Red Hat Enterprise Linux 7

Networking Guide

Configuration and Administration of Networking for Red Hat Enterprise Linux

Last Updated: 2018-05-03
Abstract

The Red Hat Enterprise Linux 7 Networking Guide documents relevant information regarding the configuration and administration of network interfaces, networks and network services in Red Hat Enterprise Linux. It is oriented towards system administrators with a basic understanding of Linux and networking. This book is based on the Red Hat Enterprise Linux 6 Deployment Guide. The chapters related to networking were taken from the Deployment Guide to form the foundation for this book.
# Table of Contents

## PART I. IP NETWORKING

### CHAPTER 1. INTRODUCTION TO RED HAT ENTERPRISE LINUX NETWORKING

1.1. HOW THIS BOOK IS STRUCTURED
1.2. IP NETWORKS VERSUS NON-IP NETWORKS
1.3. INTRODUCTION TO NETWORKMANAGER
1.4. INSTALLING NETWORKMANAGER
1.5. NETWORK CONFIGURATION USING A TEXT USER INTERFACE (NMTUI)
1.6. NETWORK CONFIGURATION USING NETWORKMANAGER'S CLI (NMCLI)
1.7. NETWORK CONFIGURATION USING THE COMMAND-LINE INTERFACE (CLI)
1.8. NETWORKMANAGER AND THE NETWORK SCRIPTS
1.9. NETWORK CONFIGURATION USING SYSCONFIG FILES
1.10. SETTING THE WIRELESS REGULATORY DOMAIN
1.11. CONFIGURING NETCONSOLE
1.12. USING NETWORK KERNEL TUNABLES WITH SYSCTL
1.13. MANAGING DATA USING THE NCAT UTILITY
1.14. ADDITIONAL RESOURCES

### CHAPTER 2. CONFIGURE IP NETWORKING

2.1. STATIC AND DYNAMIC INTERFACE SETTINGS
2.2. EDITING NETWORK CONFIGURATION FILES
2.3. USING NETWORKMANAGER WITH THE GNOME GRAPHICAL USER INTERFACE
2.4. ESTABLISHING A VPN CONNECTION
2.5. ESTABLISHING A MOBILE BROADBAND CONNECTION
2.6. ESTABLISHING A DSL CONNECTION
2.7. CONFIGURING CONNECTION SETTINGS
2.8. ADDITIONAL RESOURCES

### CHAPTER 3. CONFIGURE HOST NAMES

3.1. UNDERSTANDING HOST NAMES
3.2. CONFIGURING HOST NAMES USING TEXT USER INTERFACE, NMTUI
3.3. CONFIGURING HOST NAMES USING HOSTNAMECTL
3.4. CONFIGURING HOST NAMES USING NMCLI
3.5. ADDITIONAL RESOURCES

### CHAPTER 4. CONFIGURE NETWORK BONDING

4.1. UNDERSTANDING THE DEFAULT BEHAVIOR OF MASTER AND SLAVE INTERFACES
4.2. CONFIGURE BONDING USING THE TEXT USER INTERFACE, NMTUI
4.3. NETWORK BONDING USING THE NETWORKMANAGER COMMAND LINE TOOL, NMCLI
4.4. USING THE COMMAND LINE INTERFACE (CLI)
4.5. USING CHANNEL BONDING
4.6. CREATING A BOND CONNECTION USING A GUI
4.7. ADDITIONAL RESOURCES

