic"
     }
   },
   "ports": {"em1": {}, "em2": {}}
}
```

Configuration for active transmit (Tx) load balancing using basic load balancer.

Please see the **`teamd.conf(5)`** man page for more information.

5.12.6. Configure the LACP (802.3ad) Runner

To configure the LACP runner using **ethtool** as a link watcher, using an editor as `root`, add the following to the team JSON format configuration file:

```
{
   "device": "team0",
   "runner": {
       "name": "lacp",
       "active": true,
       "fast_rate": true,
       "tx_hash": ["eth", "ipv4", "ipv6"]
   },
     "link_watch": {"name": "ethtool"},
     "ports": {"em1": {}, "em2": {}}
}
```

Configuration for connection to a *link aggregation control protocol* (LACP) capable counterpart. The LACP runner should use **ethtool** to monitor the status of a link. It does not make sense to use any other link monitoring method besides the **ethtool** because, for example in the case of **arp_ping**, the link would never come up. The reason is that the link has to be established first and only after that can packets, ARP included, go through. Using **ethtool** prevents this because it monitors each link layer individually.

Active load balancing is possible with this runner in the same way as it is done for the loadbalance runner. To enable active transmit (Tx) load balancing, add the following section:

```
"tx_balancer": {
       "name": "basic"
}
```

Please see the **`teamd.conf(5)`** man page for more information.

5.12.7. Configure Monitoring of the Link State

The following methods of link state monitoring are available. To implement one of the methods, add the JSON format string to the team JSON format configuration file using an editor running with `root` privileges.

5.12.7.1. Configure Ethtool for link-state Monitoring

To add or edit an existing delay, in milliseconds, between the link coming up and the runner being notified about it, add or edit a section as follows:

```
"link_watch": {
       "name": "ethtool",
       "delay_up": 2500
}
```

To add or edit an existing delay, in milliseconds, between the link going down and the runner being notified about it, add or edit a section as follows:

```
"link_watch": {
       "name": "ethtool",
       "delay_down": 1000
}
```

5.12.7.2. Configure ARP Ping for Link-state Monitoring

The team daemon `teamd` sends an ARP REQUEST to an address at the remote end of the link in order to determine if the link is up. The method used is the same as the **arping** utility but it does not use that utility.

Prepare a file containing the new configuration in JSON format similar to the following example:

```
{
       "device": "team0",
       "runner": {"name": "activebackup"},
       "link_watch":{
           "name": "arp_ping",
           "interval": 100,
           "missed_max": 30,
           "source_host": "192.168.23.2",
           "target_host": "192.168.23.1"
       },
         "ports": {
             "em1": {
               "prio": -10,
               "sticky": true
             },
             "em2": {
               "prio": 100
             }
         }
}
```

This configuration uses **arp_ping** as the link watcher. The **`missed_max`** option is a limit value of the maximum allowed number of missed replies (ARP replies for example). It should be chosen in conjunction with the **`interval`** option in order to determine the total time before a link is reported as down.

To load a new configuration for a team port em2, from a file containing a JSON configuration, issue the following command as `root`:

```
~]# port config update em2 JSON-config-file
```

Note that the old configuration will be overwritten and that any options omitted will be reset to the default values. See the **`teamdctl(8)`** man page for more team daemon control tool command examples.

5.12.7.3. Configure IPv6 NA/NS for Link-state Monitoring

```
{
  "device": "team0",
    "runner": {"name": "activebackup"},
    "link_watch": {
    "name": "nsna_ping",
    "interval": 200,
    "missed_max": 15,
    "target_host": "fe80::210:18ff:feaa:bbcc"
    },
      "ports": {
        "em1": {
          "prio": -10,
          "sticky": true
        },
        "em2": {
          "prio": 100
        }
      }
}
```

To configure the interval between sending NS/NA packets, add or edit a section as follows:

```
"link_watch": {
    "name": "nsna_ping",
    "interval": 200
}
```

Value is positive number in milliseconds. It should be chosen in conjunction with the **`missed_max`** option in order to determine the total time before a link is reported as down.

To configure the maximum number of missed NS/NA reply packets to allow before reporting the link as down, add or edit a section as follows:

```
"link_watch": {
    "name": "nsna_ping",
    "missed_max": 15
}
```

Maximum number of missed NS/NA reply packets. If this number is exceeded, the link is reported as down. The **`missed_max`** option is a limit value of the maximum allowed number of missed replies (ARP replies for example). It should be chosen in conjunction with the **`interval`** option in order to determine the total time before a link is reported as down.

To configure the host name that is resolved to the `IPv6` address target address for the NS/NA packets, add or edit a section as follows:

```
"link_watch": {
    "name": "nsna_ping",
    "target_host": "MyStorage"
}
```

The “target_host” option contains the host name to be converted to an `IPv6` address which will be used as the target address for the NS/NA packets. An `IPv6` address can be used in place of a host name.

Please see the **`teamd.conf(5)`** man page for more information.

5.12.8. Configure Port Selection Override

The physical port which transmits a frame is normally selected by the kernel part of the team driver, and is not relevant to the user or system administrator. The output port is selected using the policies of the selected team mode (`teamd` runner). On occasion however, it is helpful to direct certain classes of outgoing traffic to certain physical interfaces to implement slightly more complex policies. By default the team driver is multiqueue aware and 16 queues are created when the driver initializes (see *https://www.kernel.org/doc/Documentation/networking/multiqueue.txt* for details). If more or less queues are desired, the Netlink attribute **`tx_queues`** can be used to change this value during the team driver instance creation.

The queue ID for a port can be set by the port configuration option **`queue_id`** as follows:

```
{
  "queue_id": 3
}
```

These queue ID's can be used in conjunction with the **tc** utility to configure a multiqueue queue discipline and filters to bias certain traffic to be transmitted on certain port devices. For example, if using the above configuration and wanting to force all traffic bound to `192.168.1.100` to use eth1 in the team as its output device, issue commands as `root` in the following format:

```
~]# tc qdisc add dev team0 handle 1 root multiq
~]# tc filter add dev team0 protocol ip parent 1: prio 1 u32 match ip dst \
 192.168.1.100 action skbedit queue_mapping 3
```

This mechanism of overriding runner selection logic in order to bind traffic to a specific port can be used with all runners.

5.12.9. Configure BPF-based Tx Port Selectors

The loadbalance and LACP runners uses hashes of packets to sort network traffic flow. The hash computation mechanism is based on the *Berkeley Packet Filter* (BPF) code. The BPF code is used to generate a hash rather than make a policy decision for outgoing packets. The hash length is 8 bits giving 256 variants. This means many different *socket buffers* (SKB) can have the same hash and therefore pass traffic over the same link. The use of a short hash is a quick way to sort traffic into different streams for the purposes of load balancing across multiple links. In static mode, the hash is only used to decide out of which port the traffic should be sent. In active mode, the runner will continually reassign hashes to different ports in an attempt to reach a perfect balance.

The following fragment types or strings can be used for packet Tx hash computation:

- **`eth`** — Uses source and destination MAC addresses.
- **`vlan`** — Uses VLAN ID.
- **`ipv4`** — Uses source and destination `IPv4` addresses.
- **`ipv6`** — Uses source and destination `IPv6` addresses.
- **`ip`** — Uses source and destination `IPv4` and `IPv6` addresses.
- **`l3`** — Uses source and destination `IPv4` and `IPv6` addresses.
- **`tcp`** — Uses source and destination `TCP` ports.
- **`udp`** — Uses source and destination `UDP` ports.
- **`sctp`** — Uses source and destination `SCTP` ports.
- **`l4`** — Uses source and destination `TCP` and `UDP` and `SCTP` ports.

These strings can be used by adding a line in the following format to the load balance runner:

```
"tx_hash": ["eth", "ipv4", "ipv6"]
```

See *Section 5.12.5, “Configure the loadbalance Runner”* for an example.

5.13. Configure Network Teaming Using nmcli

To view the devices available on the system, issue the following command:

```
~]$ nmcli connection show
NAME  UUID                                  TYPE            DEVICE
eth1  0e8185a1-f0fd-4802-99fb-bedbb31c689b  802-3-ethernet  --
eth0  dfe1f57b-419d-4d1c-aaf5-245deab82487  802-3-ethernet  --
```

To create a new team interface, with name *`team-ServerA`*, issue a command as follows:

```
~]$ nmcli connection add type team ifname team-ServerA
```

```
Connection 'team-ServerA' (b954c62f-5fdd-4339-97b0-40efac734c50) successfully added.
```

NetworkManager will set its internal parameter **`connection.autoconnect`** to **`yes`** and as no `IP` address was given **`ipv4.method`** will be set to **`auto`**. **NetworkManager** will also write a configuration file to **`/etc/sysconfig/network-scripts/ifcfg-team-ServerA`** where the corresponding ONBOOT will be set to **`yes`** and BOOTPROTO will be set to **`dhcp`**.

Note that manual changes to the ifcfg file will not be noticed by **NetworkManager** until the interface is next brought up. See *Section 1.7, "Network Configuration Using sysconfig Files"* for more information on using configuration files.

To view the other values assigned, issue a command as follows:

```
~]$ nmcli con show team-ServerA
connection.id:                          team-ServerA
connection.uuid:                        b954c62f-5fdd-4339-97b0-40efac734c50
connection.interface-name:              ServerA
connection.type:                        team
connection.autoconnect:                 yes
…
ipv4.method:                            auto
[output truncated]
```

As no JSON configuration file was specified the default values apply. See the **`teamd.conf(5)`** man page for more information on the team JSON parameters and their default values. Notice that the name was derived from the interface name by prepending the type. Alternatively, specify a name with the **`con-name`** option as follows:

```
~]$ nmcli connection add type team con-name Team0 ifname ServerB
Connection 'Team0' (5f7160a1-09f6-4204-8ff0-6d96a91218a7) successfully added.
```

To view the team interfaces just configured, issue a command as follows:

```
~]$ nmcli con show
NAME            UUID                                  TYPE            DEVICE
team-ServerA    b954c62f-5fdd-4339-97b0-40efac734c50  team            ServerA
eth1            0e8185a1-f0fd-4802-99fb-bedbb31c689b  802-3-ethernet  --
eth0            dfe1f57b-419d-4d1c-aaf5-245deab82487  802-3-ethernet  --
Team0           5f7160a1-09f6-4204-8ff0-6d96a91218a7  team            ServerB
```

To change the name assigned to a team, issue a command in the following format:

```
nmcli con mod old-team-name connection.id new-team-name
```

To load a team configuration file for a team that already exists, issue a command in the following format:

```
nmcli connection modify team-name team.config JSON-config
```

You can specify the team configuration either as a JSON string or provide a file name containing the configuration. The file name can include the path. In both cases, what is stored in the *`team.config`* property is the JSON string. In the case of a JSON string, use single quotes around the string and paste the entire string to the command line.

To review the **`team.config`** property, enter a command in the following format:

```
nmcli con show team-name | grep team.config
```

To add an interface *eth0* to **Team0**, with the name *Team0-port1*, issue a command as follows:

```
~]$ nmcli con add type team-slave con-name Team0-port1 ifname eth0 master Team0
Connection 'Team0-port1' (ccd87704-c866-459e-8fe7-01b06cf1cffc) successfully added.
```

Similarly, to add another interface, *eth1*, with the name *Team0-port2*, issue a command as follows:

```
~]$ nmcli con add type team-slave con-name Team0-port2 ifname eth1 master Team0
Connection 'Team0-port2' (a89ccff8-8202-411e-8ca6-2953b7db52dd) successfully added.
```

At time of writing, **nmcli** only supports Ethernet ports.

In order to bring up a team, the ports must be brought up first as follows:

```
~]$ nmcli connection up Team0-port1
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/
ActiveConnection/2)
```

```
~]$ nmcli connection up Team0-port2
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/
ActiveConnection/3)
```

You can verify that the team interface was brought up by the activation of the ports, as follows:

```
~]$ ip link
3:  Team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT
    link/ether 52:54:00:76:6f:f0 brd ff:ff:ff:ff:ff:f
```

Alternatively, issue a command to bring up the team as follows:

```
~]$ nmcli connection up Team0
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/
ActiveConnection/4)
```

See *Section 2.4, "Using the NetworkManager Command Line Tool, nmcli"* for an introduction to **nmcli**

5.14. Additional Resources

The following sources of information provide additional resources regarding network teaming.

5.14.1. Installed Documentation

- **teamd(8)** man page — Describes the `teamd` service.
- **teamdctl(8)** man page — Describes the `teamd` control tool.
- **teamd.conf(5)** man page — Describes the `teamd` configuration file.
- **teamnl(8)** man page — Describes the `teamd` Netlink library.
- **bond2team(1)** man page — Describes a tool to convert bonding options to team.

5.14.2. Online Documentation

http://www.libteam.org

The upstream project.

http://www.w3schools.com/json/json_syntax.asp
An explanation of JSON syntax.

Configure Network Bridging

A network bridge is a link-layer device which forwards traffic between networks based on MAC addresses. It makes forwarding decisions based on a table of MAC addresses which it builds by listening to network traffic and thereby learning what hosts are connected to each network. A software bridge can be used within a Linux host in order to emulate a hardware bridge, for example in virtualization applications for sharing a NIC with one or more virtual NICs.

Note that a bridge cannot be established over Wi-Fi networks operating in *Ad-Hoc* or *Infrastructure* modes. This is due to the IEEE 802.11 standard that specifies the use of 3-address frames in Wi-Fi for the efficient use of airtime. A system configured to be an *access point* (AP) running the `hostapd` can support the necessary 4-address frames.

6.1. Configure Network Bridging Using a GUI

When starting a bridge interface, **NetworkManager** waits for at least one port to enter the “forwarding” state before beginning any network-dependent `IP` configuration such as `DHCP` or `IPv6` autoconfiguration. Static `IP` addressing is allowed to proceed before any slaves or ports are connected or begin forwarding packets.

6.1.1. Establishing a Bridge Connection

Procedure 6.1. Adding a New Bridge Connection

You can configure a new Bridge connection by opening the **Network** window and selecting the plus symbol below the menu.

1. To use the graphical **Network** settings tool, press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.
2. Select the plus symbol below the menu. The **Add Network Connection** window appears.
3. Select the **Bridge** menu entry. The **Editing Bridge connection *1*** window appears.

Procedure 6.2. Editing an Existing Bridge Connection

You can configure an existing bridge connection by opening the **Network** window and selecting the name of the connection from the list. Then click the **Edit** button.

1. Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.
2. Select the **Bridge** connection you want to edit from the left hand menu.
3. Click the **Options** button.

Configuring the Connection Name, Auto-Connect Behavior, and Availability Settings

Five settings in the **Editing** dialog are common to all connection types, see the **General** tab:

- **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.
- **Automatically connect to this network when it is available** — Select this box if you want **NetworkManager** to auto-connect to this connection when it is available. See *Section 2.2.3, “Connecting to a Network Automatically”* for more information.

- **All users may connect to this network** — Select this box to create a connection available to all users on the system. Changing this setting may require root privileges. See *Section 2.2.4, “System-wide and Private Connection Profiles”* for details.
- **Automatically connect to VPN when using this connection** — Select this box if you want **NetworkManager** to auto-connect to a VPN connection when it is available. Select the VPN from the dropdown menu.
- **Firewall Zone** — Select the Firewall Zone from the dropdown menu.

6.1.1.1. Configuring the Bridge Tab

Interface name
: The name of the interface to the bridge.

Bridged connections
: One or more slave interfaces.

Aging time
: The time, in seconds, a MAC address is kept in the MAC address forwarding database.

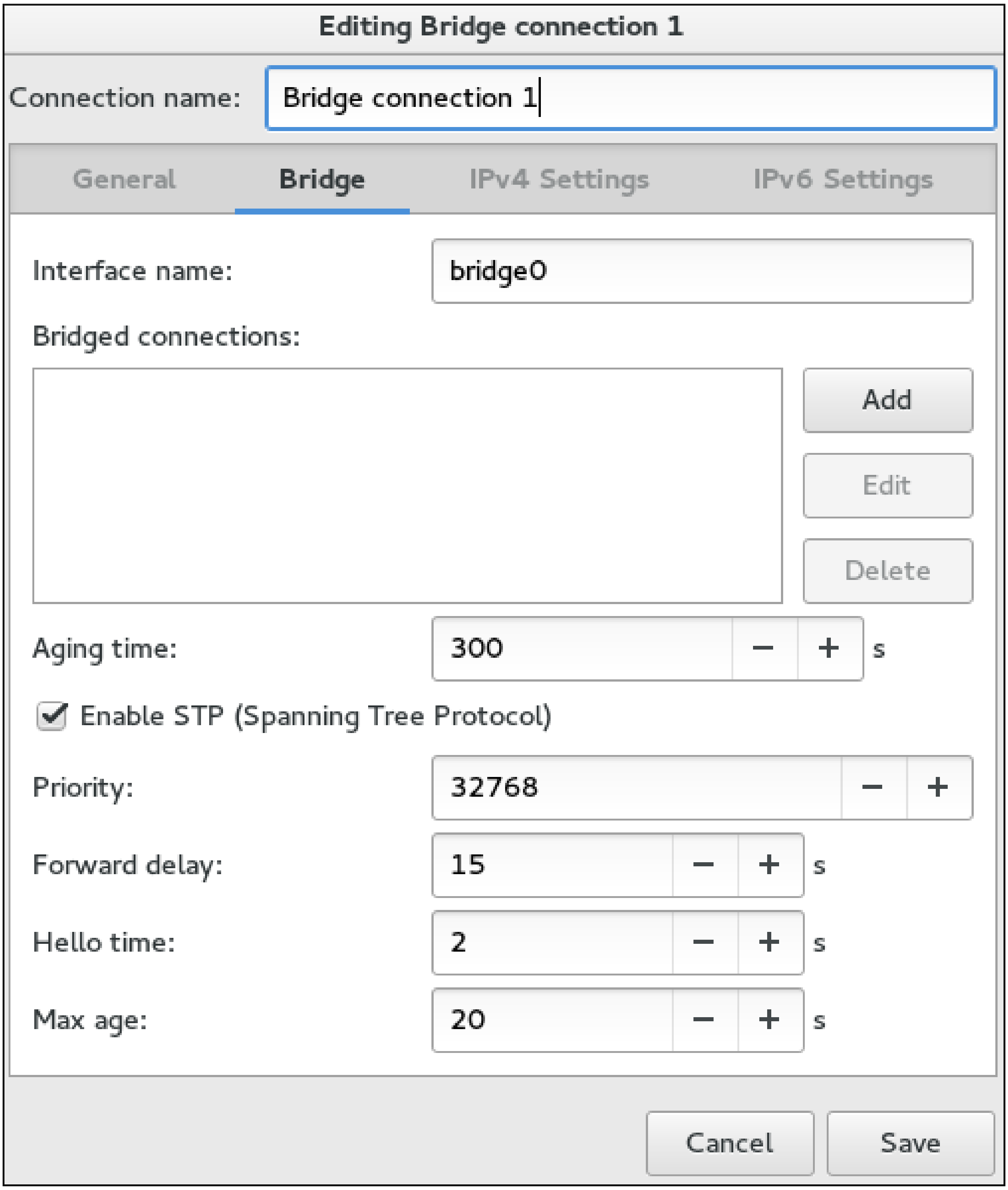
Enable STP (Spanning Tree Protocol)
: If required, select the check box to enable STP.

Priority
: The bridge priority; the bridge with the lowest priority will be elected as the root bridge.

Forward delay
: The time, in seconds, spent in both the Listening and Learning states before entering the Forwarding state. The default is 15 seconds.

Hello time
: The time interval, in seconds, between sending configuration information in bridge protocol data units (BPDU).

Max age
: The maximum time, in seconds, to store the configuration information from BPDUs. This value should be twice the Hello Time plus 1 but less than twice the Forwarding delay minus 1.


Editing Bridge connection 1

Connection name: Bridge connection 1

Interface name: bridge0

Bridged connections:

Aging time: 300 s

[✓] Enable STP (Spanning Tree Protocol)

Priority: 32768

Forward delay: 15 s

Hello time: 2 s

Max age: 20 s

Figure 6.1. Editing Bridge Connection 1

Procedure 6.3. Adding a Slave Interface to a Bridge

1. To add a port to a bridge, select the **Bridge** tab in the **Editing Bridge connection *1*** window. If necessary, open this window by following the procedure in *Procedure 6.2, “Editing an Existing Bridge Connection”*.

2. Click **Add**. The **Choose a Connection Type** menu appears.

3. Select the type of connection to be created from the list. Click **Create**. A window appropriate to the connection type selected appears.

4. Select the **Bridge Port** tab. Configure **Priority** and **Path cost** as required. Note the STP priority for a bridge port is limited by the Linux kernel. Although the standard allows a range of **`0`** to **`255`**, Linux only allows **`0`** to **`63`**. The default is **`32`** in this case.

5. If required, select the **Hairpin mode** check box to enable forwarding of frames for external processing. Also known as *virtual Ethernet port aggregator* (VEPA) mode.

Then, to configure:

- An Ethernet slave, click the **Ethernet** tab and proceed to *Section 2.2.5.1, “Configuring the Connection Name, Auto-Connect Behavior, and Availability Settings”*, or;
- A Bond slave, click the **Bond** tab and proceed to *Section 4.2.1.1, “Configuring the Bond Tab”*, or;
- A Team slave, click the **Team** tab and proceed to *Section 5.9.1.1, “Configuring the Team Tab”*, or;
- An VLAN slave, click the **VLAN** tab and proceed to *Section 7.2.1.1, “Configuring the VLAN Tab”*, or;

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your new bridge connection, click the **Save** button to save your customized configuration. If the profile was in use while being edited, power cycle the connection to make **NetworkManager** apply the changes. If the profile is OFF, set it to ON. See *Section 2.2.1, “Connecting to a Network Using a GUI”* for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network** window and clicking **Options** to return to the **Editing** dialog.

Then, to configure:

- `IPv4` settings for the connection, click the **IPv4 Settings** tab and proceed to *Section 2.2.10.4, “Configuring IPv4 Settings”*, or;
- `IPv6` settings for the connection, click the **IPv6 Settings** tab and proceed to *Section 2.2.10.5, “Configuring IPv6 Settings”*.

6.2. Using the Command Line Interface (CLI)

6.2.1. Check if Bridging Kernel Module is Installed

In Fedora, the bridging module is loaded by default. If necessary, you can make sure that the module is loaded by issuing the following command as `root`:

```
~]# modprobe --first-time bridge
modprobe: ERROR: could not insert 'bridge': Module already in kernel
```

To display information about the module, issue the following command:

```
~]$ modinfo bridge
```

See the **`modprobe(8)`** man page for more command options.

6.2.2. Create a Network Bridge

To create a network bridge, create a file in the **`/etc/sysconfig/network-scripts/`** directory called **`ifcfg-brN`**, replacing *N* with the number for the interface, such as **`0`**.

The contents of the file is similar to whatever type of interface is getting bridged to, such as an Ethernet interface. The differences in this example are as follows:

- The **`DEVICE`** directive is given an interface name as its argument in the format **`brN`**, where *N* is replaced with the number of the interface.
- The **`TYPE`** directive is given an argument **`Bridge`** or **`Ethernet`**. This directive determines the device type and the argument is case sensitive.
- The bridge interface configuration file is given an `IP` address whereas the physical interface configuration file must only have a MAC address (see below).
- An extra directive, **`DELAY=0`**, is added to prevent the bridge from waiting while it monitors traffic, learns where hosts are located, and builds a table of MAC addresses on which to base its filtering decisions. The default delay of 30 seconds is not needed if no routing loops are possible.

The following is a sample bridge interface configuration file using a static `IP` address:

Example 6.1. Sample ifcfg-br0 Interface Configuration File

```
DEVICE=br0
TYPE=Bridge
IPADDR=192.168.1.1
PREFIX=24
ONBOOT=yes
BOOTPROTO=none
DELAY=0
```

To complete the bridge another interface is created, or an existing interface is modified, and pointed to the bridge interface. The following is a sample Ethernet interface configuration file pointing to a bridge interface. Configure your physical interface in **`/etc/sysconfig/network-scripts/ifcfg-ethX`**, where *X* is a unique number corresponding to a specific interface, as follows:

Example 6.2. Sample ifcfg-ethX Interface Configuration File

```
DEVICE=ethX
TYPE=Ethernet
HWADDR=AA:BB:CC:DD:EE:FF
BOOTPROTO=none
ONBOOT=yes
BRIDGE=br0
```

Note

For the **`DEVICE`** directive, almost any interface name could be used as it does not determine the device type. **`TYPE=Ethernet`** is not strictly required. If the **`TYPE`** directive is not set, the device is treated as an Ethernet device (unless it's name explicitly matches a different interface configuration file.)

Specifying the hardware or MAC address using the **`HWADDR`** directive will influence the device naming procedure as explained in *Chapter 8, Consistent Network Device Naming*.

Warning

If you are configuring bridging on a remote host, and you are connected to that host over the physical NIC you are configuring, please consider the implications of losing connectivity before proceeding. You will lose connectivity when restarting the service and may not be able to regain connectivity if any errors have been made. Console, or out-of-band access is advised.

Restart the networking service in order for the changes to take effect. As `root` issue the following command:

```
~]# systemctl network restart
```

6.2.3. Network Bridge with Bond

An example of a network bridge formed from two or more bonded Ethernet interfaces will now be given as this is another common application in a virtualization environment. If you are not very familiar with the configuration files for bonded interfaces then please refer to *Section 4.3.2, “Create a Channel Bonding Interface”*

Create or edit two or more Ethernet interface configuration files, which are to be bonded, as follows:

```
DEVICE=ethX
TYPE=Ethernet
SLAVE=yes
MASTER=bond0
BOOTPROTO=none
HWADDR=AA:BB:CC:DD:EE:FF
```

Note

Using `eth`*X* as the interface name is common practice but almost any name could be used.

Create or edit one interface configuration file, **`/etc/sysconfig/network-scripts/ifcfg-bond0`**, as follows:

```
DEVICE=bond0
ONBOOT=yes
BONDING_OPTS='mode=1 miimon=100'
BRIDGE=brbond0
```

For further instructions and advice on configuring the bonding module and to view the list of bonding parameters, see *Section 4.4, “Using Channel Bonding”*.

Create or edit one interface configuration file, **`/etc/sysconfig/network-scripts/ifcfg-brbond0`**, as follows:

```
DEVICE=brbond0
ONBOOT=yes
TYPE=Bridge
IPADDR=192.168.1.1
PREFIX=24
```

We now have two or more interface configuration files with the **`MASTER=bond0`** directive. These point to the configuration file named **`/etc/sysconfig/network-scripts/ifcfg-bond0`**, which contains the **`DEVICE=bond0`** directive. This **`ifcfg-bond0`** in turn points to the **`/etc/sysconfig/network-scripts/ifcfg-brbond0`** configuration file, which contains the `IP` address, and acts as an interface to the virtual networks inside the host.

To bring up the new or recently configured interfaces, issue a command as `root` in the following format:

```
ifup device
```

This command will detect if **NetworkManager** is running and call **`nmcli con load`** ***`UUID`*** and then call **`nmcli con up`** ***`UUID`***.

Alternatively, to reload all interfaces, issue the following command as `root`:

```
~]# systemctl restart network
```

This command will stop the network service, start the network service, and then call **`ifup`** for all ifcfg files with **`ONBOOT=yes`**.

Note

The default behavior is for **NetworkManager** not to be aware of changes to ifcfg files and to continue using the old configuration data until the interface is next brought up. This is set by the **`monitor-connection-files`** option in the **`NetworkManager.conf`** file. See the **`NetworkManager.conf(5)`** manual page for more information.

6.3. Using the NetworkManager Command Line Tool, nmcli

To create a bridge, with name bridge-br0, issue a command as follows:

```
~]$ nmcli con add type bridge ifname br0
Connection 'bridge-br0' (79cf6a3e-0310-4a78-b759-bda1cc3eef8d) successfully added.
```

If no interface name is specified, the name will default to bridge, bridge-1, bridge-2, and so on.

To view the connections, issue the following command:

```
~]$ nmcli con show conf
NAME        UUID                                  TYPE            TIMESTAMP-REAL
eth0        4d5c449a-a6c5-451c-8206-3c9a4ec88bca 802-3-ethernet Mon 21 Oct 2013 16:01:53 BST
bridge-br0  79cf6a3e-0310-4a78-b759-bda1cc3eef8d bridge         never
```

Spanning tree protocol (STP) according to the IEEE 802.1D standard is enabled by default. To disable STP for this bridge, issue a command as follows:

```
~]$ nmcli con bridge-br0 stp no
```

To re-enable `802.1D STP` for this bridge, issue a command as follows:

```
~]$ nmcli con bridge-br0 stp yes
```

The default bridge priority for `802.1D STP` is **`32768`**. The lower number is preferred in root bridge selection. For example, a bridge with priority of **`28672`** would be selected as the root bridge in preference to a bridge with priority value of **`32768`** (the default). To create a bridge with a non-default value, issue a command as follows:

```
~]$ nmcli con add type bridge ifname br5 stp yes priority 28672
Connection 'bridge-br5' (86b83ad3-b466-4795-aeb6-4a66eb1856c7) successfully added.
```

The allowed values are in the range **`0`** to **`65535`**, but can only be set in multiples of **`4096`**.

To change the bridge priority of an existing bridge to a non-default value, issue a command in the following format:

```
~]$ nmcli connection modify bridge-br5 bridge.priority 36864
```

The allowed values are in the range **`0`** to **`65535`**, but can only be set in multiples of **`4096`**.

Further options for `802.1D STP` are listed in the bridge section of the **`nmcli(1)`** man page.

To add, or enslave an interface, for example eth1, to the bridge bridge-br0, issue a command as follows:

```
~]$ nmcli con add type bridge-slave ifname eth1 master bridge-br0
Connection 'bridge-slave-eth1' (70ffae80-7428-4d9c-8cbd-2e35de72476e) successfully added.
```

At time of writing, **nmcli** only supports Ethernet slaves.

To change a value using interactive mode, issue the following command:

```
~]$ nmcli connection edit bridge-br0
```

You will be placed at the **nmcli** prompt.