### CHAPTER 5. CONFIGURE NETWORK TEAMING

5.1. UNDERSTANDING NETWORK TEAMING
5.2. UNDERSTANDING THE DEFAULT BEHAVIOR OF MASTER AND SLAVE INTERFACES
5.3. COMPARISON OF NETWORK TEAMING TO BONDING
5.4. UNDERSTANDING THE NETWORK TEAMING DAEMON AND THE "RUNNERS"
5.5. INSTALL THE NETWORK TEAMING DAEMON
5.6. CONVERTING A BOND TO A TEAM
5.7. SELECTING INTERFACES TO USE AS PORTS FOR A NETWORK TEAM
5.8. SELECTING NETWORK TEAM CONFIGURATION METHODS
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.9</td>
<td>5.9. CONFIGURE A NETWORK TEAM USING THE TEXT USER INTERFACE, NMTUI</td>
<td>110</td>
</tr>
<tr>
<td>5.10</td>
<td>5.10. CONFIGURE A NETWORK TEAM USING THE COMMAND LINE</td>
<td>115</td>
</tr>
<tr>
<td>5.11</td>
<td>5.11. CONTROLLING TEAMD WITH TEAMDCTL</td>
<td>123</td>
</tr>
<tr>
<td>5.12</td>
<td>5.12. CONFIGURE TEAMD RUNNERS</td>
<td>125</td>
</tr>
<tr>
<td>5.13</td>
<td>5.13. CREATING A NETWORK TEAM USING A GUI</td>
<td>132</td>
</tr>
<tr>
<td>5.14</td>
<td>5.14. ADDITIONAL RESOURCES</td>
<td>135</td>
</tr>
<tr>
<td>6.1</td>
<td>6.1. CONFIGURE NETWORK BRIDGING USING THE TEXT USER INTERFACE, NMTUI</td>
<td>137</td>
</tr>
<tr>
<td>6.2</td>
<td>6.2. USING THE NETWORKMANAGER COMMAND LINE TOOL, NMCLI</td>
<td>137</td>
</tr>
<tr>
<td>6.3</td>
<td>6.3. USING THE COMMAND LINE INTERFACE (CLI)</td>
<td>140</td>
</tr>
<tr>
<td>6.4</td>
<td>6.4. CONFIGURE NETWORK BRIDGING USING A GUI</td>
<td>142</td>
</tr>
<tr>
<td>6.5</td>
<td>6.5. ETHERNET BRIDGE CONFIGURATION USING IPROUTE</td>
<td>146</td>
</tr>
<tr>
<td>6.6</td>
<td>6.6. ADDITIONAL RESOURCES</td>
<td>152</td>
</tr>
<tr>
<td>6.7</td>
<td>6.7. VLAN ON BOND AND BRIDGE USING IP COMMANDS</td>
<td>163</td>
</tr>
<tr>
<td>6.8</td>
<td>6.8. CONFIGURING VLAN SWITCHPORT MODE</td>
<td>165</td>
</tr>
<tr>
<td>6.9</td>
<td>6.9. ADDITIONAL RESOURCES</td>
<td>165</td>
</tr>
<tr>
<td>7.1</td>
<td>7.1. SELECTING VLAN INTERFACE CONFIGURATION METHODS</td>
<td>154</td>
</tr>
<tr>
<td>7.2</td>
<td>7.2. CONFIGURE 802.1Q VLAN TAGGING USING THE TEXT USER INTERFACE, NMTUI</td>
<td>154</td>
</tr>
<tr>
<td>7.3</td>
<td>7.3. CONFIGURE 802.1Q VLAN TAGGING USING THE COMMAND LINE TOOL, NMCLI</td>
<td>155</td>
</tr>
<tr>
<td>7.4</td>
<td>7.4. CONFIGURE 802.1Q VLAN TAGGING USING THE COMMAND LINE</td>
<td>156</td>
</tr>
<tr>
<td>7.5</td>
<td>7.5. CONFIGURE 802.1Q VLAN TAGGING USING A GUI</td>
<td>159</td>
</tr>
<tr>
<td>7.6</td>
<td>7.6. VLAN ON BOND AND BRIDGE USING IP COMMANDS</td>
<td>161</td>
</tr>
<tr>
<td>7.7</td>
<td>7.7. VLAN ON BOND AND BRIDGE USING THE NETWORKMANAGER COMMAND LINE TOOL, NMCLI</td>
<td>163</td>
</tr>
<tr>
<td>7.8</td>
<td>7.8. CONFIGURING VLAN SWITCHPORT MODE</td>
<td>165</td>
</tr>
<tr>
<td>7.9</td>
<td>7.9. ADDITIONAL RESOURCES</td>
<td>165</td>
</tr>
<tr>
<td>8.1</td>
<td>8.1. NAMING SCHEMES HIERARCHY</td>
<td>166</td>
</tr>
<tr>
<td>8.2</td>
<td>8.2. UNDERSTANDING THE DEVICE RENAMING PROCEDURE</td>
<td>166</td>
</tr>
<tr>
<td>8.3</td>
<td>8.3. UNDERSTANDING THE PREDICTABLE NETWORK INTERFACE DEVICE NAMES</td>
<td>167</td>
</tr>
<tr>
<td>8.4</td>
<td>8.4. NAMING SCHEME FOR NETWORK DEVICES AVAILABLE FOR LINUX ON SYSTEM Z</td>
<td>168</td>
</tr>
<tr>
<td>8.5</td>
<td>8.5. NAMING SCHEME FOR VLAN INTERFACES</td>
<td>168</td>
</tr>
<tr>
<td>8.6</td>
<td>8.6. CONSISTENT NETWORK DEVICE NAMING USING BIOSDEVNAME</td>
<td>169</td>
</tr>
<tr>
<td>8.7</td>
<td>8.7. NOTES FOR ADMINISTRATORS</td>
<td>170</td>
</tr>
<tr>
<td>8.8</td>
<td>8.8. CONTROLLING THE SELECTION OF NETWORK DEVICE NAMES</td>
<td>170</td>
</tr>
<tr>
<td>8.9</td>
<td>8.9. DISABLING CONSISTENT NETWORK DEVICE NAMING</td>
<td>171</td>
</tr>
<tr>
<td>8.10</td>
<td>8.10. TROUBLESHOOTING NETWORK DEVICE NAMING</td>
<td>172</td>
</tr>
<tr>
<td>8.11</td>
<td>8.11. ADDITIONAL RESOURCES</td>
<td>174</td>
</tr>
<tr>
<td>9.1</td>
<td>9.1. UNDERSTANDING INFINIBAND AND RDMA TECHNOLOGIES</td>
<td>176</td>
</tr>
<tr>
<td>9.2</td>
<td>9.2. TRANSFERRING DATA USING ROCE</td>
<td>176</td>
</tr>
<tr>
<td>9.3</td>
<td>9.3. CONFIGURING SOFT-ROCE</td>
<td>177</td>
</tr>
<tr>
<td>9.4</td>
<td>9.4. INFINIBAND AND RDMA RELATED SOFTWARE PACKAGES</td>
<td>179</td>
</tr>
<tr>
<td>9.5</td>
<td>9.5. CONFIGURING THE BASE RDMA SUBSYSTEM</td>
<td>181</td>
</tr>
<tr>
<td>9.6</td>
<td>9.6. CONFIGURING THE SUBNET MANAGER</td>
<td>182</td>
</tr>
<tr>
<td>9.7</td>
<td>9.7. TESTING EARLY INFINIBAND RDMA OPERATION</td>
<td>189</td>
</tr>
<tr>
<td>9.8</td>
<td>9.8. CONFIGURING IPOIB</td>
<td>192</td>
</tr>
<tr>
<td>9.9</td>
<td>9.9. CONFIGURE INFINIBAND USING THE TEXT USER INTERFACE, NMTUI</td>
<td>195</td>
</tr>
<tr>
<td>9.10</td>
<td>9.10. CONFIGURE IPOIB USING THE COMMAND-LINE TOOL, NMCLI</td>
<td>197</td>
</tr>
<tr>
<td>9.11</td>
<td>9.11. CONFIGURE IPOIB USING THE COMMAND LINE</td>
<td>198</td>
</tr>
<tr>
<td>9.12</td>
<td>9.12. TESTING AN RDMA NETWORK AFTER IPOIB IS CONFIGURED</td>
<td>200</td>
</tr>
</tbody>
</table>

**PART II. INFINIBAND AND RDMA NETWORKING** ................................. 175

**CHAPTER 9. CONFIGURE INFINIBAND AND RDMA NETWORKS** ................................. 176

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>9.1. UNDERSTANDING INFINIBAND AND RDMA TECHNOLOGIES</td>
<td>176</td>
</tr>
<tr>
<td>9.2</td>
<td>9.2. TRANSFERRING DATA USING ROCE</td>
<td>177</td>
</tr>
<tr>
<td>9.3</td>
<td>9.3. CONFIGURING SOFT-ROCE</td>
<td>179</td>
</tr>
<tr>
<td>9.4</td>
<td>9.4. INFINIBAND AND RDMA RELATED SOFTWARE PACKAGES</td>
<td>181</td>
</tr>
<tr>
<td>9.5</td>
<td>9.5. CONFIGURING THE BASE RDMA SUBSYSTEM</td>
<td>182</td>
</tr>
<tr>
<td>9.6</td>
<td>9.6. CONFIGURING THE SUBNET MANAGER</td>
<td>189</td>
</tr>
<tr>
<td>9.7</td>
<td>9.7. TESTING EARLY INFINIBAND RDMA OPERATION</td>
<td>192</td>
</tr>
<tr>
<td>9.8</td>
<td>9.8. CONFIGURING IPOIB</td>
<td>195</td>
</tr>
<tr>
<td>9.9</td>
<td>9.9. CONFIGURE INFINIBAND USING THE TEXT USER INTERFACE, NMTUI</td>
<td>197</td>
</tr>
<tr>
<td>9.10</td>
<td>9.10. CONFIGURE IPOIB USING THE COMMAND-LINE TOOL, NMCLI</td>
<td>198</td>
</tr>
<tr>
<td>9.11</td>
<td>9.11. CONFIGURE IPOIB USING THE COMMAND LINE</td>
<td>200</td>
</tr>
<tr>
<td>9.12</td>
<td>9.12. TESTING AN RDMA NETWORK AFTER IPOIB IS CONFIGURED</td>
<td>201</td>
</tr>
</tbody>
</table>
This part describes how to configure the network on Red Hat Enterprise Linux.
CHAPTER 1. INTRODUCTION TO RED HAT ENTERPRISE LINUX 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. Red Hat Engineering Content Services 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, 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:

- the text user interface tool, nmtui,
- NetworkManager's command-line tool nmcli,
- other command-line methods and the use of configuration files,
- a graphical user interface (GUI) method, such as the use of nm-connection-editor or control-network to direct NetworkManager.

The command line can be used to issue commands, hence the term command-line interface (CLI) however the command line can also start an editor, to compose or edit configuration files. Therefore the use of ip commands and configuration files, such as ifcfg files, will be documented together.

1.2. IP NETWORKS VERSUS NON-IP NETWORKS

Most modern networks fall into one of two very broad categories: IP based networks. These are all networks that communicate through Internet Protocol addresses, which is the standard for the Internet and for most internal networks today. This generally includes Ethernet, Cable Modems, DSL Modems, dial up modems, wireless networks, VPN connections and more.

Then there are non-IP based networks. These are usually very specific niche networks, but one in particular has grown in usage enough to warrant mention here and that is InfiniBand. Because InfiniBand is not an IP network, many features and configurations normally used on IP networks are not applicable to InfiniBand. Chapter 9, Configure InfiniBand and RDMA Networks in this guide covers the specific requirements of configuring and administrating an InfiniBand network and also the broader class of RDMA capable devices.

IMPORTANT

Red Hat Enterprise Linux does not provide consistent naming when attempting to use the ethX naming convention. For more information, see Section 8.10, “Troubleshooting Network Device Naming”
1.3. INTRODUCTION TO NETWORKMANAGER

In Red Hat Enterprise Linux 7, 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 1.8, “NetworkManager and the Network Scripts” for more information.

Table 1.1. A Summary of Networking Tools and Applications

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

NetworkManager can configure network aliases, IP addresses, static routes, DNS information, and VPN connections, as well as many connection-specific parameters. NetworkManager provides an API through D-Bus which allows applications to query and control network configuration and state.

Finally, NetworkManager now maintains the state of devices after the reboot process and takes over interfaces which are set into managed mode during restart. In addition, NetworkManager can handle devices which are not explicitly set as unmanaged but controlled manually by the user or another network service.

1.4. INSTALLING NETWORKMANAGER

NetworkManager is installed by default on Red Hat Enterprise Linux. If necessary, to ensure that it is, enter the following command as the root user:

```
$ yum install NetworkManager
```

For information on user privileges and gaining privileges, see the Red Hat Enterprise Linux System Administrator's Guide.

1.4.1. 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
```
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 enter the following command as the root user:

```bash
[~]# systemctl start NetworkManager
```

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

```bash
[~]# systemctl enable NetworkManager
```

For more information on starting, stopping and managing services, see the *Red Hat Enterprise Linux System Administrator's Guide*.

### 1.4.2. Interacting with NetworkManager

Users do not interact with the `NetworkManager` system service directly. Instead, users perform network configuration tasks using graphical and command-line user interface tools. The following tools are available in Red Hat Enterprise Linux 7:

1. A simple curses-based text user interface (TUI) for `NetworkManager`, `nmtui`, is available.
2. A command-line tool, `nmcli`, is provided to allow users and scripts to interact with `NetworkManager`. Note that `nmcli` can be used on systems without a GUI such as servers to control all aspects of `NetworkManager`. It is on an equal footing with the GUI tools.
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.
4. 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 `Network` and then press `Enter`. The `Network` settings tool appears.
5. A graphical user interface tool, `nm-connection-editor`, is available for certain tasks not yet handled by `control-center`. To start it, enter `nm-connection-editor` in a terminal:

```bash
[~]$ nm-connection-editor
```

### 1.5. NETWORK CONFIGURATION USING A TEXT USER INTERFACE (NMTUI)

The `NetworkManager` text user interface (TUI) tool, `nmtui`, provides a text interface to configure networking by controlling `NetworkManager`. The tool is contained in the `NetworkManager-tui` package. At time of writing, it is not installed along with `NetworkManager` by default. To install `NetworkManager-tui`, issue the following command as root:

```bash
[~]# yum install NetworkManager-tui
```
If required, for details on how to verify that NetworkManager is running, see Section 1.4.1, “The NetworkManager Daemon”.

To start nmtui, issue a command as follows:

```
~]$ nmtui
```

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

The following commands are available:

- `nmtui edit connection-name`
  
  If no connection name is supplied, the selection menu appears. If the connection name is supplied and correctly identified, the relevant Edit connection screen appears.

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

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

1.6. 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 1.4.1, “The NetworkManager Daemon”.

Examples of using the nmcli tool for each task will be included where possible, before explaining the use of other command-line methods and graphical user interfaces. See Section 2.1.6, “Using the NetworkManager Command Line Tool, nmcli” for an introduction to nmcli, and check the man nmcli-examples(7) page for useful examples. To find out which properties are available for the nmcli c add and nmcli c modify commands, use the man nm-settings(5) page.

1.7. 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 Red Hat Enterprise Linux 7 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.

To bring an interface up:

```
ip link set ifname up
```

To bring an interface down:

```
ip link set ifname down
```

**NOTE**

The `ip link set ifname` command sets a network interface in IFF_UP state and enables it from the kernel's scope. This is different from the `ifup ifname` command for initscripts or NetworkManager's activation state of a device. In fact, NetworkManager always sets an interface up even if it is currently disconnected. Disconnecting the device through the nmcli tool, does not remove the IFF_UP flag. In this way, NetworkManager gets notifications about the carrier state.

The ip utility replaces the `ifconfig` utility because the net-tools package (which provides `ifconfig`) does not support InfiniBand addresses. The command `ip help` prints a usage message. Specific help is available for OBJECTS, for example: `ip link help` and `ip addr help`.

**NOTE**

ip commands given on the command line will not persist after a system restart. Where persistence is required, make use of configuration files (`ifcfg` files) or add the commands to a script.

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

### 1.8. 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

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.

NOTE

The systemctl reload network.service command does not work due to technical limitations of initscripts. To apply a new configuration for the network service, use the restart command:

```bash
~]# systemctl restart network.service
```

This brings down and brings up all the Network Interface Cards (NICs) to load the new configuration. For more information, see the Red Hat Knowledgebase solution Reload and force-reload options for network service.

### 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. The ifup-local file does not exist by default. If required, create it under the /sbin/ directory.

The ifup-local script is readable only by the initscripts and not by NetworkManager. To run a custom script using NetworkManager, create it under the dispatcher.d/ directory. See the section called “Running Dispatcher scripts” for an explanation of the dispatcher scripts.

NOTE

Red Hat does not provide support if a user modifies any files included with the initscripts package or related rpms.

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.

**Running Dispatcher scripts**

NetworkManager provides a way to run additional custom scripts to start or stop services based on the connection status. By default, the `/etc/NetworkManager/dispatcher.d/` directory exists and NetworkManager runs scripts there, in alphabetical order. Each script must be an executable file owned by root and must have write permission only for the file owner. For more information about running NetworkManager dispatcher scripts, see the Red Hat Knowledgebase solution How to write a NetworkManager dispatcher script to apply ethtool commands.

### 1.9. 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 Red Hat Enterprise Linux 7 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:

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

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

```bash
~]# 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 Red Hat Enterprise Linux System Administrator's Guide 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:

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

Followed by:
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 1.8, “NetworkManager and the Network Scripts” for an explanation of the network scripts.