```
nmcli> set bridge.priority 4096
nmcli> save
Connection 'bridge-br0' (79cf6a3e-0310-4a78-b759-bda1cc3eef8d) successfully saved.
nmcli> quit
```

See *Section 2.4, “Using the NetworkManager Command Line Tool, nmcli”* for an introduction to **nmcli**

6.4. Additional Resources

The following sources of information provide additional resources regarding network bridging.

6.4.1. Installed Documentation

- `nmcli(1)` man page — Describes **NetworkManager**'s command-line tool.
- `nmcli-examples(5)` man page — Gives examples of **nmcli** commands.
- `nm-settings(5)` man page — Description of settings and parameters of **NetworkManager** connections.

Configure 802.1Q VLAN tagging

To create a VLAN, an interface is created on another interface referred to as the *parent interface*. The VLAN interface will tag packets with the VLAN ID as they pass through the interface, and returning packets will be untagged. VLAN interface can be configured similarly to any other interface. The parent interface need not be an Ethernet interface. An 802.1Q VLAN tagging interface can be created on bridge, bond, and team interfaces, however there are some things to note:

- In the case of VLANs over bonds, it is important that the bond has slaves and that they are “up” before bringing up the VLAN interface. At the time of writing, adding a VLAN interface to a bond without slaves does not work.
- A VLAN slave cannot be configured on a bond with the **`fail_over_mac=follow`** option, because the VLAN virtual device cannot change its MAC address to match the parent's new MAC address. In such a case, traffic would still be sent with the now incorrect source MAC address.
- Sending VLAN tagged packets through a network switch requires configuration of the switch. Refer to the documentation for the switch. For example, for Cisco switches the port must be assigned to one VLAN or configured to be a trunk port to accept tagged packets for multiple VLANs. Untagged packets can also be processed by a trunk port and processed as belonging to the *native VLAN*, however this is a security risk and may have been disabled, or by default not enabled, depending on the make of the switch.
- Some older network interface cards, loopback interfaces, Wimax cards, and some InfiniBand devices, are said to be *VLAN challenged*, meaning they cannot support VLANs. This is usually because the devices cannot cope with VLAN headers and the larger MTU size associated with tagged packets.

7.1. Selecting VLAN Interface Configuration Methods

- **To configure a network using graphical user interface tools**, proceed to *Section 7.2, “Configure 802.1Q VLAN Tagging Using a GUI”*
- **To configure a VLAN interface using NetworkManager**'s command-line tool, **nmcli**, proceed to *Section 7.4, “Configure 802.1Q VLAN Tagging Using the Command Line Tool, nmcli”*
- **To configure a network interface manually**, see *Section 7.3, “Configure 802.1Q VLAN Tagging Using the Command Line”*.

7.2. Configure 802.1Q VLAN Tagging Using a GUI

7.2.1. Establishing a VLAN Connection

You can use the GNOME **control-center** utility to direct **NetworkManager** to create a VLAN using an existing interface as the parent interface. At time of writing, you can only make VLANs on Ethernet devices. Note that VLAN devices are only created automatically if the parent interface is set to connect automatically.

Procedure 7.1. Adding a New VLAN Connection

You can configure a VLAN connection by opening the **Network** window, clicking the plus symbol, and selecting **VLAN** from the list.

1. Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.

2. Click the plus symbol to open the selection list. Select **VLAN**. The **Editing VLAN Connection *1*** window appears.

3. On the **VLAN** tab, select the parent interface from the drop-down list you want to use for the VLAN connection.

4. Enter the VLAN ID

5. Enter a VLAN interface name. This is the name of the VLAN interface that will be created. For example, **`eth0.1`** or **`vlan2`**. (Normally this is either the parent interface name plus "**`.`**" and the VLAN ID, or "**`vlan`**" plus the VLAN ID.)

6. Review and confirm the settings and then click the **Save** button.

7. To edit the VLAN-specific settings see *Section 7.2.1.1, "Configuring the VLAN Tab"*.

Procedure 7.2. Editing an Existing VLAN Connection

Follow these steps to edit an existing VLAN connection.

1. Press the **`Super`** key to enter the Activities Overview, type **`control network`** and then press **`Enter`**. The **Network** settings tool appears.

2. Select the connection you want to edit and click the **Options** button.

3. Select the **General** tab.

4. Configure the connection name, auto-connect behavior, and availability settings.

   These settings in the **Editing** dialog are common to all connection types:

   - **Connection name** — Enter a descriptive name for your network connection. This name will be used to list this connection in the **VLAN** section of the **Network** window.

   - **Automatically connect to this network when it is available** — Select this box if you want **NetworkManager** to auto-connect to this connection when it is available. Refer to *Section 2.2.3, "Connecting to a Network Automatically"* for more information.

   - **Available to all users** — Select this box to create a connection available to all users on the system. Changing this setting may require root privileges. Refer to *Section 2.2.4, "System-wide and Private Connection Profiles"* for details.

5. To edit the VLAN-specific settings see *Section 7.2.1.1, "Configuring the VLAN Tab"*.

Saving Your New (or Modified) Connection and Making Further Configurations

Once you have finished editing your VLAN connection, click the **Save** button to save your customized configuration. If the profile was in use while being edited, power cycle the connection to make **NetworkManager** apply the changes. If the profile is OFF, set it to ON. See *Section 2.2.1, "Connecting to a Network Using a GUI"* for information on using your new or altered connection.

You can further configure an existing connection by selecting it in the **Network** window and clicking **Options** to return to the **Editing** dialog.

Then, to configure:

- IPv4 settings for the connection, click the **IPv4 Settings** tab and proceed to *Section 2.2.10.4, "Configuring IPv4 Settings"*.

7.2.1.1. Configuring the VLAN Tab

If you have already added a new VLAN connection (refer to *Procedure 7.1, “Adding a New VLAN Connection”* for instructions), you can edit the **VLAN** tab to set the parent interface and the VLAN ID.

Parent Interface
: A previously configured interface can be selected in the drop-down list.

VLAN ID
: The identification number to be used to tag the VLAN network traffic.

VLAN interface name
: The name of the VLAN interface that will be created. For example, **`eth0.1`** or **`vlan2`**.

Cloned MAC address
: Optionally sets an alternate MAC address to use for identifying the VLAN interface. This can be used to change the source MAC address for packets sent on this VLAN.

MTU
: Optionally sets a Maximum Transmission Unit (MTU) size to be used for packets to be sent over the VLAN connection.

7.3. Configure 802.1Q VLAN Tagging Using the Command Line

In Fedora, the `8021q` module is loaded by default. If necessary, you can make sure that the module is loaded by issuing the following command as `root`:

```
~]# modprobe --first-time 8021q
modprobe: ERROR: could not insert '8021q': Module already in kernel
```

To display information about the module, issue the following command:

```
~]$ modinfo 8021q
```

See the **`modprobe(8)`** man page for more command options.

7.3.1. Setting Up 802.1Q VLAN Tagging Using ifcfg Files

1. Configure the parent interface in **`/etc/sysconfig/network-scripts/ifcfg-ethX`**, where *X* is a unique number corresponding to a specific interface, as follows:

   ```
   DEVICE=ethX
   TYPE=Ethernet
   BOOTPROTO=none
   ONBOOT=yes
   ```

2. Configure the VLAN interface configuration in the **`/etc/sysconfig/network-scripts/`** directory. The configuration file name should be the parent interface plus a **`.`** character plus the VLAN ID number. For example, if the VLAN ID is 192, and the parent interface is *`eth0`*, then the configuration file name should be **`ifcfg-eth0.192`**:

   ```
   DEVICE=ethX.192
   BOOTPROTO=none
   ```

```
ONBOOT=yes
IPADDR=192.168.1.1
PREFIX=24
NETWORK=192.168.1.0
VLAN=yes
```

If there is a need to configure a second VLAN, with for example, VLAN ID 193, on the same interface, *eth0*, add a new file with the name **eth0.193** with the VLAN configuration details.

3. Restart the networking service in order for the changes to take effect. As `root` issue the following command:

```
~]# systemctl restart network
```

7.3.2. Configure 802.1Q VLAN Tagging Using ip Commands

To create an 802.1Q VLAN interface on Ethernet interface *eth0*, with name *VLAN8* and ID **8**, issue a command as `root` as follows:

```
~]# ip link add link eth0 name eth0.8 type vlan id 8
```

To view the VLAN, issue the following command:

```
~]$ ip -d link show eth0.8
4: eth0.8@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode
 DEFAULT
    link/ether 52:54:00:ce:5f:6c brd ff:ff:ff:ff:ff:ff promiscuity 0
    vlan protocol 802.1Q id 8 <REORDER_HDR>
```

Note that the **ip** utility interprets the VLAN ID as a hexadecimal value if it is preceded by **0x** and as an octal value if it has a leading **0**. This means that in order to assign a VLAN ID with a decimal value of **22**, you must not add any leading zeros.

To remove the VLAN, issue a command as `root` as follows:

```
~]# ip link delete eth0.8
```

Note

VLAN interfaces created using **ip** commands at the command prompt will be lost if the system is shutdown or restarted. To configure VLAN interfaces to be persistent after a system restart, use **ifcfg** files. See *Section 7.3.1, “Setting Up 802.1Q VLAN Tagging Using ifcfg Files”*

7.4. Configure 802.1Q VLAN Tagging Using the Command Line Tool, nmcli

To view the available interfaces on the system, issue a command as follows:

```
~]$ nmcli con show
NAME         UUID                                  TYPE            DEVICE
System eth1  9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  802-3-ethernet  eth1
System eth0  5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  802-3-ethernet  eth0
```

Note that the NAME field in the output always denotes the connection ID. It is not the interface name even though it might look the same. The ID can be used in **`nmcli connection`** commands to identify a connection. Use the DEVICE name with other applications such as `firewalld`.

To create an 802.1Q VLAN interface on Ethernet interface *`eth0`*, with VLAN interface *`VLAN10`* and ID **`10`**, issue a command as follows:

```
~]$ nmcli con add type vlan ifname VLAN10 dev eth0 id 10
Connection 'vlan-VLAN10' (37750b4a-8ef5-40e6-be9b-4fb21a4b6d17) successfully added.
```

Note that as no **`con-name`** was given for the VLAN interface, the name was derived from the interface name by prepending the type. Alternatively, specify a name with the **`con-name`** option as follows:

```
~]$ nmcli con add type vlan con-name VLAN12 dev eth0 id 12
Connection 'VLAN12' (b796c16a-9f5f-441c-835c-f594d40e6533) successfully added.
```

Assigning Addresses to VLAN Interfaces

You can use the same **nmcli** commands to assign static and dynamic interface addresses as with any other interface.

For example, a command to create a VLAN interface with a static `IPv4` address and gateway is as follows:

```
~]$ nmcli con add type vlan con-name VLAN20 ifname VLAN20 id 20 \
dev eth0 ip4 10.10.10.10/24 gw4 10.10.10.254
```

To create a VLAN interface with dynamically assigned addressing, issue a command as follows:

```
~]$ nmcli con add type vlan con-name VLAN30 ifname VLAN30 id 30 dev eth0
```

See *Section 2.4.2, “Connecting to a Network Using nmcli”* for examples of using **nmcli** commands to configure interfaces.

To review the VLAN interfaces created, issue a command as follows:

```
~]$ nmcli con show
NAME         UUID                                  TYPE            DEVICE
VLAN12       4129a37d-4feb-4be5-ac17-14a193821755  vlan            eth0.12
System eth1  9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  802-3-ethernet  eth1
System eth0  5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  802-3-ethernet  eth0
vlan-VLAN10  1be91581-11c2-461a-b40d-893d42fed4f4  vlan            VLAN10
```

To view detailed information about the newly configured connection, issue a command as follows:

```
~]$ nmcli -p con show VLAN12
===============================================================================
                      Connection profile details (VLAN12)
===============================================================================
connection.id:                          VLAN12
connection.uuid:                        4129a37d-4feb-4be5-ac17-14a193821755
connection.interface-name:              --
connection.type:                        vlan
connection.autoconnect:                 yes
```

```
…
-------------------------------------------------------------------------------
802-3-ethernet.port:                    --
802-3-ethernet.speed:                   0
802-3-ethernet.duplex:                  --
802-3-ethernet.auto-negotiate:          yes
802-3-ethernet.mac-address:             --
802-3-ethernet.cloned-mac-address:      --
802-3-ethernet.mac-address-blacklist:
802-3-ethernet.mtu:                     auto
…
vlan.interface-name:                    --
vlan.parent:                            eth0
vlan.id:                                12
vlan.flags:                             0 (NONE)
vlan.ingress-priority-map:
vlan.egress-priority-map:
-------------------------------------------------------------------------------
===============================================================================
      Activate connection details (4129a37d-4feb-4be5-ac17-14a193821755)
===============================================================================
GENERAL.NAME:                           VLAN12
GENERAL.UUID:                           4129a37d-4feb-4be5-ac17-14a193821755
GENERAL.DEVICES:                        eth0.12
GENERAL.STATE:                          activating
[output truncated]
```

Further options for the VLAN command are listed in the VLAN section of the **nmcli(1)** man page. In the man pages the device on which the VLAN is created is referred to as the parent device. In the example above the device was specified by its interface name, **eth0**, it can also be specified by the connection UUID or MAC address.

To create an 802.1Q VLAN connection profile with ingress priority mapping on Ethernet interface *eth1*, with name VLAN13 and ID **13**, issue a command as follows:

```
~]$ nmcli con add type vlan con-name VLAN13 dev eth2 id 13 ingress "2:3,3:5"
```

To view all the parameters associated with the VLAN created above, issue a command as follows:

```
~]$ nmcli connection show VLAN13
```

To change the MTU, issue a command as follows:

```
~]$ nmcli connection modify VLAN13 802.mtu 1496
```

The MTU setting determines the maximum size of the network layer packet. The maximum size of the payload the link-layer frame can carry in turn limits the network layer MTU. For standard Ethernet frames this means an MTU of 1500 bytes. It should not be necessary to change the MTU when setting up a VLAN as the link-layer header is increased in size by 4 bytes to accommodate the 802.1Q tag.

At time of writing, **connection.interface-name** and **vlan.interface-name** have to be the same (if they are set). They must therefore be changed simultaneously using **nmcli**'s interactive mode. To change a VLAN connections name, issue commands as follows:

```
~]$ nmcli con edit vlan-VLAN10
nmcli> set vlan.interface-name superVLAN
nmcli> set connection.interface-name superVLAN
nmcli> save
nmcli> quit
```

The **nmcli** utility can be used to set and clear **`ioctl`** flags which change the way the 802.1Q code functions. The following VLAN flags are supported by **NetworkManager**:

- 0x01 - reordering of output packet headers
- 0x02 - use GVRP protocol
- 0x04 - loose binding of the interface and its master

The state of the VLAN is synchronized to the state of the parent or master interface (the interface or device on which the VLAN is created). If the parent interface is set to the “down” administrative state then all associated VLANs are set down and all routes are flushed from the routing table. Flag **`0x04`** enables a *loose binding* mode, in which only the operational state is passed from the parent to the associated VLANs, but the VLAN device state is not changed.

To set a VLAN flag, issue a command as follows:

```
~]$ nmcli connection modify vlan-VLAN10 vlan.flags 1
```

See *Section 2.4, “Using the NetworkManager Command Line Tool, nmcli”* for an introduction to **nmcli**.

7.5. Additional Resources

The following sources of information provide additional resources regarding Network Teaming.

7.5.1. Installed Documentation

- **`ip-link(8)`** man page — Describes the **ip** utility's network device configuration commands.
- **`nmcli(1)`** man page — Describes **NetworkManager**'s command-line tool.
- **`nmcli-examples(5)`** man page — Gives examples of **nmcli** commands.
- **`nm-settings(5)`** man page — Description of settings and parameters of **NetworkManager** connections.

Consistent Network Device Naming

Fedora 24 provides methods for consistent and predictable network device naming for network interfaces. These features change the name of network interfaces on a system in order to make locating and differentiating the interfaces easier.

Traditionally, network interfaces in Linux are enumerated as `eth[0123…]`, but these names do not necessarily correspond to actual 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 motherboard (*Lan-on-Motherboard*, or *LOM*) and add-in (single and multiport) adapters.

In Fedora 24, **udev** supports a number of different naming schemes. The default is to assign fixed names based on firmware, topology, and location information. This has the advantage that the names are fully automatic, fully predictable, that they stay fixed even if hardware is added or removed (no re-enumeration takes place), and that broken hardware can be replaced seamlessly. The disadvantage is that they are sometimes harder to read than the eth0 or wlan0 names traditionally used. For example: enp5s0.

8.1. Naming Schemes Hierarchy

By default, `systemd` will name interfaces using the following policy to apply the supported naming schemes:

- **Scheme 1:** Names incorporating Firmware or BIOS provided index numbers for on-board devices (example: **`eno1`**), are applied if that information from the firmware or BIOS is applicable and available, else falling back to scheme 2.
- **Scheme 2:** Names incorporating Firmware or BIOS provided PCI Express hotplug slot index numbers (example: **`ens1`**) are applied if that information from the firmware or BIOS is applicable and available, else falling back to scheme 3.
- **Scheme 3:** Names incorporating physical location of the connector of the hardware (example: **`enp2s0`**), are applied if applicable, else falling directly back to scheme 5 in all other cases.
- **Scheme 4:** Names incorporating interface's MAC address (example: **`enx78e7d1ea46da`**), is not used by default, but is available if the user chooses.
- **Scheme 5:** The traditional unpredictable kernel naming scheme, is used if all other methods fail (example: **`eth0`**).

This policy, the procedure outlined above, is the default. If the system has **biosdevname** enabled, it will be used. Note that enabling **biosdevname** requires passing **`biosdevname=1`** as a command-line parameter except in the case of a Dell system, where **biosdevname** will be used by default as long as it is installed. If the user has added **udev** rules which change the name of the kernel devices, those rules will take precedence.

8.2. Understanding the Device Renaming Procedure

The device name procedure in detail is as follows:

1. A rule in **`/usr/lib/udev/rules.d/60-net.rules`** instructs the **udev** helper utility, **/lib/udev/rename_device**, to look into all **`/etc/sysconfig/network-scripts/ifcfg-`*`suffix`*** files. If it finds an **`ifcfg`** file with a **`HWADDR`** entry matching the MAC address of an interface it renames the interface to the name given in the **`ifcfg`** file by the **`DEVICE`** directive.

2. A rule in **`/usr/lib/udev/rules.d/71-biosdevname.rules`** instructs **biosdevname** to rename the interface according to its naming policy, provided that it was not renamed in a previous step, **biosdevname** is installed, and **`biosdevname=0`** was not given as a kernel command on the boot command line.

3. A rule in **`/lib/udev/rules.d/75-net-description.rules`** instructs **udev** to fill in the internal **udev** device property values ID_NET_NAME_ONBOARD, ID_NET_NAME_SLOT, ID_NET_NAME_PATH, ID_NET_NAME_MAC by examining the network interface device. Note, that some device properties might be undefined.

4. A rule in **`/usr/lib/udev/rules.d/80-net-name-slot.rules`** instructs **udev** to rename the interface, provided that it was not renamed in step 1 or 2, and the kernel parameter **`net.ifnames=0`** was not given, according to the following priority: ID_NET_NAME_ONBOARD, ID_NET_NAME_SLOT, ID_NET_NAME_PATH. It falls through to the next in the list, if one is unset. If none of these are set, then the interface will not be renamed.

Steps 3 and 4 are implementing the naming schemes 1, 2, 3, and optionally 4, described in *Section 8.1, “Naming Schemes Hierarchy”*. Step 2 is explained in more detail in *Section 8.6, “Consistent Network Device Naming Using biosdevname”*.

8.3. Understanding the Predictable Network Interface Device Names

The names have two character prefixes based on the type of interface:

1. **`en`** for Ethernet,

2. **`wl`** for wireless LAN (WLAN),

3. **`ww`** for wireless wide area network (WWAN).

The names have the following types:

Table 8.1. Device Name Types

Format	Description
o<*index*>	on-board device index number
s<*slot*>[f<*function*>][d<*dev_id*>]	hotplug slot index number
x<*MAC*>	MAC address
p<*bus*>s<*slot*>[f<*function*>][d<*dev_id*>]	PCI geographical location
p<*bus*>s<*slot*>[f<*function*>][u<*port*>][..][c<*config*>][i<*interface*>]	USB port number chain

- All multi-function PCI devices will carry the [f<*function*>] number in the device name, including the function 0 device.

- For USB devices the full chain of port numbers of hubs is composed. If the name gets longer than the maximum number of 15 characters, the name is not exported.

- The USB configuration descriptors == 1 and USB interface descriptors == 0 values are suppressed (configuration == 1 and interface == 0 are the default values if only one USB configuration or interface exists).

8.4. Naming Scheme for Network Devices Available for Linux on System z

Use the bus-ID to create predictable device names for network interfaces in Linux on System z instances. The bus-ID identifies a device in the s390 channel subsystem. A bus ID identifies the device within the scope of a Linux instance. For a CCW device, the bus ID is the device's device number with a leading **`0.n`**, where **`n`** is the subchannel set ID. For example, **`0.1.0ab1`**.

Network interfaces of device type Ethernet are named as follows:

```
enccw0.0.1234
```

CTC network devices of device type SLIP are named as follows:

```
slccw0.0.1234
```

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

Table 8.2. Device Name Types for Linux on System z

Format	Description
enccw*bus-ID*	device type Ethernet
slccw*bus-ID*	CTC network devices of device type SLIP

8.5. Naming Scheme for VLAN Interfaces

Traditionally, VLAN interface names in the format: *interface-name.VLAN-ID* are used. The **`VLAN-ID`** ranges from **`0`** to **`4096`**, which is a maximum of four characters and the total interface name has a limit of 15 characters. The maximum interface name length is defined by the kernel headers and is a global limit, affecting all applications.

In Fedora, four naming conventions for VLAN interface names are supported:

VLAN plus VLAN ID
: The word **`vlan`** plus the VLAN ID. For example: vlan0005

VLAN plus VLAN ID without padding
: The word **`vlan`** plus the VLAN ID without padding by means of additional leading zeros. For example: vlan5

Device name plus VLAN ID
: The name of the parent interface plus the VLAN ID. For example: eth0.0005

Device name plus VLAN ID without padding
: The name of the parent interface plus the VLAN ID without padding by means of additional leading zeros. For example: eth0.5

8.6. Consistent Network Device Naming Using biosdevname

This feature, implemented via the **biosdevname udev** helper utility, will change the name of all embedded network interfaces, PCI card network interfaces, and virtual function network interfaces from the existing `eth[0123…]` to the new naming convention as shown in *Table 8.3, “The biosdevname Naming Convention”*. Note that unless the system is a Dell system, or **biosdevname** is explicitly enabled as described in *Section 8.6.2, “Enabling and Disabling the Feature”*, the `systemd` naming scheme will take precedence.

Table 8.3. The biosdevname Naming Convention

Device	Old Name	New Name
Embedded network interface (LOM)	`eth[0123…]`	`em[1234…]`[1]
PCI card network interface	`eth[0123…]`	p*<slot>*p*<ethernet port>*[2]
Virtual function	`eth[0123…]`	p*<slot>*p*<ethernet port>*_*<virtual interface>*[3]

[1] New enumeration starts at **1**.
[2] For example: `p3p4`
[3] For example: `p3p4_1`

8.6.1. System Requirements

The **biosdevname** program uses information from the system's BIOS, specifically the *type 9* (System Slot) and *type 41* (Onboard Devices Extended Information) fields contained within the SMBIOS. If the system's BIOS does not have SMBIOS version 2.6 or higher and this data, the new naming convention will not be used. Most older hardware does not support this feature because of a lack of BIOSes with the correct SMBIOS version and field information. For BIOS or SMBIOS version information, contact your hardware vendor.

For this feature to take effect, the *biosdevname* package must also be installed. To install it, issue the following command as `root`:

```
~]# dnf install biosdevname
```

8.6.2. Enabling and Disabling the Feature

To disable this feature, pass the following option on the boot command line, both during and after installation:

```
biosdevname=0
```

To enable this feature, pass the following option on the boot command line, both during and after installation:

```
biosdevname=1
```

Unless the system meets the minimum requirements, this option will be ignored and the system will use the `systemd` naming scheme as described in the beginning of the chapter.

If the **`biosdevname`** install option is specified, it must remain as a boot option for the lifetime of the system.

8.7. Notes for Administrators

Many system customization files can include network interface names, and thus will require updates if moving a system from the old convention to the new convention. If you use the new naming convention, you will also need to update network interface names in areas such as custom **iptables** rules, scripts altering `irqbalance`, and other similar configuration files. Also, enabling this change for installation will require modification to existing **kickstart** files that use device names via the **ksdevice** parameter; these **kickstart** files will need to be updated to use the network device's MAC address or the network device's new name.

Note

The maximum interface name length is defined by the kernel headers and is a global limit, affecting all applications.

8.8. Controlling the Selection of Network Device Names

Device naming can be controlled in the following manner:

By identifying the network interface device
: Setting the MAC address in an **ifcfg** file using the **HWADDR** directive enables it to be identified by **udev**. The name will be taken from the string given by the **DEVICE** directive, which by convention is the same as the **ifcfg** suffix. For example, **ifcfg**-*eth0*.

By turning on or off **biosdevname**
: The name provided by **biosdevname** will be used (if **biosdevname** can determine one).

By turning on or off the `systemd-udev` naming scheme
: The name provided by `systemd-udev` will be used (if `systemd-udev` can determine one).

8.9. Disabling Consistent Network Device Naming

To disable consistent network device naming, choose from one of the following:

- Disable the assignment of fixed names, so that the unpredictable kernel names are used again, by masking **udev**'s rule file for the default policy. This “masking” can be done by making a symbolic link to **/dev/null**. As `root`, issue a command as follows:

```
~]# ln -s /dev/null /etc/udev/rules.d/80-net-name-slot.rules
```

- Create your own manual naming scheme, for example by naming your interfaces “internet0”, “dmz0” or “lan0”. To do that create your own **udev** rules file and set the NAME property for the devices. Make sure to order it before the default policy file, for example by naming it **/etc/udev/rules.d/70-my-net-names.rules**.

- Alter the default policy file to pick a different naming scheme, for example to name all interfaces after their MAC address by default. As `root` copy the default policy file as follows:

```
~]# cp /usr/lib/udev/rules.d/80-net-name-slot.rules /etc/udev/rules.d/80-net-name-
slot.rules
```

  Edit the file in the **/etc/udev/rules.d/** directory and change the lines as necessary.

- Append the following directive to the kernel command line in the GRUB 2 menu:

```
net.ifnames=0
```

  To update all the GRUB 2 kernel menu entries, enter a command as `root` as follows:

```
~]# grubby --update-kernel=ALL --args=net.ifnames=0
```

For more information about working with GRUB 2 see the *Fedora 24 System Administrator's Guide*[1].

8.10. Troubleshooting Network Device Naming

Predictable interface names will be assigned for each interface, if applicable, as per the procedure described in *Section 8.2, “Understanding the Device Renaming Procedure”*. To view the list of possible names **udev** will use, issue a command in the following form as `root`:

```
~]# udevadm info /sys/class/net/ifname | grep ID_NET_NAME
```

where *`ifname`* is one of the interfaces listed by the following command:

```
~]$ ls /sys/class/net/
```

One of the possible names will be applied by **udev** according to the rules as described in *Section 8.2, “Understanding the Device Renaming Procedure”*, and summarized here:

- **`/usr/lib/udev/rules.d/60-net.rules`** - from initscripts,
- **`/usr/lib/udev/rules.d/71-biosdevname.rules`** - from **biosdevname**,
- **`/usr/lib/udev/rules.d/80-net-name-slot.rules`** - from `systemd`

From the above list of rule files it can be seen that if interface naming is done via initscripts or **biosdevname** it always takes precedence over **udev** native naming. However if initscripts renaming is not taking place and **biosdevname** is disabled, then to alter the interface names copy the **`80-net-name-slot.rules`** from **`/usr/`** to **`/etc/`** and edit the file appropriately. In other words, comment out or arrange schemes to be used in a certain order.

Example 8.1. Some interfaces have names from the kernel namespace (eth[0,1,2...]) while others are successfully renamed by udev

Mixed up schemes most likely means that either for some hardware there is no usable information provided by the kernel to **udev**, thus it could not figure out any names, or the information provided to **udev** is not suitable, for example non-unique device IDs. The latter is more common and the solution is to use a custom naming scheme in ifcfg files or alter which **udev** scheme is in use by editing 80-net-name-slot.rules.

Example 8.2. In /var/log/messages or the systemd journal, renaming is seen to be done twice for each interface

Systems with the naming scheme encoded in ifcfg files but which do not have a regenerated `initrd` image are likely to encounter this issue. The interface name is initially assigned (via **biosdevname** or **udev** or dracut parameters on the kernel command line) during early-boot while still in `initrd`. Then after switching to real `rootfs`, renaming is done a second time and a new interface name is determined by the **`/usr/lib/udev/rename_device`** binary spawned by **udev** because of processing 60-net.rules. You can safely ignore such messages.

[1] https://docs.fedoraproject.org/sysadmin-guide

> Example 8.3. Using naming scheme in ifcfg files with ethX names does not work
> Use of interface names from kernel namespace is discouraged. To get predictable and stable interface names please use some other prefix than "eth".

8.11. Additional Resources

The following sources of information provide additional resources regarding Network Teaming.

8.11.1. Installed Documentation

- **`udev(7)`** man page — Describes the Linux dynamic device management daemon, `udevd`.
- **`systemd(1)`** man page — Describes `systemd` system and service manager.
- **`biosdevname(1)`** man page — Describes the utility for obtaining the BIOS-given name of a device.

Part II. Servers

This part discusses how to set up servers normally required for networking.

DHCP Servers

Dynamic Host Configuration Protocol (DHCP) is a network protocol that automatically assigns TCP/IP information to client machines. Each `DHCP` client connects to the centrally located `DHCP` server, which returns the network configuration (including the `IP` address, gateway, and `DNS` servers) of that client.

9.1. Why Use DHCP?

`DHCP` is useful for automatic configuration of client network interfaces. When configuring the client system, you can choose `DHCP` instead of specifying an `IP` address, netmask, gateway, or `DNS` servers. The client retrieves this information from the `DHCP` server. `DHCP` is also useful if you want to change the `IP` addresses of a large number of systems. Instead of reconfiguring all the systems, you can just edit one configuration file on the server for the new set of `IP` addresses. If the `DNS` servers for an organization changes, the changes happen on the `DHCP` server, not on the `DHCP` clients. When you restart the network or reboot the clients, the changes go into effect.

If an organization has a functional `DHCP` server correctly connected to a network, laptops and other mobile computer users can move these devices from office to office.

Note that administrators of `DNS` and `DHCP` servers, as well as any provisioning applications, should agree on the host name format used in an organization. See *Section 3.1.1, “Recommended Naming Practices”* for more information on the format of host names.

9.2. Configuring a DHCP Server

The *dhcp* package contains an *Internet Systems Consortium* (ISC) `DHCP` server. Install the package as `root`:

```
~]# dnf install dhcp
```

Installing the *dhcp* package creates a file, **`/etc/dhcp/dhcpd.conf`**, which is merely an empty configuration file. As `root`, issue the following command:

```
~]# cat /etc/dhcp/dhcpd.conf
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp/dhcpd.conf.example
#   see dhcpd.conf(5) man page
#
```

The example configuration file can be found at **`/usr/share/doc/dhcp/dhcpd.conf.example`**. You should use this file to help you configure **`/etc/dhcp/dhcpd.conf`**, which is explained in detail below.

`DHCP` also uses the file **`/var/lib/dhcpd/dhcpd.leases`** to store the client lease database. Refer to *Section 9.2.2, “Lease Database”* for more information.

9.2.1. Configuration File

The first step in configuring a `DHCP` server is to create the configuration file that stores the network information for the clients. Use this file to declare options for client systems.

The configuration file can contain extra tabs or blank lines for easier formatting. Keywords are case-insensitive and lines beginning with a hash sign (#) are considered comments.

There are two types of statements in the configuration file:

- Parameters — State how to perform a task, whether to perform a task, or what network configuration options to send to the client.
- Declarations — Describe the topology of the network, describe the clients, provide addresses for the clients, or apply a group of parameters to a group of declarations.

The parameters that start with the keyword option are referred to as *options*. These options control `DHCP` options; whereas, parameters configure values that are not optional or control how the `DHCP` server behaves.

Parameters (including options) declared before a section enclosed in curly brackets (**`{ }`**) are considered global parameters. Global parameters apply to all the sections below it.

Restart the DHCP Daemon for the Changes to Take Effect

If the configuration file is changed, the changes do not take effect until the `DHCP` daemon is restarted with the command **`systemctl restart dhcpd`**.

Use the omshell Command

Instead of changing a `DHCP` configuration file and restarting the service each time, using the **`omshell`** command provides an interactive way to connect to, query, and change the configuration of a `DHCP` server. By using **`omshell`**, all changes can be made while the server is running. For more information on **`omshell`**, see the **`omshell`** man page.

In *Example 9.1, “Subnet Declaration”*, the **`routers`**, **`subnet-mask`**, **`domain-search`**, **`domain-name-servers`**, and **`time-offset`** options are used for any **`host`** statements declared below it.

For every subnet which will be served, and for every subnet to which the `DHCP` server is connected, there must be one **`subnet`** declaration, which tells the `DHCP` daemon how to recognize that an address is on that subnet. A **`subnet`** declaration is required for each subnet even if no addresses will be dynamically allocated to that subnet.

In this example, there are global options for every `DHCP` client in the subnet and a **`range`** declared. Clients are assigned an `IP` address within the **`range`**.

Example 9.1. Subnet Declaration

```
subnet 192.168.1.0 netmask 255.255.255.0 {
        option routers                  192.168.1.254;
        option subnet-mask              255.255.255.0;
        option domain-search              "example.com";
        option domain-name-servers       192.168.1.1;
        option time-offset              -18000;     # Eastern Standard Time
 range 192.168.1.10 192.168.1.100;
}
```

To configure a `DHCP` server that leases a dynamic `IP` address to a system within a subnet, modify the example values from *Example 9.2, "Range Parameter"*. It declares a default lease time, maximum lease time, and network configuration values for the clients. This example assigns `IP` addresses in the **`range`** `192.168.1.10` and `192.168.1.100` to client systems.

Example 9.2. Range Parameter

```
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.254;
option domain-name-servers 192.168.1.1, 192.168.1.2;
option domain-search "example.com";
subnet 192.168.1.0 netmask 255.255.255.0 {
   range 192.168.1.10 192.168.1.100;
}
```

To assign an `IP` address to a client based on the MAC address of the network interface card, use the **`hardware ethernet`** parameter within a **`host`** declaration. As demonstrated in *Example 9.3, "Static IP Address Using DHCP"*, the **`host apex`** declaration specifies that the network interface card with the MAC address `00:A0:78:8E:9E:AA` always receives the `IP` address `192.168.1.4`.

Note that you can also use the optional parameter **`host-name`** to assign a host name to the client.

Example 9.3. Static IP Address Using DHCP

```
host apex {
   option host-name "apex.example.com";
   hardware ethernet 00:A0:78:8E:9E:AA;
   fixed-address 192.168.1.4;
}
```

All subnets that share the same physical network should be declared within a **`shared-network`** declaration as shown in *Example 9.4, "Shared-network Declaration"*. Parameters within the **`shared-network`**, but outside the enclosed subnet declarations, are considered to be global parameters. The name assigned to **`shared-network`** must be a descriptive title for the network, such as using the title "test-lab" to describe all the subnets in a test lab environment.

Example 9.4. Shared-network Declaration