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 dial-up 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`, `.rpمورig`, and `.rpmsave`. The best way is not to store backup files anywhere within the `/etc` directory.

### 1.10. Setting the Wireless Regulatory Domain

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

The `setregdomain` utility sets the regulatory domain for your system; it takes no arguments and is normally called through system script such as udev rather than manually by the administrator. If a country code look-up fails, the system administrator can define the `COUNTRY` environment variable in the `/etc/sysconfig/regdomain` file.
See the following man pages for more detailed information about the regulatory domain:

- **setregdomain(1)** man page – set regulatory domain based on country code
- **crda(8)** man page – send to the kernel a wireless regulatory domain for a given ISO or IEC 3166 alpha2
- **regulatory.bin(5)** man page – The Linux wireless regulatory database
- **iw(8)** man page – show or manipulate wireless devices and their configuration

### 1.11. CONFIGURING NETCONSOLE

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

It is necessary to have a *rsyslog* server on the network with the *rsyslogd* daemon listening on the 514/udp port. To configure *rsyslogd* to listen on the 514/udp port and receive messages from the network, uncomment the following lines in the `MODULES` section of `/etc/rsyslog.conf`:

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

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

```
# systemctl restart rsyslog
```

To verify that *rsyslogd* is listening on the 514/udp port, use the following command:

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

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

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

**Configuring a Sending Machine**

In Red Hat Enterprise Linux, *netconsole* is configured using the file `/etc/sysconfig/netconsole`, which is part of the `initscripts` package. This package is installed by default and it also provides the *netconsole* service.

To configure a sending machine, set the value of the `SYSLOGADDR` variable in the `/etc/sysconfig/netconsole` file to match the IP address of the *rsyslogd* server, for example:
SYSLOGADDR=192.168.0.1

Restart the netconsole service so the changes take effect. Then, enable `netconsole.service` so it is running again after next reboot:

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

By default, the rsyslogd server writes the netconsole messages from the client in `/var/log/messages` or in the file specified in `rsyslog.conf`.

**NOTE**

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

```
$UDPServerRun <PORT>
```

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

```
SYSLOGPORT=514
```

For more information about netconsole configuration and troubleshooting tips, see [Netconsole Kernel Documentation](#).

### 1.12. USING NETWORK KERNEL TUNABLES WITH SYSCTL

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

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

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

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

For more details on network kernel tunables using `sysctl`, see the [Using PTP with Multiple Interfaces](#) section in the System Administrator’s Guide.

For more information on network kernel tunables, see the [Network Interface Tunables](#) section in the Kernel Administration Guide.

### 1.13. MANAGING DATA USING THE NCAT UTILITY

The `ncat` networking utility replaces `netcat` in Red Hat Enterprise Linux 7. `ncat` is a reliable back-end tool that provides network connectivity to other applications and users. It reads and writes data across the network from the command line, and uses Transmission Control Protocol (TCP), User Datagram
Protocol (UDP), Stream Control Transmission Protocol (SCTP) or Unix sockets for communication. `ncat` can deal with both IPv4 and IPv6, open connections, send packets, perform port scanning, and supports higher-level features such as SSL, and connection broker.

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

**Installing ncat**

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

```
~# yum install ncat
```

**Brief Selection of ncat Use Cases**

**Example 1.1. Enabling Communication between a Client and a Server**

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

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

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

   ```
   ~$ ncat 10.0.11.60 8080
   ```

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

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

   **NOTE**

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

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

**Example 1.2. Sending Files**

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

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

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

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

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

   After the file is transferred, the connection closes automatically.
NOTE
You can transfer a file in the other direction as well:

-~$ ncat -l 8080 < inputfile

-~$ ncat -l 10.0.11.60 8080 > outputfile

Example 1.3. Creating an HTTP proxy server
To create an HTTP proxy server on localhost port 8080:

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

Example 1.4. Port Scanning
To view which ports are open, use the --z option and specify a range of ports to scan:

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

Example 1.5. Setting up Secure Client-Server Communication Using SSL
Set up SSL on a server:

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

On a client machine:

-~$ ncat --ssl 10.0.11.60 8080

NOTE
To ensure true confidentiality of the SSL connection, the server requires the --ssl-cert and --ssl-key options, and the client requires the --ssl-verify and --ssl-trustfile options. For information on OpenSSL, see the Using OpenSSL section in the Security Guide.

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

1.14. ADDITIONAL RESOURCES
- 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 `ifcfg` configuration files and their directives as understood by the legacy network service.

- `/usr/share/doc/initscripts-version/examples/networking/` – A directory containing example configuration files.
CHAPTER 2. CONFIGURE IP 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 just basic guidance to those who have not already been through this decision-making process. Experienced system administrators will likely have their own set of rules and requirements that may differ from what is discussed here. For more information on automated configuration and management, see the OpenLMI section in the Red Hat Enterprise Linux 7 System Administrators Guide. The Red Hat Enterprise Linux 7 Installation Guide 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 2.1.4, “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 10.1, “Why Use DHCP?” for more information on this subject.

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

2.1.3. Configuring the DHCP Client Behavior

A Dynamic Host Configuration Protocol (DHCP) client requests the dynamic IP address and corresponding configuration information from a DHCP server each time a client connects to the network.
Note that **NetworkManager** calls the **DHCP** client, **dhclient** by default.

**Requesting an IP Address**

When a **DHCP** connection is started, a dhcp client requests an IP address from a **DHCP** server. The time that a dhcp client waits for this request to be completed is 60 seconds by default. You can configure the *ipv4.dhcp-timeout* property using the `nmcli` tool or the **IPV4_DHCP_TIMEOUT** option in the `/etc/sysconfig/network-scripts/ifcfg-<ifname>` file. For example, using `nmcli`:

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

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

- **If ipv4.may-fail** is set to **yes** (default), the state of the connection depends on IPv6 configuration:
  1. If the IPv6 configuration is enabled and successful, the connection is activated, but the IPv4 configuration can never be retried again.
  2. If the IPv6 configuration is disabled or does not get configured, the connection fails.

- **If ipv4.may-fail** is set to **no** the connection is deactivated. In this case:
  1. If the **autoconnect** property of the connection is enabled, **NetworkManager** retries to activate the connection as many times as set in the **autoconnect-retries** property. The default is 4.
  2. If the connection still cannot acquire the dhcp address, auto-activation fails.

  Note that after 5 minutes, the auto-connection process starts again and the dhcp client retries to acquire an address from the dhcp server.

**Requesting a Lease Renewal**

When a dhcp address is acquired and the IP address lease cannot be renewed, the dhcp client is restarted for three times every 2 minutes to try to get a lease from the dhcp server. Each time, it is configured by setting the *ipv4.dhcp-timeout* property in seconds (default is 60) to get the lease. If you get a reply during your attempts, the process stops and you get your lease renewed.

After three attempts failed:

- **If ipv4.may-fail** is set to **yes** (default) and IPv6 is successfully configured, the connection is activated and the dhcp client is restarted again every 2 minutes.

- **If ipv4.may-fail** is set to **no**, the connection is deactivated. In this case, if the connection has the **autoconnect** property enabled, the connection is activated from scratch.

**2.1.3.1. Making DHCPv4 Persistent**

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

```
~]$ nmcli connection modify eth1 ipv4.dhcp-timeout infinity
```
As a result, **NetworkManager** never stops trying to get or renew a lease from a DHCP server until it is successful.

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

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

When an IP address lease expires, the static IP preserves the IP state as configured or partially configured (you can have an IP address, but you are not connected to the Internet), making sure that the dhcp client is restarted every 2 minutes.

### 2.1.4. Selecting Network Configuration Methods

- **To configure an interface using NetworkManager's text user interface tool, `nmtui`, proceed to Section 2.1.5, “Using the Text User Interface, nmtui”**

- **To configure an interface using NetworkManager's command-line tool, `nmcli`, proceed to Section 2.1.6, “Using the NetworkManager Command Line Tool, `nmcli`”**

- **To configure a network interface manually, see Section 2.2, “Editing Network Configuration Files”**.

- **To configure a network using graphical user interface tools**, proceed to Section 2.3, “Using NetworkManager with the GNOME Graphical User Interface”

#### 2.1.5. Using the Text User Interface, nmtui

The text user interface tool `nmtui` can be used to configure an interface in a terminal window. Issue the following command to start the tool:

```
~$ nmtui
```

The text user interface appears. Any invalid command prints a usage message.

![NetworkManager TUI](image-url)

*Figure 2.1. The NetworkManager Text User Interface starting menu*
To navigate, use the arrow keys or press Tab to step forwards and press Shift+Tab to step back through the options. Press Enter to select an option. The Space bar toggles the status of a check box.

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

1. Select the Activate a connection menu entry.

   Figure 2.2. Activate a Connection

2. Select the modified connection. On the right, click the Deactivate button.
3. Choose the connection again and click the **Activate** button.
Figure 2.4. Reactivate the Modified Connection

See Section 1.5, “Network Configuration Using a Text User Interface (nmtui)” for information on installing nmtui.

2.1.6. Using the NetworkManager Command Line Tool, nmcli

The nmcli (NetworkManager Command Line Interface) command-line utility is used for controlling NetworkManager and reporting network status. It can be utilized as a replacement for nm-applet or other graphical clients. nmcli is used to create, display, edit, delete, activate, and deactivate network connections, as well as control and display network device status.

The nmcli utility can be used by both users and scripts for controlling NetworkManager:

- For servers, headless machines, and terminals, nmcli can be used to control NetworkManager directly, without GUI, including creating, editing, starting and stopping network connections and viewing network status.

- For scripts, nmcli supports a terse output format which is better suited for script processing. It is a way to integrate network configuration instead of managing network connections manually.

The basic format of a nmcli command is as follows:
**nmcli OPTIONS OBJECT { COMMAND | help }**

where OBJECT can be one of the following options: `general`, `networking`, `radio`, `connection`, `device`, `agent`, and `monitor`. You can use any prefix of these options in your commands. For example: `nmcli con help`.

Some of useful OPTIONS to get started are:

- **-t, terse**
  
  This mode is designed and suitable for computer (script) processing.

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

- **-h, help**
  
  Prints help information.

The `nmcli` tool has some built-in context-sensitive help:

- **`nmcli help`**
  
  This command lists the available options and object names to be used in subsequent commands.

- **`nmcli object help`**
  
  This command displays the list of available actions related to a specified object. For example,

  ```
  nmcli c help
  ```

**Brief Selection of nmcli Examples**

The `nmcli-examples(5)` man page has many useful examples. Some of them show:

- the overall status of NetworkManager:

  ```
  nmcli general status
  ```

- the current NetworkManager logging status:

  ```
  nmcli general logging
  ```

- all connections:

  ```
  nmcli connection show
  ```

- only currently active connections, add the `--active` (or `-a`) option as follows:

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

- devices recognized by NetworkManager and their state:
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 ens3
```

NOTE

The `nmcli connection down` command, deactivates a connection from a device without preventing the device from further auto-activation. The `nmcli device disconnect` command, disconnects a device and prevent the device from automatically activating further connections without manual intervention.

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 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
```

Creating and Modifying a Connection Profile
A connection profile contains the connection property information needed to connect to a data source. To create a new profile for NetworkManager, use the following command:

```
nmcli c add {ARGUMENTS}
```

The `nmcli c add` accepts two different types of parameters:

Property names
the names which NetworkManager uses to describe the connection internally. The most important are:

- connection.type

  ```
  nmcli c add connection.type bond
  ```

- connection.interface-name

  ```
  nmcli c add connection.interface-name eth0
  ```

- connection.id

  ```
  nmcli c add connection.id "My Connection"
  ```

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

**Aliases names**

the human-readable names which are translated to properties internally. The most common are:

- type (the connection.type property)

  ```
  nmcli c add type bond
  ```

- ifname (the connection.interface-name property)

  ```
  nmcli c add ifname eth0
  ```

- con-name (the connection.id property)

  ```
  nmcli c add con-name "My Connection"
  ```

In previous versions of `nmcli`, to create a connection required using the aliases. For example, `ifname eth0` and `con-name My Connection`. A command in the following format could be used:

```
nmcli c add type ethernet ifname eth0 con-name "My Connection"
```

In more recent versions, both the property names and the aliases can be used interchangeably. The following examples are all valid and equivalent:

```
nmcli c add type ethernet ifname eth0 con-name "My Connection" ethernet.mtu 1600
```

```
nmcli c add connection.type ethernet ifname eth0 con-name "My Connection" ethernet.mtu 1600
```

```
nmcli c add connection.type ethernet connection.interface-name eth0 connection.id "My Connection" ethernet.mtu 1600
```
The arguments differ according to the connection types. Only the **type** argument is mandatory for all connection types and **ifname** is mandatory for all types except **bond**, **team**, **bridge** and **vlan**.

**type** **type_name**
connection type. For example:

```
$ nmcli c add type bond
```

**ifname** **interface_name**
interface to bind the connection to. For example:

```
$ nmcli c add ifname interface_name type ethernet
```

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

```
$ nmcli c modify
```

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

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

**NOTE**

It is preferable to use the **property names**. The **aliases** are used only for compatibility reasons.