```
shared-network name {
    option domain-search              "test.redhat.com";
    option domain-name-servers      ns1.redhat.com, ns2.redhat.com;
    option routers                  192.168.0.254;
    #more parameters for EXAMPLE shared-network
    subnet 192.168.1.0 netmask 255.255.252.0 {
        #parameters for subnet
        range 192.168.1.1 192.168.1.254;
    }
    subnet 192.168.2.0 netmask 255.255.252.0 {
        #parameters for subnet
        range 192.168.2.1 192.168.2.254;
    }
}
```

As demonstrated in *Example 9.5, “Group Declaration”*, the **`group`** declaration is used to apply global parameters to a group of declarations. For example, shared networks, subnets, and hosts can be grouped.

Example 9.5. Group Declaration

```
group {
   option routers                  192.168.1.254;
   option subnet-mask              255.255.255.0;
   option domain-search              "example.com";
   option domain-name-servers       192.168.1.1;
   option time-offset              -18000;     # Eastern Standard Time
   host apex {
      option host-name "apex.example.com";
      hardware ethernet 00:A0:78:8E:9E:AA;
      fixed-address 192.168.1.4;
   }
   host raleigh {
      option host-name "raleigh.example.com";
      hardware ethernet 00:A1:DD:74:C3:F2;
      fixed-address 192.168.1.6;
   }
}
```

Using the Example Configuration File

You can use the provided example configuration file as a starting point and add custom configuration options to it. To copy this file to the proper location, use the following command as `root`:

```
~]# cp /usr/share/doc/dhcp-version_number/dhcpd.conf.example /etc/dhcp/dhcpd.conf
```

... where *`version_number`* is the `DHCP` version number.

For a complete list of option statements and what they do, see the **`dhcp-options(5)`** man page.

9.2.2. Lease Database

On the `DHCP` server, the file **`/var/lib/dhcpd/dhcpd.leases`** stores the `DHCP` client lease database. Do not change this file. `DHCP` lease information for each recently assigned `IP` address is automatically stored in the lease database. The information includes the length of the lease, to whom the `IP` address has been assigned, the start and end dates for the lease, and the MAC address of the network interface card that was used to retrieve the lease.

All times in the lease database are in Coordinated Universal Time (UTC), not local time.

The lease database is recreated from time to time so that it is not too large. First, all known leases are saved in a temporary lease database. The **`dhcpd.leases`** file is renamed **`dhcpd.leases~`** and the temporary lease database is written to **`dhcpd.leases`**.

The `DHCP` daemon could be killed or the system could crash after the lease database has been renamed to the backup file but before the new file has been written. If this happens, the **`dhcpd.leases`** file does not exist, but it is required to start the service. Do not create a new lease file. If you do, all old leases are lost which causes many problems. The correct solution is to rename the **`dhcpd.leases~`** backup file to **`dhcpd.leases`** and then start the daemon.

9.2.3. Starting and Stopping the Server

Starting the DHCP Server for the First Time

When the `DHCP` server is started for the first time, it fails unless the **`dhcpd.leases`** file exists. You can use the command **`touch /var/lib/dhcpd/dhcpd.leases`** to create the file if it does not exist. If the same server is also running BIND as a `DNS` server, this step is not necessary, as starting the **named** service automatically checks for a **`dhcpd.leases`** file.

Do not create a new lease file on a system that was previously running. If you do, all old leases are lost which causes many problems. The correct solution is to rename the **`dhcpd.leases~`** backup file to **`dhcpd.leases`** and then start the daemon.

To start the `DHCP` service, use the following command:

```
systemctl start dhcpd.service
```

To stop the `DHCP` server, type:

```
systemctl stop dhcpd.service
```

By default, the `DHCP` service does not start at boot time. To configure the daemon to start automatically at boot time, run:

```
systemctl enable dhcpd.service
```

If more than one network interface is attached to the system, but the `DHCP` server should only listen for `DHCP` requests on one of the interfaces, configure the `DHCP` server to listen only on that device. The `DHCP` daemon will only listen on interfaces for which it finds a subnet declaration in the **`/etc/dhcp/dhcpd.conf`** file.

This is useful for a firewall machine with two network cards. One network card can be configured as a `DHCP` client to retrieve an `IP` address to the Internet. The other network card can be used as a `DHCP` server for the internal network behind the firewall. Specifying only the network card connected to the internal network makes the system more secure because users can not connect to the daemon via the Internet.

To specify command-line options, copy and then edit the **dhcpd.service** file as the `root` user. For example, as follows:

```
~]# cp /usr/lib/systemd/system/dhcpd.service /etc/systemd/system/
~]# vi /etc/systemd/system/dhcpd.service
```

Edit the line under section [Service]:

```
ExecStart=/usr/sbin/dhcpd -f -cf /etc/dhcp/dhcpd.conf -user dhcpd -group dhcpd --no-
pid your_interface_name(s)
```

Then, as the `root` user, restart the service:

```
~]# systemctl --system daemon-reload
~]# systemctl restart dhcpd
```

Command line options can be appended to **ExecStart=/usr/sbin/dhcpd** in the **/etc/systemd/system/dhcpd.service** unit file under section [Service]. They include:

- **-p *portnum*** — Specifies the UDP port number on which **dhcpd** should listen. The default is port 67. The `DHCP` server transmits responses to the `DHCP` clients at a port number one greater than the UDP port specified. For example, if the default port 67 is used, the server listens on port 67 for requests and responds to the client on port 68. If a port is specified here and the `DHCP` relay agent is used, the same port on which the `DHCP` relay agent should listen must be specified. See *Section 9.3, “DHCP Relay Agent”* for details.

- **-f** — Runs the daemon as a foreground process. This is mostly used for debugging.

- **-d** — Logs the `DHCP` server daemon to the standard error descriptor. This is mostly used for debugging. If this is not specified, the log is written to **/var/log/messages**.

- **-cf *filename*** — Specifies the location of the configuration file. The default location is **/etc/dhcp/dhcpd.conf**.

- **-lf *filename*** — Specifies the location of the lease database file. If a lease database file already exists, it is very important that the same file be used every time the `DHCP` server is started. It is strongly recommended that this option only be used for debugging purposes on non-production machines. The default location is **/var/lib/dhcpd/dhcpd.leases**.

- **-q** — Do not print the entire copyright message when starting the daemon.

9.3. 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 the DHCP Relay Agent is started. When a `DHCP` server returns a reply, the reply is broadcast or unicast on the network that sent the original request.

The DHCP Relay Agent for `IPv4`, **dhcrelay**, listens for `DHCPv4` and `BOOTP` requests on all interfaces unless the interfaces are specified in **/etc/sysconfig/dhcrelay** with the **INTERFACES** directive.

See *Section 9.3.1, “Configure dhcrelay as a DHCPv4 and BOOTP relay agent”*. The DHCP Relay Agent for `IPv6`, **dhcrelay6**, does not have this default behavior and interfaces to listen for `DHCPv6` requests must be specified. See *Section 9.3.2, “Configure dhcrelay as a DHCPv6 relay agent”*.

dhcrelay can either be run as a `DHCPv4` and `BOOTP` relay agent (by default) or as a `DHCPv6` relay agent (with **`-6`** argument). To see the usage message, issue the command **`dhcrelay -h`**.

9.3.1. Configure dhcrelay as a DHCPv4 and BOOTP relay agent

To run **dhcrelay** in `DHCPv4` and `BOOTP` mode specify the servers to which the requests should be forwarded to. Copy and then edit the **`dhcrelay.service`** file as the `root` user:

```
~]# cp /lib/systemd/system/dhcrelay.service /etc/systemd/system/
~]# vi /etc/systemd/system/dhcrelay.service
```

Edit the **`ExecStart`** option under section [Service] and add one or more server `IPv4` addresses to the end of the line, for example:

```
ExecStart=/usr/sbin/dhcrelay -d --no-pid 192.168.1.1
```

If you also want to specify interfaces where the DHCP Relay Agent listens for `DHCP` requests, add them to the **`ExecStart`** option with **`-i`** argument (otherwise it will listen on all interfaces), for example:

```
ExecStart=/usr/sbin/dhcrelay -d --no-pid 192.168.1.1 -i em1
```

For other options see the **`dhcrelay(8)`** man page.

To activate the changes made, as the `root` user, restart the service:

```
~]# systemctl --system daemon-reload
~]# systemctl restart dhcrelay
```

9.3.2. Configure dhcrelay as a DHCPv6 relay agent

To run **dhcrelay** in `DHCPv6` mode add the **`-6`** argument and specify the “lower interface” (on which queries will be received from clients or from other relay agents) and the “upper interface” (to which queries from clients and other relay agents should be forwarded). Copy **`dhcrelay.service`** to **`dhcrelay6.service`** and edit it as the `root` user:

```
~]# cp /lib/systemd/system/dhcrelay.service /etc/systemd/system/dhcrelay6.service
~]# vi /etc/systemd/system/dhcrelay6.service
```

Edit the **`ExecStart`** option under section [Service] add **`-6`** argument and add the “lower interface” and “upper interface” interface, for example:

```
ExecStart=/usr/sbin/dhcrelay -d --no-pid -6 -l em1 -u em2
```

For other options see the **`dhcrelay(8)`** man page.

To activate the changes made, as the `root` user, restart the service:

```
~]# systemctl --system daemon-reload
~]# systemctl restart dhcrelay6
```

9.4. Configuring a Multihomed DHCP Server

A multihomed DHCP server serves multiple networks, that is, multiple subnets. The examples in these sections detail how to configure a DHCP server to serve multiple networks, select which network interfaces to listen on, and how to define network settings for systems that move networks.

Before making any changes, back up the existing **/etc/dhcp/dhcpd.conf** file.

The DHCP daemon will only listen on interfaces for which it finds a subnet declaration in the **/etc/dhcp/dhcpd.conf** file.

The following is a basic **/etc/dhcp/dhcpd.conf** file, for a server that has two network interfaces, eth0 in a 10.0.0.0/24 network, and eth1 in a 172.16.0.0/24 network. Multiple **subnet** declarations allow you to define different settings for multiple networks:

```
default-lease-time 600;
max-lease-time 7200;
subnet 10.0.0.0 netmask 255.255.255.0 {
 option subnet-mask 255.255.255.0;
 option routers 10.0.0.1;
 range 10.0.0.5 10.0.0.15;
}
subnet 172.16.0.0 netmask 255.255.255.0 {
 option subnet-mask 255.255.255.0;
 option routers 172.16.0.1;
 range 172.16.0.5 172.16.0.15;
}
```

subnet *10.0.0.0* netmask *255.255.255.0*;

A **subnet** declaration is required for every network your DHCP server is serving. Multiple subnets require multiple **subnet** declarations. If the DHCP server does not have a network interface in a range of a **subnet** declaration, the DHCP server does not serve that network.

If there is only one **subnet** declaration, and no network interfaces are in the range of that subnet, the DHCP daemon fails to start, and an error such as the following is logged to **/var/log/messages**:

```
dhcpd: No subnet declaration for eth0 (0.0.0.0).
dhcpd: ** Ignoring requests on eth0.  If this is not what
dhcpd:    you want, please write a subnet declaration
dhcpd:    in your dhcpd.conf file for the network segment
dhcpd:    to which interface eth1 is attached. **
dhcpd:
dhcpd:
dhcpd: Not configured to listen on any interfaces!
```

option subnet-mask *255.255.255.0*;

The **option subnet-mask** option defines a subnet mask, and overrides the **netmask** value in the **subnet** declaration. In simple cases, the subnet and netmask values are the same.

option routers *10.0.0.1*;

The **option routers** option defines the default gateway for the subnet. This is required for systems to reach internal networks on a different subnet, as well as external networks.

range *10.0.0.5 10.0.0.15*;

The **range** option specifies the pool of available IP addresses. Systems are assigned an address from the range of specified IP addresses.

For further information, see the **`dhcpd.conf(5)`** man page.

9.4.1. Host Configuration

Before making any changes, back up the existing **`/etc/sysconfig/dhcpd`** and **`/etc/dhcp/dhcpd.conf`** files.

Configuring a Single System for Multiple Networks

The following **`/etc/dhcp/dhcpd.conf`** example creates two subnets, and configures an `IP` address for the same system, depending on which network it connects to:

```
default-lease-time 600;
max-lease-time 7200;
subnet 10.0.0.0 netmask 255.255.255.0 {
 option subnet-mask 255.255.255.0;
 option routers 10.0.0.1;
 range 10.0.0.5 10.0.0.15;
}
subnet 172.16.0.0 netmask 255.255.255.0 {
 option subnet-mask 255.255.255.0;
 option routers 172.16.0.1;
 range 172.16.0.5 172.16.0.15;
}
host example0 {
 hardware ethernet 00:1A:6B:6A:2E:0B;
 fixed-address 10.0.0.20;
}
host example1 {
 hardware ethernet 00:1A:6B:6A:2E:0B;
 fixed-address 172.16.0.20;
}
```

`host` ***`example0`***

The **`host`** declaration defines specific parameters for a single system, such as an `IP` address. To configure specific parameters for multiple hosts, use multiple **`host`** declarations.

Most `DHCP` clients ignore the name in **`host`** declarations, and as such, this name can be anything, as long as it is unique to other **`host`** declarations. To configure the same system for multiple networks, use a different name for each **`host`** declaration, otherwise the `DHCP` daemon fails to start. Systems are identified by the **`hardware ethernet`** option, not the name in the **`host`** declaration.

`hardware ethernet` ***`00:1A:6B:6A:2E:0B`*****`;`**

The **`hardware ethernet`** option identifies the system. To find this address, run the **`ip link`** command.

`fixed-address` ***`10.0.0.20`*****`;`**

The **`fixed-address`** option assigns a valid `IP` address to the system specified by the **`hardware ethernet`** option. This address must be outside the `IP` address pool specified with the **`range`** option.

If **`option`** statements do not end with a semicolon, the `DHCP` daemon fails to start, and an error such as the following is logged to **`/var/log/messages`**:

```
/etc/dhcp/dhcpd.conf line 20: semicolon expected.
dhcpd: }
dhcpd: ^
```

```
dhcpd: /etc/dhcp/dhcpd.conf line 38: unexpected end of file
dhcpd:
dhcpd: ^
dhcpd: Configuration file errors encountered -- exiting
```

Configuring Systems with Multiple Network Interfaces

The following **`host`** declarations configure a single system, which has multiple network interfaces, so that each interface receives the same `IP` address. This configuration will not work if both network interfaces are connected to the same network at the same time:

```
host interface0 {
 hardware ethernet 00:1a:6b:6a:2e:0b;
 fixed-address 10.0.0.18;
}
host interface1 {
 hardware ethernet 00:1A:6B:6A:27:3A;
 fixed-address 10.0.0.18;
}
```

For this example, **`interface0`** is the first network interface, and **`interface1`** is the second interface. The different **`hardware ethernet`** options identify each interface.

If such a system connects to another network, add more **`host`** declarations, remembering to:

- assign a valid **`fixed-address`** for the network the host is connecting to.
- make the name in the **`host`** declaration unique.

When a name given in a **`host`** declaration is not unique, the `DHCP` daemon fails to start, and an error such as the following is logged to **`/var/log/messages`**:

```
dhcpd: /etc/dhcp/dhcpd.conf line 31: host interface0: already exists
dhcpd: }
dhcpd: ^
dhcpd: Configuration file errors encountered -- exiting
```

This error was caused by having multiple **`host interface0`** declarations defined in **`/etc/dhcp/dhcpd.conf`**.

9.5. DHCP for IPv6 (DHCPv6)

The ISC `DHCP` includes support for `IPv6` (`DHCPv6`) since the 4.x release with a `DHCPv6` server, client, and relay agent functionality. The agents support both `IPv4` and `IPv6`, however the agents can only manage one protocol at a time; for dual support they must be started separately for `IPv4` and `IPv6`. For example, configure both `DHCPv4` and `DHCPv6` by editing their respective configuration files **`/etc/dhcp/dhcpd.conf`** and **`/etc/dhcp/dhcpd6.conf`** and then issue the following commands:

```
~]# systemctl start dhcpd
~]# systemctl start dhcpd6
```

The `DHCPv6` server configuration file can be found at **`/etc/dhcp/dhcpd6.conf`**.

The example server configuration file can be found at **`/usr/share/doc/dhcp/dhcpd6.conf.example`**.

A simple DHCPv6 server configuration file can look like this:

```
subnet6 2001:db8:0:1::/64 {
        range6 2001:db8:0:1::129 2001:db8:0:1::254;
        option dhcp6.name-servers fec0:0:0:1::1;
        option dhcp6.domain-search "domain.example";
}
```

9.6. Additional Resources

For additional information, see *The DHCP Handbook; Ralph Droms and Ted Lemon; 2003* or the following resources.

9.6.1. Installed Documentation

- **dhcpd(8)** man page — Describes how the DHCP daemon works.
- **dhcpd.conf(5)** man page — Explains how to configure the DHCP configuration file; includes some examples.
- **dhcpd.leases(5)** man page — Describes a persistent database of leases.
- **dhcp-options(5)** man page — Explains the syntax for declaring DHCP options in **dhcpd.conf**; includes some examples.
- **dhcrelay(8)** man page — Explains the DHCP Relay Agent and its configuration options.
- **/usr/share/doc/dhcp/** — Contains example configuration files.
- **/usr/share/doc/dhcp-common/** — Contains README files, and release notes for current versions of the DHCP service.

DNS Servers

DNS (Domain Name System), is a distributed database system that is used to associate host names with their respective IP addresses. For users, this has the advantage that they can refer to machines on the network by names that are usually easier to remember than the numerical network addresses. For system administrators, using a DNS server, also known as a *nameserver*, enables changing the IP address for a host without ever affecting the name-based queries. The use of the DNS databases is not only for resolving IP addresses to domain names and their use is becoming broader and broader as DNSSEC is deployed.

10.1. Introduction to DNS

DNS is usually implemented using one or more centralized servers that are authoritative for certain domains. When a client host requests information from a nameserver, it usually connects to port 53. The nameserver then attempts to resolve the name requested. If the nameserver is configured to be a recursive name servers and it does not have an authoritative answer, or does not already have the answer cached from an earlier query, it queries other nameservers, called *root nameservers*, to determine which nameservers are authoritative for the name in question, and then queries them to get the requested name. Nameservers configured as purely authoritative, with recursion disabled, will not do lookups on behalf of clients.

10.1.1. Nameserver Zones

In a DNS server, all information is stored in basic data elements called *resource records* (RR). Resource records are defined in *RFC 1034*[1]. The domain names are organized into a tree structure. Each level of the hierarchy is divided by a period (`.`). For example: The root domain, denoted by `.`, is the root of the DNS tree, which is at level zero. The domain name `com`, referred to as the *top-level domain* (TLD) is a child of the root domain (`.`) so it is the first level of the hierarchy. The domain name `example.com` is at the second level of the hierarchy.

Example 10.1. A Simple Resource Record

An example of a simple *resource record* (RR):

```
example.com.      86400    IN         A           192.0.2.1
```

The domain name, `example.com`, is the *owner* for the RR. The value **`86400`** is the *time to live* (TTL). The letters **`IN`**, meaning “the Internet system”, indicate the *class* of the RR. The letter **`A`** indicates the *type* of RR (in this example, a host address). The host address `192.0.2.1` is the data contained in the final section of this RR. This one line example is a RR. A set of RRs with the same type, owner, and class is called a *resource record set* (RRSet).

Zones are defined on authoritative nameservers through the use of *zone files*, which contain definitions of the resource records in each zone. Zone files are stored on *primary nameservers* (also called *master nameservers*), where changes are made to the files, and *secondary nameservers* (also called *slave nameservers*), which receive zone definitions from the primary nameservers. Both primary and secondary nameservers are authoritative for the zone and look the same to clients. Depending on

[1] http://www.rfc-editor.org/rfc/rfc1034.txt

the configuration, any nameserver can also serve as a primary or secondary server for multiple zones at the same time.

Note that administrators of DNS and DHCP servers, as well as any provisioning applications, should agree on the host name format used in an organization. See *Section 3.1.1, “Recommended Naming Practices”* for more information on the format of host names.

10.1.2. Nameserver Types

There are two nameserver configuration types:

authoritative
: Authoritative nameservers answer to resource records that are part of their zones only. This category includes both primary (master) and secondary (slave) nameservers.

recursive
: Recursive nameservers offer resolution services, but they are not authoritative for any zone. Answers for all resolutions are cached in a memory for a fixed period of time, which is specified by the retrieved resource record.

Although a nameserver can be both authoritative and recursive at the same time, it is recommended not to combine the configuration types. To be able to perform their work, authoritative servers should be available to all clients all the time. On the other hand, since the recursive lookup takes far more time than authoritative responses, recursive servers should be available to a restricted number of clients only, otherwise they are prone to distributed denial of service (DDoS) attacks.

10.1.3. BIND as a Nameserver

BIND consists of a set of DNS-related programs. It contains a nameserver called `named`, an administration utility called **`rndc`**, and a debugging tool called **`dig`**. See *Fedora 24 System Administrator's Guide*[2] for more information on how to run a service in Fedora.

10.2. BIND

This section covers `BIND` (Berkeley Internet Name Domain), the `DNS` server included in Fedora. It focuses on the structure of its configuration files, and describes how to administer it both locally and remotely.

10.2.1. Empty Zones

`BIND` configures a number of “empty zones” to prevent recursive servers from sending unnecessary queries to Internet servers that cannot handle them (thus creating delays and SERVFAIL responses to clients who query for them). These empty zones ensure that immediate and authoritative NXDOMAIN responses are returned instead. The configuration option **`empty-zones-enable`** controls whether or not empty zones are created, whilst the option **`disable-empty-zone`** can be used in addition to disable one or more empty zones from the list of default prefixes that would be used.

[2] https://docs.fedoraproject.org/sysadmin-guide

The number of empty zones created for *RFC 1918*[3] prefixes has been increased, and users of `BIND 9.9` and above will see the *RFC 1918*[4] empty zones both when **`empty-zones-enable`** is unspecified (defaults to **`yes`**), and when it is explicitly set to **`yes`**.

10.2.2. Configuring the named Service

When the `named` service is started, it reads the configuration from the files as described in *Table 10.1, "The named Service Configuration Files"*.

Table 10.1. The named Service Configuration Files

Path	Description
`/etc/named.conf`	The main configuration file.
`/etc/named/`	An auxiliary directory for configuration files that are included in the main configuration file.

The configuration file consists of a collection of statements with nested options surrounded by opening and closing curly brackets (**{** and **}**). Note that when editing the file, you have to be careful not to make any syntax error, otherwise the `named` service will not start. A typical **`/etc/named.conf`** file is organized as follows:

```
statement-1 ["statement-1-name"] [statement-1-class] {
  option-1;
  option-2;
  option-N;
};
statement-2 ["statement-2-name"] [statement-2-class] {
  option-1;
  option-2;
  option-N;
};
statement-N ["statement-N-name"] [statement-N-class] {
  option-1;
  option-2;
  option-N;
};
```

[3] http://www.rfc-editor.org/info/rfc1918
[4] http://www.rfc-editor.org/info/rfc1918

Running BIND in a chroot environment

If you have installed the *bind-chroot* package, the BIND service will run in the **chroot** environment. In that case, the initialization script will mount the above configuration files using the **mount --bind** command, so that you can manage the configuration outside this environment. There is no need to copy anything into the **/var/named/chroot/** directory because it is mounted automatically. This simplifies maintenance since you do not need to take any special care of BIND configuration files if it is run in a **chroot** environment. You can organize everything as you would with BIND not running in a **chroot** environment.

The following directories are automatically mounted into the **/var/named/chroot/** directory if the corresponding mount point directories underneath **/var/named/chroot/** are empty:

- **/etc/named**
- **/etc/pki/dnssec-keys**
- **/run/named**
- **/var/named**
- **/usr/lib64/bind** or **/usr/lib/bind** (architecture dependent).

The following files are also mounted if the target file does not exist in **/var/named/chroot/**:

- **/etc/named.conf**
- **/etc/rndc.conf**
- **/etc/rndc.key**
- **/etc/named.rfc1912.zones**
- **/etc/named.dnssec.keys**
- **/etc/named.iscdlv.key**
- **/etc/named.root.key**

Important

Editing files which have been mounted in a **chroot** environment requires creating a backup copy and then editing the original file. Alternatively, use an editor with “edit-a-copy” mode disabled. For example, to edit the BIND's configuration file, **/etc/named.conf**, with Vim while it is running in a **chroot** environment, issue the following command as root:

```
~]# vim -c "set backupcopy=yes" /etc/named.conf
```

10.2.2.1. Installing BIND in a chroot Environment

To install **BIND** to run in a **chroot** environment, issue the following command as root:

```
~]# dnf install bind-chroot
```

To enable the `named-chroot` service, first check if the `named` service is running by issuing the following command:

```
~]$ systemctl status named
```

If it is running, it must be disabled.

To disable `named`, issue the following commands as `root`:

```
~]# systemctl stop named
```

```
~]# systemctl disable named
```

Then, to enable the `named-chroot` service, issue the following commands as `root`:

```
~]# systemctl enable named-chroot
```

```
~]# systemctl start named-chroot
```

To check the status of the `named-chroot` service, issue the following command as `root`:

```
~]# systemctl status named-chroot
```

10.2.2.2. Common Statement Types

The following types of statements are commonly used in **`/etc/named.conf`**:

`acl`

The **`acl`** (Access Control List) statement allows you to define groups of hosts, so that they can be permitted or denied access to the nameserver. It takes the following form:

```
acl acl-name {
  match-element;
  ...
};
```

The *`acl-name`* statement name is the name of the access control list, and the *`match-element`* option is usually an individual `IP` address (such as **`10.0.1.1`**) or a *Classless Inter-Domain Routing* (CIDR) network notation (for example, **`10.0.1.0/24`**). For a list of already defined keywords, see *Table 10.2, “Predefined Access Control Lists”*.

Table 10.2. Predefined Access Control Lists

Keyword	Description
`any`	Matches every `IP` address.
`localhost`	Matches any `IP` address that is in use by the local system.
`localnets`	Matches any `IP` address on any network to which the local system is connected.
`none`	Does not match any `IP` address.

The **`acl`** statement can be especially useful in conjunction with other statements such as **`options`**. *Example 10.2, “Using acl in Conjunction with Options”* defines two access control lists, **`black-hats`** and **`red-hats`**, and adds **`black-hats`** on the blacklist while granting **`red-hats`** normal access.

Example 10.2. Using acl in Conjunction with Options

```
acl black-hats {
  10.0.2.0/24;
  192.168.0.0/24;
  1234:5678::9abc/24;
};
acl red-hats {
  10.0.1.0/24;
};
options {
  blackhole { black-hats; };
  allow-query { red-hats; };
  allow-query-cache { red-hats; };
};
```

`include`

The **`include`** statement allows you to include files in the **`/etc/named.conf`**, so that potentially sensitive data can be placed in a separate file with restricted permissions. It takes the following form:

```
include "file-name"
```

The *`file-name`* statement name is an absolute path to a file.

Example 10.3. Including a File to /etc/named.conf

```
include "/etc/named.rfc1912.zones";
```

`options`

The **`options`** statement allows you to define global server configuration options as well as to set defaults for other statements. It can be used to specify the location of the `named` working directory, the types of queries allowed, and much more. It takes the following form:

```
options {
  option;
  ...
};
```

For a list of frequently used *`option`* directives, see *Table 10.3, “Commonly Used Configuration Options”* below.

Table 10.3. Commonly Used Configuration Options

Option	Description
`allow-query`	Specifies which hosts are allowed to query the nameserver for authoritative resource records. It accepts an access control list, a collection of `IP` addresses, or networks in the CIDR notation. All hosts are allowed by default.

Option	Description
`allow-query-cache`	Specifies which hosts are allowed to query the nameserver for non-authoritative data such as recursive queries. Only **`localhost`** and **`localnets`** are allowed by default.
`blackhole`	Specifies which hosts are *not* allowed to query the nameserver. This option should be used when a particular host or network floods the server with requests. The default option is **none**.
`directory`	Specifies a working directory for the `named` service. The default option is **`/var/named/`**.
`disable-empty-zone`	Used to disable one or more empty zones from the list of default prefixes that would be used. Can be specified in the options statement and also in view statements. It can be used multiple times.
`dnssec-enable`	Specifies whether to return DNSSEC related resource records. The default option is **`yes`**.
`dnssec-validation`	Specifies whether to prove that resource records are authentic via DNSSEC. The default option is **`yes`**.
`empty-zones-enable`	Controls whether or not empty zones are created. Can be specified only in the options statement.
`forwarders`	Specifies a list of valid `IP` addresses for nameservers to which the requests should be forwarded for resolution.
`forward`	Specifies the behavior of the **`forwarders`** directive. It accepts the following options: • **`first`** — The server will query the nameservers listed in the **`forwarders`** directive before attempting to resolve the name on its own. • **`only`** — When unable to query the nameservers listed in the **`forwarders`** directive, the server will not attempt to resolve the name on its own.
`listen-on`	Specifies the `IPv4` network interface on which to listen for queries. On a `DNS` server that also acts as a gateway, you can use this option to answer queries originating from a single network only. All `IPv4` interfaces are used by default.
`listen-on-v6`	Specifies the `IPv6` network interface on which to listen for queries. On a `DNS` server that also acts as a gateway, you can use this option to answer queries originating from a single network only. All `IPv6` interfaces are used by default.
`max-cache-size`	Specifies the maximum amount of memory to be used for server caches. When the limit is reached, the server causes records to expire prematurely so that the limit is not exceeded. In a server with multiple views, the limit applies separately to the cache of each view. The default option is **`32M`**.
`notify`	Specifies whether to notify the secondary nameservers when a zone is updated. It accepts the following options: • **`yes`** — The server will notify all secondary nameservers. • **no** — The server will *not* notify any secondary nameserver.

Option	Description
	• **master-only** — The server will notify primary server for the zone only. • **explicit** — The server will notify only the secondary servers that are specified in the **also-notify** list within a zone statement.
pid-file	Specifies the location of the process ID file created by the named service.
recursion	Specifies whether to act as a recursive server. The default option is **yes**.
statistics-file	Specifies an alternate location for statistics files. The **/var/named/named.stats** file is used by default.

Note

The directory used by named for runtime data has been moved from the BIND default location, **/var/run/named/**, to a new location **/run/named/**. As a result, the PID file has been moved from the default location **/var/run/named/named.pid** to the new location **/run/named/named.pid**. In addition, the session-key file has been moved to **/run/named/session.key**. These locations need to be specified by statements in the options section. See *Example 10.4, “Using the options Statement”*.

Restrict recursive servers to selected clients only

To prevent distributed denial of service (DDoS) attacks, it is recommended that you use the **allow-query-cache** option to restrict recursive DNS services for a particular subset of clients only.

Refer to the *BIND 9 Administrator Reference Manual* referenced in *Section 10.2.8.1, “Installed Documentation”*, and the **named.conf** manual page for a complete list of available options.

Example 10.4. Using the options Statement

```
options {
  allow-query       { localhost; };
  listen-on port    53 { 127.0.0.1; };
  listen-on-v6 port 53 { ::1; };
  max-cache-size    256M;
  directory         "/var/named";
  statistics-file   "/var/named/data/named_stats.txt";

  recursion         yes;
  dnssec-enable     yes;
  dnssec-validation yes;

  pid-file          "/run/named/named.pid";
  session-keyfile   "/run/named/session.key";
};
```

zone

The **zone** statement allows you to define the characteristics of a zone, such as the location of its configuration file and zone-specific options, and can be used to override the global **options** statements. It takes the following form:

```
zone zone-name [zone-class] {
  option;
  ...
};
```

The *zone-name* attribute is the name of the zone, *zone-class* is the optional class of the zone, and *option* is a **zone** statement option as described in *Table 10.4, “Commonly Used Options in Zone Statements”*.

The *zone-name* attribute is particularly important, as it is the default value assigned for the **$ORIGIN** directive used within the corresponding zone file located in the **/var/named/** directory. The named daemon appends the name of the zone to any non-fully qualified domain name listed in the zone file. For example, if a **zone** statement defines the namespace for **example.com**, use **example.com** as the *zone-name* so that it is placed at the end of host names within the **example.com** zone file.

For more information about zone files, refer to *Section 10.2.3, “Editing Zone Files”*.

Table 10.4. Commonly Used Options in Zone Statements

Option	Description
allow-query	Specifies which clients are allowed to request information about this zone. This option overrides global **allow-query** option. All query requests are allowed by default.
allow-transfer	Specifies which secondary servers are allowed to request a transfer of the zone's information. All transfer requests are allowed by default.
allow-update	Specifies which hosts are allowed to dynamically update information in their zone. The default option is to deny all dynamic update requests. Note that you should be careful when allowing hosts to update information about their zone. Do not set IP addresses in this option unless the server is in the trusted network. Instead, use TSIG key as described in *Section 10.2.6.3, “Transaction SIGnatures (TSIG)”*.
file	Specifies the name of the file in the named working directory that contains the zone's configuration data.
masters	Specifies from which IP addresses to request authoritative zone information. This option is used only if the zone is defined as **type slave**.
notify	Specifies whether to notify the secondary nameservers when a zone is updated. It accepts the following options: • **yes** — The server will notify all secondary nameservers. • **no** — The server will *not* notify any secondary nameserver. • **master-only** — The server will notify primary server for the zone only. • **explicit** — The server will notify only the secondary servers that are specified in the **also-notify** list within a zone statement.

Option	Description
`type`	Specifies the zone type. It accepts the following options: • **`delegation-only`** — Enforces the delegation status of infrastructure zones such as COM, NET, or ORG. Any answer that is received without an explicit or implicit delegation is treated as **`NXDOMAIN`**. This option is only applicable in TLDs (Top-Level Domain) or root zone files used in recursive or caching implementations. • **`forward`** — Forwards all requests for information about this zone to other nameservers. • **`hint`** — A special type of zone used to point to the root nameservers which resolve queries when a zone is not otherwise known. No configuration beyond the default is necessary with a **`hint`** zone. • **`master`** — Designates the nameserver as authoritative for this zone. A zone should be set as the **`master`** if the zone's configuration files reside on the system. • **`slave`** — Designates the nameserver as a slave server for this zone. Master server is specified in **`masters`** directive.

Most changes to the **`/etc/named.conf`** file of a primary or secondary nameserver involve adding, modifying, or deleting **`zone`** statements, and only a small subset of **`zone`** statement options is usually needed for a nameserver to work efficiently.

In *Example 10.5, “A Zone Statement for a Primary nameserver”*, the zone is identified as **`example.com`**, the type is set to **`master`**, and the `named` service is instructed to read the **`/var/named/example.com.zone`** file. It also allows only a secondary nameserver (**`192.168.0.2`**) to transfer the zone.

Example 10.5. A Zone Statement for a Primary nameserver

```
zone "example.com" IN {
  type master;
  file "example.com.zone";
  allow-transfer { 192.168.0.2; };
};
```

A secondary server's **`zone`** statement is slightly different. The type is set to **`slave`**, and the **`masters`** directive is telling `named` the `IP` address of the master server.

In *Example 10.6, “A Zone Statement for a Secondary nameserver”*, the `named` service is configured to query the primary server at the **`192.168.0.1`** `IP` address for information about the **`example.com`** zone. The received information is then saved to the **`/var/named/slaves/example.com.zone`** file. Note that you have to put all slave zones in the **`/var/named/slaves/`** directory, otherwise the service will fail to transfer the zone.

Example 10.6. A Zone Statement for a Secondary nameserver

```
zone "example.com" {
  type slave;
  file "slaves/example.com.zone";
  masters { 192.168.0.1; };
};
```

10.2.2.3. Other Statement Types

The following types of statements are less commonly used in **/etc/named.conf**:

controls
: The **controls** statement allows you to configure various security requirements necessary to use the **rndc** command to administer the `named` service.

  Refer to *Section 10.2.4, “Using the rndc Utility”* for more information on the **rndc** utility and its usage.

key
: The **key** statement allows you to define a particular key by name. Keys are used to authenticate various actions, such as secure updates or the use of the **rndc** command. Two options are used with **key**:

  - **algorithm *algorithm-name*** — The type of algorithm to be used (for example, **hmac-md5**).
  - **secret "*key-value*"** — The encrypted key.

  Refer to *Section 10.2.4, “Using the rndc Utility”* for more information on the **rndc** utility and its usage.

logging
: The **logging** statement allows you to use multiple types of logs, so called *channels*. By using the **channel** option within the statement, you can construct a customized type of log with its own file name (**file**), size limit (**size**), version number (**version**), and level of importance (**severity**). Once a customized channel is defined, a **category** option is used to categorize the channel and begin logging when the `named` service is restarted.

  By default, `named` sends standard messages to the `rsyslog` daemon, which places them in **/var/log/messages**. Several standard channels are built into BIND with various severity levels, such as **default_syslog** (which handles informational logging messages) and **default_debug** (which specifically handles debugging messages). A default category, called **default**, uses the built-in channels to do normal logging without any special configuration.

  Customizing the logging process can be a very detailed process and is beyond the scope of this chapter. For information on creating custom BIND logs, refer to the *BIND 9 Administrator Reference Manual* referenced in *Section 10.2.8.1, “Installed Documentation”*.

server
: The **server** statement allows you to specify options that affect how the `named` service should respond to remote nameservers, especially with regard to notifications and zone transfers.

  The **transfer-format** option controls the number of resource records that are sent with each message. It can be either **one-answer** (only one resource record), or **many-answers** (multiple resource records). Note that while the **many-answers** option is more efficient, it is not supported by older versions of BIND.

trusted-keys
: The **trusted-keys** statement allows you to specify assorted public keys used for secure `DNS` (DNSSEC). Refer to *Section 10.2.6.4, “DNS Security Extensions (DNSSEC)”* for more information on this topic.

`view`

The **`view`** statement allows you to create special views depending upon which network the host querying the nameserver is on. This allows some hosts to receive one answer regarding a zone while other hosts receive totally different information. Alternatively, certain zones may only be made available to particular trusted hosts while non-trusted hosts can only make queries for other zones.

Multiple views can be used as long as their names are unique. The **`match-clients`** option allows you to specify the `IP` addresses that apply to a particular view. If the **`options`** statement is used within a view, it overrides the already configured global options. Finally, most **`view`** statements contain multiple **`zone`** statements that apply to the **`match-clients`** list.

Note that the order in which the **`view`** statements are listed is important, as the first statement that matches a particular client's `IP` address is used. For more information on this topic, refer to *Section 10.2.6.1, “Multiple Views”*.

10.2.2.4. Comment Tags

Additionally to statements, the **`/etc/named.conf`** file can also contain comments. Comments are ignored by the `named` service, but can prove useful when providing additional information to a user. The following are valid comment tags:

`//`

Any text after the **`//`** characters to the end of the line is considered a comment. For example:

```
notify yes;  // notify all secondary nameservers
```

`#`

Any text after the **`#`** character to the end of the line is considered a comment. For example:

```
notify yes;  # notify all secondary nameservers
```

`/*` and **`*/`**

Any block of text enclosed in **`/*`** and **`*/`** is considered a comment. For example:

```
notify yes;  /* notify all secondary nameservers */
```

10.2.3. Editing Zone Files

As outlined in *Section 10.1.1, “Nameserver Zones”*, zone files contain information about a namespace. They are stored in the `named` working directory located in **`/var/named/`** by default. Each zone file is named according to the **`file`** option in the **`zone`** statement, usually in a way that relates to the domain in and identifies the file as containing zone data, such as **`example.com.zone`**.

Table 10.5. The named Service Zone Files

Path	Description
`/var/named/`	The working directory for the `named` service. The nameserver is *not* allowed to write to this directory.

Path	Description
`/var/named/slaves/`	The directory for secondary zones. This directory is writable by the `named` service.
`/var/named/dynamic/`	The directory for other files, such as dynamic `DNS` (DDNS) zones or managed DNSSEC keys. This directory is writable by the `named` service.
`/var/named/data/`	The directory for various statistics and debugging files. This directory is writable by the `named` service.

A zone file consists of directives and resource records. Directives tell the nameserver to perform tasks or apply special settings to the zone, resource records define the parameters of the zone and assign identities to individual hosts. While the directives are optional, the resource records are required in order to provide name service to a zone.

All directives and resource records should be entered on individual lines.

10.2.3.1. Common Directives

Directives begin with the dollar sign character (**`$`**) followed by the name of the directive, and usually appear at the top of the file. The following directives are commonly used in zone files:

`$INCLUDE`

The **`$INCLUDE`** directive allows you to include another file at the place where it appears, so that other zone settings can be stored in a separate zone file.

Example 10.7. Using the $INCLUDE Directive

```
$INCLUDE /var/named/penguin.example.com
```

`$ORIGIN`

The **`$ORIGIN`** directive allows you to append the domain name to unqualified records, such as those with the host name only. Note that the use of this directive is not necessary if the zone is specified in **`/etc/named.conf`**, since the zone name is used by default.

In *Example 10.8, “Using the $ORIGIN Directive”*, any names used in resource records that do not end in a trailing period (the **`.`** character) are appended with **`example.com`**.

Example 10.8. Using the $ORIGIN Directive

```
$ORIGIN example.com.
```

`$TTL`

The **`$TTL`** directive allows you to set the default *Time to Live* (TTL) value for the zone, that is, how long is a zone record valid. Each resource record can contain its own TTL value, which overrides this directive.

Increasing this value allows remote nameservers to cache the zone information for a longer period of time, reducing the number of queries for the zone and lengthening the amount of time required to propagate resource record changes.

Example 10.9. Using the $TTL Directive

```
$TTL 1D
```

10.2.3.2. Common Resource Records

The following resource records are commonly used in zone files:

A

The *Address* record specifies an `IP` address to be assigned to a name. It takes the following form:

```
hostname IN A IP-address
```

If the *`hostname`* value is omitted, the record will point to the last specified *`hostname`*.

In *Example 10.10, “Using the A Resource Record”*, the requests for `server1.example.com` are pointed to **`10.0.1.3`** or **`10.0.1.5`**.

Example 10.10. Using the A Resource Record

```
server1  IN  A  10.0.1.3
         IN  A  10.0.1.5
```

CNAME

The *Canonical Name* record maps one name to another. Because of this, this type of record is sometimes referred to as an *alias record*. It takes the following form:

```
alias-name IN CNAME real-name
```

`CNAME` records are most commonly used to point to services that use a common naming scheme, such as **`www`** for Web servers. However, there are multiple restrictions for their usage:

- CNAME records should not point to other CNAME records. This is mainly to avoid possible infinite loops.
- CNAME records should not contain other resource record types (such as A, NS, MX, etc.). The only exception are DNSSEC related records (RRSIG, NSEC, etc.) when the zone is signed.
- Other resource records that point to the fully qualified domain name (FQDN) of a host (NS, MX, PTR) should not point to a CNAME record.

In *Example 10.11, “Using the CNAME Resource Record”*, the **A** record binds a host name to an `IP` address, while the **`CNAME`** record points the commonly used **`www`** host name to it.

Example 10.11. Using the CNAME Resource Record

```
server1  IN  A      10.0.1.5
www      IN  CNAME  server1
```

MX

The *Mail Exchange* record specifies where the mail sent to a particular namespace controlled by this zone should go. It takes the following form:

```
IN MX preference-value email-server-name
```

The *`email-server-name`* is a fully qualified domain name (FQDN). The *`preference-value`* allows numerical ranking of the email servers for a namespace, giving preference to some email systems over others. The **`MX`** resource record with the lowest *`preference-value`* is preferred over the others. However, multiple email servers can possess the same value to distribute email traffic evenly among them.

In *Example 10.12, “Using the MX Resource Record”*, the first `mail.example.com` email server is preferred to the `mail2.example.com` email server when receiving email destined for the `example.com` domain.

Example 10.12. Using the MX Resource Record

```
example.com.  IN  MX  10  mail.example.com.
              IN  MX  20  mail2.example.com.
```

`NS`

The *Nameserver* record announces authoritative nameservers for a particular zone. It takes the following form:

```
IN NS nameserver-name
```

The *`nameserver-name`* should be a fully qualified domain name (FQDN). Note that when two nameservers are listed as authoritative for the domain, it is not important whether these nameservers are secondary nameservers, or if one of them is a primary server. They are both still considered authoritative.

Example 10.13. Using the NS Resource Record

```
IN  NS  dns1.example.com.
IN  NS  dns2.example.com.
```

`PTR`

The *Pointer* record points to another part of the namespace. It takes the following form:

```
last-IP-digit IN PTR FQDN-of-system
```

The *`last-IP-digit`* directive is the last number in an `IP` address, and the *`FQDN-of-system`* is a fully qualified domain name (FQDN).

`PTR` records are primarily used for reverse name resolution, as they point `IP` addresses back to a particular name. Refer to *Section 10.2.3.4.2, “A Reverse Name Resolution Zone File”* for examples of **`PTR`** records in use.

`SOA`

The *Start of Authority* record announces important authoritative information about a namespace to the nameserver. Located after the directives, it is the first resource record in a zone file. It takes the following form:

```
@  IN  SOA  primary-name-server hostmaster-email (
       serial-number
```

```
        time-to-refresh
        time-to-retry
        time-to-expire
        minimum-TTL )
```

The directives are as follows:

- The `@` symbol places the **`$ORIGIN`** directive (or the zone's name if the **`$ORIGIN`** directive is not set) as the namespace being defined by this **`SOA`** resource record.
- The *`primary-name-server`* directive is the host name of the primary nameserver that is authoritative for this domain.
- The *`hostmaster-email`* directive is the email of the person to contact about the namespace.
- The *`serial-number`* directive is a numerical value incremented every time the zone file is altered to indicate it is time for the `named` service to reload the zone.
- The *`time-to-refresh`* directive is the numerical value secondary nameservers use to determine how long to wait before asking the primary nameserver if any changes have been made to the zone.
- The *`time-to-retry`* directive is a numerical value used by secondary nameservers to determine the length of time to wait before issuing a refresh request in the event that the primary nameserver is not answering. If the primary server has not replied to a refresh request before the amount of time specified in the *`time-to-expire`* directive elapses, the secondary servers stop responding as an authority for requests concerning that namespace.
- In BIND 4 and 8, the *`minimum-TTL`* directive is the amount of time other nameservers cache the zone's information. In BIND 9, it defines how long negative answers are cached for. Caching of negative answers can be set to a maximum of 3 hours (**`3H`**).

When configuring BIND, all times are specified in seconds. However, it is possible to use abbreviations when specifying units of time other than seconds, such as minutes (**`M`**), hours (**`H`**), days (**`D`**), and weeks (**`W`**). *Table 10.6, “Seconds compared to other time units”* shows an amount of time in seconds and the equivalent time in another format.

Table 10.6. Seconds compared to other time units

Seconds	Other Time Units
60	**`1M`**
1800	**`30M`**
3600	**`1H`**
10800	**`3H`**
21600	**`6H`**
43200	**`12H`**
86400	**`1D`**
259200	**`3D`**
604800	**`1W`**
31536000	**`365D`**

Example 10.14. Using the SOA Resource Record

```
@  IN  SOA  dns1.example.com.  hostmaster.example.com. (
        2001062501  ; serial
        21600       ; refresh after 6 hours
        3600        ; retry after 1 hour
        604800      ; expire after 1 week
        86400 )     ; minimum TTL of 1 day
```

10.2.3.3. Comment Tags

Additionally to resource records and directives, a zone file can also contain comments. Comments are ignored by the `named` service, but can prove useful when providing additional information to the user. Any text after the semicolon character (**`;`**) to the end of the line is considered a comment. For example:

```
    604800  ; expire after 1 week
```

10.2.3.4. Example Usage

The following examples show the basic usage of zone files.

10.2.3.4.1. A Simple Zone File

Example 10.15, “A simple zone file” demonstrates the use of standard directives and **`SOA`** values.

Example 10.15. A simple zone file

```
$ORIGIN example.com.
$TTL 86400
@         IN  SOA  dns1.example.com.  hostmaster.example.com. (
              2001062501  ; serial
              21600       ; refresh after 6 hours
              3600        ; retry after 1 hour
              604800      ; expire after 1 week
              86400 )     ; minimum TTL of 1 day
;
;
          IN  NS     dns1.example.com.
          IN  NS     dns2.example.com.
dns1      IN  A      10.0.1.1
          IN  AAAA   aaaa:bbbb::1
dns2      IN  A      10.0.1.2
          IN  AAAA   aaaa:bbbb::2
;
;
@         IN  MX     10  mail.example.com.
          IN  MX     20  mail2.example.com.
mail      IN  A      10.0.1.5
          IN  AAAA   aaaa:bbbb::5
mail2     IN  A      10.0.1.6
          IN  AAAA   aaaa:bbbb::6
;
;
; This sample zone file illustrates sharing the same IP addresses
; for multiple services:
;
services  IN  A      10.0.1.10
          IN  AAAA   aaaa:bbbb::10
          IN  A      10.0.1.11
          IN  AAAA   aaaa:bbbb::11
```

```
ftp       IN  CNAME  services.example.com.
www       IN  CNAME  services.example.com.
;
;
```

In this example, the authoritative nameservers are set as `dns1.example.com` and `dns2.example.com`, and are tied to the `10.0.1.1` and `10.0.1.2` `IP` addresses respectively using the **`A`** record.

The email servers configured with the **`MX`** records point to `mail` and `mail2` via **`A`** records. Since these names do not end in a trailing period (`.` character), the **`$ORIGIN`** domain is placed after them, expanding them to `mail.example.com` and `mail2.example.com`.

Services available at the standard names, such as `www.example.com` (WWW), are pointed at the appropriate servers using the **`CNAME`** record.

This zone file would be called into service with a **`zone`** statement in the **`/etc/named.conf`** similar to the following:

```
zone "example.com" IN {
  type master;
  file "example.com.zone";
  allow-update { none; };
};
```

10.2.3.4.2. A Reverse Name Resolution Zone File

A reverse name resolution zone file is used to translate an `IP` address in a particular namespace into a fully qualified domain name (FQDN). It looks very similar to a standard zone file, except that the **`PTR`** resource records are used to link the `IP` addresses to a fully qualified domain name as shown in *Example 10.16, “A reverse name resolution zone file”*.

Example 10.16. A reverse name resolution zone file

```
$ORIGIN 1.0.10.in-addr.arpa.
$TTL 86400
@  IN  SOA  dns1.example.com.  hostmaster.example.com. (
       2001062501  ; serial
       21600       ; refresh after 6 hours
       3600        ; retry after 1 hour
       604800      ; expire after 1 week
       86400 )     ; minimum TTL of 1 day
;
@  IN  NS   dns1.example.com.
;
1  IN  PTR  dns1.example.com.
2  IN  PTR  dns2.example.com.
;
5  IN  PTR  server1.example.com.
6  IN  PTR  server2.example.com.
;
3  IN  PTR  ftp.example.com.
4  IN  PTR  ftp.example.com.
```

In this example, `IP` addresses `10.0.1.1` through `10.0.1.6` are pointed to the corresponding fully qualified domain name.

This zone file would be called into service with a **zone** statement in the **/etc/named.conf** file similar to the following:

```
zone "1.0.10.in-addr.arpa" IN {
  type master;
  file "example.com.rr.zone";
  allow-update { none; };
};
```

There is very little difference between this example and a standard **zone** statement, except for the zone name. Note that a reverse name resolution zone requires the first three blocks of the `IP` address reversed followed by **.in-addr.arpa**. This allows the single block of `IP` numbers used in the reverse name resolution zone file to be associated with the zone.

10.2.4. Using the rndc Utility

The **rndc** utility is a command-line tool that allows you to administer the `named` service, both locally and from a remote machine. Its usage is as follows:

```
rndc [option...] command [command-option]
```

10.2.4.1. Configuring the Utility

To prevent unauthorized access to the service, `named` must be configured to listen on the selected port (**953** by default), and an identical key must be used by both the service and the **rndc** utility.

Table 10.7. Relevant files

Path	Description
/etc/named.conf	The default configuration file for the `named` service.
/etc/rndc.conf	The default configuration file for the **rndc** utility.
/etc/rndc.key	The default key location.

The **rndc** configuration is located in **/etc/rndc.conf**. If the file does not exist, the utility will use the key located in **/etc/rndc.key**, which was generated automatically during the installation process using the **rndc-confgen -a** command.

The `named` service is configured using the **controls** statement in the **/etc/named.conf** configuration file as described in *Section 10.2.2.3, "Other Statement Types"*. Unless this statement is present, only the connections from the loopback address (`127.0.0.1`) will be allowed, and the key located in **/etc/rndc.key** will be used.

For more information on this topic, refer to manual pages and the *BIND 9 Administrator Reference Manual* listed in *Section 10.2.8, "Additional Resources"*.

Set the correct permissions

To prevent unprivileged users from sending control commands to the service, make sure only `root` is allowed to read the **`/etc/rndc.key`** file:

```
~]# chmod o-rwx /etc/rndc.key
```

10.2.4.2. Checking the Service Status

To check the current status of the `named` service, use the following command:

```
~]# rndc status
version: 9.7.0-P2-RedHat-9.7.0-5.P2.el6
CPUs found: 1
worker threads: 1
number of zones: 16
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
recursive clients: 0/0/1000
tcp clients: 0/100
server is up and running
```

10.2.4.3. Reloading the Configuration and Zones

To reload both the configuration file and zones, type the following at a shell prompt:

```
~]# rndc reload
server reload successful
```

This will reload the zones while keeping all previously cached responses, so that you can make changes to the zone files without losing all stored name resolutions.

To reload a single zone, specify its name after the **`reload`** command, for example:

```
~]# rndc reload localhost
zone reload up-to-date
```

Finally, to reload the configuration file and newly added zones only, type:

```
~]# rndc reconfig
```

Modifying zones with dynamic DNS

If you intend to manually modify a zone that uses Dynamic DNS (DDNS), make sure you run the **freeze** command first:

```
~]# rndc freeze localhost
```

Once you are finished, run the **thaw** command to allow the DDNS again and reload the zone:

```
~]# rndc thaw localhost
The zone reload and thaw was successful.
```

10.2.4.4. Updating Zone Keys

To update the DNSSEC keys and sign the zone, use the **sign** command. For example:

```
~]# rndc sign localhost
```

Note that to sign a zone with the above command, the **auto-dnssec** option has to be set to **maintain** in the zone statement. For example:

```
zone "localhost" IN {
  type master;
  file "named.localhost";
  allow-update { none; };
  auto-dnssec maintain;
};
```

10.2.4.5. Enabling the DNSSEC Validation

To enable the DNSSEC validation, issue the following command as `root`:

```
~]# rndc validation on
```

Similarly, to disable this option, type:

```
~]# rndc validation off
```

Refer to the **options** statement described in *Section 10.2.2.2, “Common Statement Types”* for information on how to configure this option in **/etc/named.conf**.

10.2.4.6. Enabling the Query Logging

To enable (or disable in case it is currently enabled) the query logging, issue the following command as `root`:

```
~]# rndc querylog
```

To check the current setting, use the **status** command as described in *Section 10.2.4.2, “Checking the Service Status”*.

10.2.5. Using the dig Utility

The **dig** utility is a command-line tool that allows you to perform DNS lookups and debug a nameserver configuration. Its typical usage is as follows:

```
dig [@server] [option...] name type
```

Refer to *Section 10.2.3.2, “Common Resource Records”* for a list of common values to use for *type*.

10.2.5.1. Looking Up a Nameserver

To look up a nameserver for a particular domain, use the command in the following form:

```
dig name NS
```

In *Example 10.17, “A sample nameserver lookup”*, the **dig** utility is used to display nameservers for `example.com`.

Example 10.17. A sample nameserver lookup

```
~]$ dig example.com NS

; <<>> DiG 9.7.1-P2-RedHat-9.7.1-2.P2.fc13 <<>> example.com NS
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57883
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com.                   IN      NS

;; ANSWER SECTION:
example.com.            99374   IN      NS      a.iana-servers.net.
example.com.            99374   IN      NS      b.iana-servers.net.

;; Query time: 1 msec
;; SERVER: 10.34.255.7#53(10.34.255.7)
;; WHEN: Wed Aug 18 18:04:06 2010
;; MSG SIZE  rcvd: 77
```

10.2.5.2. Looking Up an IP Address

To look up an IP address assigned to a particular domain, use the command in the following form:

```
dig name A
```

In *Example 10.18, “A sample IP address lookup”*, the **dig** utility is used to display the IP address of `example.com`.

Example 10.18. A sample IP address lookup

```
~]$ dig example.com A
```

```
; <<>> DiG 9.7.1-P2-RedHat-9.7.1-2.P2.fc13 <<>> example.com A
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4849
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com.                   IN      A

;; ANSWER SECTION:
example.com.            155606  IN      A       192.0.32.10

;; AUTHORITY SECTION:
example.com.            99175   IN      NS      a.iana-servers.net.
example.com.            99175   IN      NS      b.iana-servers.net.

;; Query time: 1 msec
;; SERVER: 10.34.255.7#53(10.34.255.7)
;; WHEN: Wed Aug 18 18:07:25 2010
;; MSG SIZE  rcvd: 93
```

10.2.5.3. Looking Up a Host Name

To look up a host name for a particular `IP` address, use the command in the following form:

```
dig -x address
```

In *Example 10.19, “A Sample Host Name Lookup”*, the **`dig`** utility is used to display the host name assigned to `192.0.32.10`.

Example 10.19. A Sample Host Name Lookup

```
~]$ dig -x 192.0.32.10

; <<>> DiG 9.7.1-P2-RedHat-9.7.1-2.P2.fc13 <<>> -x 192.0.32.10
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29683
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 6

;; QUESTION SECTION:
;10.32.0.192.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
10.32.0.192.in-addr.arpa. 21600 IN      PTR     www.example.com.

;; AUTHORITY SECTION:
32.0.192.in-addr.arpa.  21600   IN      NS      b.iana-servers.org.
32.0.192.in-addr.arpa.  21600   IN      NS      c.iana-servers.net.
32.0.192.in-addr.arpa.  21600   IN      NS      d.iana-servers.net.
32.0.192.in-addr.arpa.  21600   IN      NS      ns.icann.org.
32.0.192.in-addr.arpa.  21600   IN      NS      a.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net.     13688   IN      A       192.0.34.43
b.iana-servers.org.     5844    IN      A       193.0.0.236
b.iana-servers.org.     5844    IN      AAAA    2001:610:240:2::c100:ec
c.iana-servers.net.     12173   IN      A       139.91.1.10
c.iana-servers.net.     12173   IN      AAAA    2001:648:2c30::1:10
ns.icann.org.           12884   IN      A       192.0.34.126

;; Query time: 156 msec
;; SERVER: 10.34.255.7#53(10.34.255.7)
;; WHEN: Wed Aug 18 18:25:15 2010
;; MSG SIZE  rcvd: 310
```

10.2.6. Advanced Features of BIND

Most BIND implementations only use the `named` service to provide name resolution services or to act as an authority for a particular domain. However, BIND version 9 has a number of advanced features that allow for a more secure and efficient `DNS` service.

Make sure the feature is supported

Before attempting to use advanced features like DNSSEC, TSIG, or IXFR (Incremental Zone Transfer), make sure that the particular feature is supported by all nameservers in the network environment, especially when you use older versions of BIND or non-BIND servers.

All of the features mentioned are discussed in greater detail in the *BIND 9 Administrator Reference Manual* referenced in *Section 10.2.8.1, “Installed Documentation”*.

10.2.6.1. Multiple Views

Optionally, different information can be presented to a client depending on the network a request originates from. This is primarily used to deny sensitive `DNS` entries from clients outside of the local network, while allowing queries from clients inside the local network.

To configure multiple views, add the **`view`** statement to the **`/etc/named.conf`** configuration file. Use the **`match-clients`** option to match `IP` addresses or entire networks and give them special options and zone data.

10.2.6.2. Incremental Zone Transfers (IXFR)

Incremental Zone Transfers (*IXFR*) allow a secondary nameserver to only download the updated portions of a zone modified on a primary nameserver. Compared to the standard transfer process, this makes the notification and update process much more efficient.

Note that IXFR is only available when using dynamic updating to make changes to master zone records. If manually editing zone files to make changes, *Automatic Zone Transfer* (*AXFR*) is used.

10.2.6.3. Transaction SIGnatures (TSIG)

Transaction SIGnatures (TSIG) ensure that a shared secret key exists on both primary and secondary nameservers before allowing a transfer. This strengthens the standard `IP` address-based method of transfer authorization, since attackers would not only need to have access to the `IP` address to transfer the zone, but they would also need to know the secret key.

Since version 9, BIND also supports *TKEY*, which is another shared secret key method of authorizing zone transfers.

Secure the transfer

When communicating over an insecure network, do not rely on `IP` address-based authentication only.

10.2.6.4. DNS Security Extensions (DNSSEC)

Domain Name System Security Extensions (*DNSSEC*) provide origin authentication of `DNS` data, authenticated denial of existence, and data integrity. When a particular domain is marked as secure, the **`SERVFAIL`** response is returned for each resource record that fails the validation.

Note that to debug a DNSSEC-signed domain or a DNSSEC-aware resolver, you can use the **`dig`** utility as described in *Section 10.2.5, "Using the dig Utility"*. Useful options are **`+dnssec`** (requests DNSSEC-related resource records by setting the DNSSEC OK bit), **`+cd`** (tells recursive nameserver not to validate the response), and **`+bufsize=512`** (changes the packet size to 512B to get through some firewalls).

10.2.6.5. Internet Protocol version 6 (IPv6)

Internet Protocol version 6 (*IPv6*) is supported through the use of **`AAAA`** resource records, and the **`listen-on-v6`** directive as described in *Table 10.3, "Commonly Used Configuration Options"*.

10.2.7. Common Mistakes to Avoid

The following is a list of recommendations on how to avoid common mistakes users make when configuring a nameserver:

Use semicolons and curly brackets correctly
: An omitted semicolon or unmatched curly bracket in the **`/etc/named.conf`** file can prevent the `named` service from starting.

Use period (the **`.`** character) correctly
: In zone files, a period at the end of a domain name denotes a fully qualified domain name. If omitted, the `named` service will append the name of the zone or the value of **`$ORIGIN`** to complete it.

Increment the serial number when editing a zone file
: If the serial number is not incremented, the primary nameserver will have the correct, new information, but the secondary nameservers will never be notified of the change, and will not attempt to refresh their data of that zone.

Configure the firewall
: If a firewall is blocking connections from the `named` service to other nameservers, the recommended practice is to change the firewall settings.

Avoid using fixed UDP source ports

Using a fixed `UDP` source port for `DNS` queries is a potential security vulnerability that could allow an attacker to conduct cache-poisoning attacks more easily. To prevent this, by default `DNS` sends from a random ephemeral port. Configure your firewall to allow outgoing queries from a random `UDP` source port. The range **`1024`** to **`65535`** is used by default.

10.2.8. Additional Resources

The following sources of information provide additional resources regarding BIND.

10.2.8.1. Installed Documentation

BIND features a full range of installed documentation covering many different topics, each placed in its own subject directory. For each item below, replace *`version`* with the version of the *bind* package installed on the system:

`/usr/share/doc/bind-`*`version`*`/`
: The main directory containing the most recent documentation. The directory contains the *BIND 9 Administrator Reference Manual* in HTML and PDF formats, which details BIND resource requirements, how to configure different types of nameservers, how to perform load balancing, and other advanced topics.

`/usr/share/doc/bind-`*`version`*`/sample/etc/`
: The directory containing examples of `named` configuration files.

`rndc(8)`
: The manual page for the **`rndc`** name server control utility, containing documentation on its usage.

`named(8)`
: The manual page for the Internet domain name server `named`, containing documentation on assorted arguments that can be used to control the BIND nameserver daemon.

`lwresd(8)`
: The manual page for the lightweight resolver daemon `lwresd`, containing documentation on the daemon and its usage.

`named.conf(5)`
: The manual page with a comprehensive list of options available within the `named` configuration file.

`rndc.conf(5)`
: The manual page with a comprehensive list of options available within the **`rndc`** configuration file.

10.2.8.2. Useful Websites

http://www.isc.org/software/bind
: The home page of the BIND project containing information about current releases as well as a PDF version of the *BIND 9 Administrator Reference Manual*.

10.2.8.3. Related Books

DNS and BIND by Paul Albitz and Cricket Liu; O'Reilly & Associates
: A popular reference that explains both common and esoteric BIND configuration options, and provides strategies for securing a DNS server.

The Concise Guide to DNS and BIND by Nicolai Langfeldt; Que
: Looks at the connection between multiple network services and BIND, with an emphasis on task-oriented, technical topics.

Appendix A. Revision History

Revision 3-2 **Tue June 21 2016** **Stephen Wadeley** *`swadeley@redhat.com`*

Fedora 24 release of the *Networking Guide*.

Revision 3-0 **Mon Nov 02 2015** **Stephen Wadeley** *`swadeley@redhat.com`*

Fedora 23 release of the *Networking Guide*.

Revision 2-3 **Tue May 26 2015** **Stephen Wadeley** *`swadeley@redhat.com`*

Fedora 22 release of the *Networking Guide*.

Revision 2-2 **Mon Mar 30 2015** **Stephen Wadeley** *`swadeley@redhat.com`*

Added "Modifying a Static Ethernet Connection" to the Configure Networking chapter.

Revision 2-1 **Fri Feb 6 2015** **Stephen Wadeley** *`swadeley@redhat.com`*

Added "VLAN over a Bond" to the bonding chapter.

Revision 2-0 **Tues Jan 6 2015** **Stephen Wadeley** *`swadeley@redhat.com`*

Update for Fedora 21.

Revision 1-1.2 **Mon Jan 5 2015** **Stephen Wadeley** *`swadeley@redhat.com`*

General update and corrections for Fedora 20.

Revision 1-1 **Fri Aug 1 2014** **Stephen Wadeley** *`swadeley@redhat.com`*

First version of the Fedora Networking Guide.

Index