In addition, to set the ethernet MTU to 1600, modify the size as follows:

```
$ nmcli c modify "My favorite connection" ethernet.mtu 1600
```

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

```
$ nmcli con up con-name
```

For example:

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

### 2.1.7. Understanding the nmcli Options

Following are some of the important `nmcli` property options. See the comprehensive list in the `nmcli(1)` man page:

```
connection.type
```
A connection type. Allowed values are: adsl, bond, bond-slave, bridge, bridge-slave, bluetooth, cdma, ethernet, gsm, infiniband, olpc-mesh, team, team-slave, vlan, wifi, wimax. Each connection type has type-specific command options. You can see the TYPE_SPECIFIC_OPTIONS list in the nmcli(1) man page. For example:

- A gsm connection requires the access point name specified in an apn.

  ```
  nmcli c add connection.type gsm apn access_point_name
  ```

- A wifi device requires the service set identifier specified in a ssid.

  ```
  nmcli c add connection.type wifi ssid My identifier
  ```

**connection.interface-name**

A device name relevant for the connection.

```
nmcli con add connection.interface-name eth0 type ethernet
```

**connection.id**

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

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

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

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

**id**

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

**uuid**

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

### 2.1.8. Connecting to a Network Using nmcli

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

```
-]$
mcli con show
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>UUID</th>
<th>TYPE</th>
</tr>
</thead>
</table>
### Network Configuration

#### Adding an Ethernet connection

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:

```bash
$ nmcli device status
```

<table>
<thead>
<tr>
<th>DEVICE</th>
<th>TYPE</th>
<th>STATE</th>
<th>CONNECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ens3</td>
<td>ethernet</td>
<td>disconnected</td>
<td>--</td>
</tr>
<tr>
<td>ens9</td>
<td>ethernet</td>
<td>disconnected</td>
<td>--</td>
</tr>
<tr>
<td>lo</td>
<td>loopback</td>
<td>unmanaged</td>
<td>--</td>
</tr>
</tbody>
</table>

To set the device unmanaged by the NetworkManager:

```bash
$ nmcli device set ifname managed no
```

For example, to set eth2 unmanaged:

```bash
$ nmcli device status
```

<table>
<thead>
<tr>
<th>DEVICE</th>
<th>TYPE</th>
<th>STATE</th>
<th>CONNECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>bond0</td>
<td>bond</td>
<td>connected</td>
<td>bond0</td>
</tr>
<tr>
<td>virbr0</td>
<td>bridge</td>
<td>connected</td>
<td>virbr0</td>
</tr>
<tr>
<td>eth1</td>
<td>ethernet</td>
<td>connected</td>
<td>bond-slave-eth1</td>
</tr>
<tr>
<td>eth2</td>
<td>ethernet</td>
<td>connected</td>
<td>bond-slave-eth2</td>
</tr>
<tr>
<td>eth0</td>
<td>ethernet</td>
<td>unmanaged</td>
<td>--</td>
</tr>
</tbody>
</table>

```bash
$ nmcli device set eth2 managed no
```

```bash
$ nmcli device status
```

<table>
<thead>
<tr>
<th>DEVICE</th>
<th>TYPE</th>
<th>STATE</th>
<th>CONNECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>bond0</td>
<td>bond</td>
<td>connected</td>
<td>bond0</td>
</tr>
<tr>
<td>virbr0</td>
<td>bridge</td>
<td>connected</td>
<td>virbr0</td>
</tr>
<tr>
<td>eth1</td>
<td>ethernet</td>
<td>connected</td>
<td>bond-slave-eth1</td>
</tr>
<tr>
<td>eth2</td>
<td>ethernet</td>
<td>unmanaged</td>
<td>--</td>
</tr>
<tr>
<td>eth0</td>
<td>ethernet</td>
<td>unmanaged</td>
<td>--</td>
</tr>
</tbody>
</table>

#### NOTE

When you set the device unmanaged, NetworkManager does not control it. However, the device is still connected.

#### Adding a Dynamic Ethernet Connection

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:

```bash
$ nmcli con add type ethernet con-name my-office ifname ens3
Connection 'my-office' (fb157a65-ad32-47ed-858c-102a48e064a2) 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 1.9, “Network Configuration Using sysconfig Files” for more information on using configuration files.

To open the Ethernet connection, issue a command as follows:

```bash
$ nmcli con up my-office
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/5)
```

Review the status of the devices and connections:

```bash
$ nmcli device status
```

<table>
<thead>
<tr>
<th>DEVICE</th>
<th>TYPE</th>
<th>STATE</th>
<th>CONNECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ens3</td>
<td>ethernet</td>
<td>connected</td>
<td>my-office</td>
</tr>
<tr>
<td>ens9</td>
<td>ethernet</td>
<td>disconnected</td>
<td>--</td>
</tr>
<tr>
<td>lo</td>
<td>loopback</td>
<td>unmanaged</td>
<td></td>
</tr>
</tbody>
</table>

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

```bash
$ 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:

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

There is no `dhcp-client-id` property for **IPv6**; **dhclient** 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:

```bash
$ 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:

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

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

Adding a new '802-3-ethernet' connection

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

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

nmcli> describe ipv4.method

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

nmcli> set ipv4.method auto

nmcli> save

Saving the connection with 'autoconnect=yes'. That might result in an immediate activation of the connection.
Do you still want to save? [yes] yes

Connection 'ens3' (090b61f7-540f-4dd6-bf1f-a905831fc287) successfully saved.

~$ 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:

```bash
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:

```bash
~$ nmcli con add type ethernet con-name test-lab ifname ens9 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:

```bash
~$ nmcli con add type ethernet con-name test-lab ifname ens9 ip4 10.10.10.10/24 gw4 10.10.10.254 ip6 abbe::cafe gw6 2001:db8::1
```

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 1.9, “Network Configuration Using sysconfig Files” for more information on using configuration files.

To set two IPv4 DNS server addresses:

```bash
~$ nmcli con mod test-lab ipv4.dns "8.8.8 8.8.4.4"
```

Note that this will replace any previously set DNS servers. To set two IPv6 DNS server addresses:

```bash
~$ 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:

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

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

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

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

Review the status of the devices and connections:

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

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

~]$ nmcli -p con show test-lab

Connection profile details (test-lab)

collection.id:                          test-lab
connection.uuid:                        05abfd5e-324e-4461-844e-8501ba704773
connection.interface-name:              ens9
connection.type:                        802-3-ethernet
connection.autoconnect:                 yes
connection.timestamp:                   1410428968
connection.read-only:                   no
connection.permissions:
connection.zone:                        --
connection.master:                      --
connection.slave-type:                  --
connection.secondaries:
connection.gateway-ping-timeout:        0

[output truncated]

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

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 ens3

Adding a new '802-3-ethernet' connection

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

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

nmcli> set ipv4.addresses 192.168.122.88/24
Do you also want to set 'ipv4.method' to 'manual'? [yes]: yes

nmcli> save temporary
Saving the connection with 'autoconnect=yes'. That might result in an
immediate activation of the connection.
Do you still want to save? [yes] 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 'ens3' (704a5666-8cbd-4d89-b5f9-fa65a3dbc916) successfully
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
```

<table>
<thead>
<tr>
<th>SSID</th>
<th>MODE</th>
<th>CHAN</th>
<th>RATE</th>
<th>SIGNAL</th>
<th>BARS</th>
<th>SECURITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>FedoraTest</td>
<td>Infra</td>
<td>11</td>
<td>54 MB/s</td>
<td>98</td>
<td>▂▂▂▆▆█</td>
<td>WPA1</td>
</tr>
<tr>
<td>Red Hat Guest</td>
<td>Infra</td>
<td>6</td>
<td>54 MB/s</td>
<td>97</td>
<td>▂▂▂▆▆▆</td>
<td>WPA2</td>
</tr>
<tr>
<td>Red Hat</td>
<td>Infra</td>
<td>6</td>
<td>54 MB/s</td>
<td>77</td>
<td>▂▂▂▆▆▆</td>
<td>WPA2 802.1X</td>
</tr>
<tr>
<td>* Red Hat</td>
<td>Infra</td>
<td>40</td>
<td>54 MB/s</td>
<td>66</td>
<td>▂▂▂▆▆▆</td>
<td>WPA2 802.1X</td>
</tr>
<tr>
<td>VoIP</td>
<td>Infra</td>
<td>1</td>
<td>54 MB/s</td>
<td>32</td>
<td>▂▂▂▆▆▂</td>
<td>WEP</td>
</tr>
<tr>
<td>MyCafe</td>
<td>Infra</td>
<td>11</td>
<td>54 MB/s</td>
<td>39</td>
<td>▂▂▂▆▆▂</td>
<td>WPA2</td>
</tr>
</tbody>
</table>

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
```

See the *Red Hat Enterprise Linux 7 Security Guide* for information on password security.

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

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

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

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

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

To verify the change, issue a command as follows:

```bash
~$ 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.

2.1.9. 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:

```bash
~# 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:

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

```bash
==| 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
FIGURE 2.2. EDITING NETWORK CONFIGURATION FILES

2.2.1. 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 open 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
For example, to configure an interface with static network settings using `ifcfg` files, for an interface with the name `eth0`, create a file with the name `ifcfg-eth0` in the `/etc/sysconfig/network-scripts/` directory, that contains:

- For IPv4 configuration

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

- For IPv6 configuration

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

You do not need to specify the network or broadcast address as this is calculated automatically by `ipcalc`.

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

In Red Hat Enterprise Linux 7, the naming convention for network interfaces has been changed, as explained in Chapter 8, Consistent Network Device Naming. Specifying the hardware or MAC address using HWADDR directive can influence the device naming procedure.

Dynamic Network Settings

For example, to configure an interface with dynamic network settings using ifcfg files, for an interface with the name em1, create a file with the name ifcfg-em1 in the /etc/sysconfig/network-scripts/ directory, that contains:

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

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 send a different fully qualified domain name (FQDN) to the DHCP server, add the following line to the ifcfg file:

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

NOTE

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

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

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

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

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

IMPORTANT

In order to apply the configuration, you need to enter the nmcli c reload command.
2.2.2. Configuring the Network Settings from the Kernel Command-line

When connecting to the root file system on an iSCSI target from an interface, the network settings are not configured on the installed system. To work around this problem:

1. Install the **dracut** utility. For information on using **dracut**, see *Red Hat Enterprise Linux System Administrator's Guide*

2. Set the configuration using the **ip** option on the kernel command-line:

   ```
   ip=<client-IP-number>:<server-id>:<gateway-IP-number>:<netmask>:
   <client-hostname>:<interface>:{dhcp|dhpc6|auto6|on|any|none|off}
   ``

   - **dhcp**: DHCP configuration
   - **dhpc6**: DHCP IPv6 configuration
   - **auto6**: automatic IPv6 configuration
   - **on, any**: any protocol available in the kernel (default)
   - **none, off**: no autoconfiguration, which means static network configuration

   For example:
   ```
   ip=192.168.180.120:192.168.180.100:192.168.180.1:255.255.255.0::eth0
   :off
   ```

3. Set the name server configuration:

   ```
   nameserver=srv1 [nameserver=srv2 [nameserver=srv3 [...]]]
   ```

   The **dracut** utility sets up a network connection and generates new **ifcfg** files that can be copied to the `/etc/sysconfig/network-scripts/` file.

2.2.3. 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:

```bash
-]# 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
```
Further examples and command options can be found in the \texttt{ip-address(8)} manual page.

### Configuring Multiple Addresses Using \texttt{ip} Commands

As the \texttt{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 \texttt{ip} command to assign an address can be repeated multiple times in order to assign multiple address. For example:

\begin{verbatim}
~]$ 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 \texttt{ip} utility are documented in the \texttt{ip(8)} manual page.

\textbf{NOTE}

\texttt{ip} commands given on the command line will not persist after a system restart.

### 2.2.4. 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 Red Hat Enterprise Linux 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.

\textbf{NOTE}

To expand your expertise, you might also be interested in the \texttt{Red Hat System Administration I (RH124)} training course.

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

\begin{verbatim}
ip route [ add | del | change | append | replace ] destination-address
\end{verbatim}

See the \texttt{ip-route(8)} man page for more details on the options and formats.

Use the \texttt{ip route} command without options to display the \texttt{IP} routing table. For example:

\begin{verbatim}
~]$ ip route
default via 192.168.122.1 dev ens9 proto static metric 1024
\end{verbatim}
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 through 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 Red Hat Enterprise Linux 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.

### 2.2.5. Configuring Static Routes in ifcfg files

Static routes set using `ip` commands at the command prompt 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 2.2.5.1, “Static Routes Using the IP Command Arguments Format” and the Network/Netmask format as explained in Section 2.2.5.2, “Network/Netmask Directives Format”.

---

192.168.122.0/24 dev ens9 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
### 2.2.5.1. 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 through 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.

For IPv6 configuration, an example of a `route6-interface` file in `ip route` format:

```
2001:db8:1::/48 via 2001:db8::1 metric 2048
2001:db8:2::/48
```

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.

The `ip route` format can be used to specify a source address. For example:

```
10.10.10.0/24 via 192.168.0.10 src 192.168.0.2
```
To define an existing policy-based routing configuration, which specifies multiple routing tables, see Section 2.2.6, “Understanding Policy-routing”.

**IMPORTANT**

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.

### 2.2.5.2. 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:

```bash
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:

```bash
ADDRESS0=10.10.10.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.0.1
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.

By default, forwarding packets from one interface to another, or out of the same interface, is disabled for security reasons. This prevents the system acting as a router for external traffic. If you need the system to route external traffic, such as when sharing a connection or configuring a VPN server, you will need to enable IP forwarding. See *Enabling Packet Forwarding* in the Red Hat Enterprise Linux 7 Security Guide for more details.

### 2.2.6. Understanding Policy-routing

**Policy-routing** also known as source-routing, is a mechanism for more flexible routing configurations. Routing decisions are commonly made based on the destination IP address of a package. **Policy-routing** allows more flexibility to select routes based on other routing properties, such as source IP address, source port, protocol type. Routing tables stores route information about
networks. They are identified by either numeric values or names, which can be configured in the
/etc/iproute2/rt_tables file. The default table is identified with 254. Using policy-routing, you also need rules. Rules are used to select a routing table, based on certain properties of packets.

For initscripts, the routing table is a property of the route that can be configured through the table argument. The ip route format can be used to define an existing policy-based routing configuration, which specifies multiple routing tables:

```
10.10.10.0/24 via 192.168.0.10 table 1
10.10.10.0/24 via 192.168.0.10 table 2
```

To specify routing rules in initscripts, edit them to the /etc/sysconfig/network-scripts/rule-eth0 file for IPv4 or to the /etc/sysconfig/network-scripts/rule6-eth0 file for IPv6.

NetworkManager supports policy-routing, but rules are not supported yet. The rules must be configured by the user running a custom script. For each manual static route, a routing table can be selected:

- **ipv4.route-table** for IPv4

  and


By setting routes to a particular table, all routes from DHCP, autoconf6, DHCP6 are placed in that specific table. In addition, all routes for subnets that have already configured addresses, are placed in the corresponding routing table. For example, if you configure the 192.168.1.10/24 address, the 192.168.1.0/24 subnet is contained in ipv4.route-table.

For more details about policy-routing rules, see the ip-rule(8) man page. For routing tables, see the ip-route(8) man page.

2.2.7. Configuring a VPN

IPsec, provided by Libreswan, is the preferred method for creating a VPN in Red Hat Enterprise Linux 7. Libreswan is an open-source, user-space IPsec implementation for VPN. Configuring an IPsec VPN using the command line is documented in the Red Hat Enterprise Linux 7 Security Guide.

2.3. USING NETWORKMANAGER WITH THE Gnome Graphical User Interface

In Red Hat Enterprise Linux 7, NetworkManager does not have its own graphical user interface (GUI). The network connection icon on the top right of the desktop is provided as part of the GNOME Shell and the Network settings configuration tool is provided as part of the new GNOME control-center GUI. The nm-connection-editor GUI applies the functionality which is not provided the GNOME control-center such as configuring bonds and teaming connections.

2.3.1. Connecting to a Network Using a GUI

There are two ways to access the Network settings window of the control-center application:
Press the Super key to enter the Activities Overview, type control network, and then press Enter. The Network settings tool appears. Proceed to Section 2.3.2, “Configuring New and Editing Existing Connections”.

Click on the GNOME Shell network connection icon in the top right-hand corner of the screen to open its menu.

![Network Utility in GNOME Shell](image)

**Figure 2.5. The Network utility being selected in GNOME**

When you click on the GNOME Shell network connection icon, you are presented with:

- a list of categorized networks you are currently connected to (such as Wired and Wi-Fi);
- a list of all Available Networks that NetworkManager has detected;
- options for connecting to any configured Virtual Private Networks (VPNs); and,
- an option for selecting the Network Settings menu entry.

If you are connected to a network, this is indicated by a black bullet on the left of the connection name.

Click Network Settings. The Network settings tool appears. Proceed to Section 2.3.2, “Configuring New and Editing Existing Connections”.

**2.3.2. 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.

**Figure 2.6. 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 2.3.2.1, “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.

**Figure 2.7. Configure Networks Using the Network Connection Details Window**
To apply changes after a connection modification, you can click the button from ON to OFF to deactivate and set it to ON again to reactivate the device. See Section 2.3.2, “Configuring New and Editing Existing Connections” for more details.

2.3.2.1. 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 2.4, “Establishing a VPN Connection”;

- **Bond connections**, click the **Bond** entry and proceed to Section 4.6.1, “Establishing a Bond Connection”;

- **Bridge connections**, click the **Bridge** entry and proceed to Section 6.4.1, “Establishing a Bridge Connection with a GUI”;

- **VLAN connections**, click the **VLAN** entry and proceed to Section 7.5.1, “Establishing a VLAN Connection”; or,

- **Team connections**, click the **Team** entry and proceed to Section 5.13, “Creating a Network Team Using a GUI”.

2.3.3. 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 connection icon’s menu to cause it to connect.

2.3.4. Common Configuration Options in nm-connection-editor
If you are using the `nm-connection-editor` utility, there are five configuration options which are common to the most connection types (ethernet, wifi, mobile broadband, DSL):

1. Enter `nm-connection-editor` in a terminal:
   ```
   ~
   
   [~]$ nm-connection-editor
   ```

2. Click the Add button. The **Choose a Connection Type** appears:

   ![Choose a Connection Type]

   **Choose a Connection Type**

   Select the type of connection you wish to create.

   If you are creating a VPN, and the VPN connection you wish to create does not appear in the list, you may not have the correct VPN plugin installed.

   ![Ethernet Drop-down]

   **Figure 2.8. Choose a connection type**

   Alternatively for an existing connection type, click the Edit button from the Network Connections dialog.

3. Select the General tab in the Editing dialog:
Figure 2.9. Configuration options in nm-connection-editor

- **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 2.3.3, “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 2.3.5, “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. See the *Red Hat Enterprise Linux 7 Security Guide* for more information on firewall zones.

**NOTE**

For the VPN connection type, only three of the above configuration options are available: Connection name, All users may connect to this network and Firewall Zone.

### 2.3.5. 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

Using control-center

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

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

To configure a wired network connection, press the **Super** key to enter the Activities Overview, type **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**.

**Basic Configuration Options**

You can see the following configuration settings in the **Wired** dialog, by selecting the **Identity** menu entry on the left:
Figure 2.10. Basic Configuration options of a Wired Connection

- **Name** – Enter a descriptive name for your network connection. This name will be used to list this connection in the menu of the Network window.

- **MAC Address** – Select the MAC address of the interface this profile must be applied to.

- **Cloned Address** – If required, enter a different MAC address to use.

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

- **Firewall Zone** – If required, select a different firewall zone to apply. See the *Red Hat Enterprise Linux 7 Security Guide* 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 2.3.3, “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 2.3.5, “System-wide and Private Connection Profiles” for details.

**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, restart the connection to make NetworkManager apply the changes. If the profile is OFF, set it to ON or select it in the network.
connection icon's menu. See Section 2.3.1, “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 2.7.2, “Configuring 802.1X Security”;

- IPv4 settings for the connection, click the IPv4 Settings tab and proceed to Section 2.7.6, “Configuring IPv4 Settings”; or,

- IPv6 settings for the connection, click the IPv6 Settings tab and proceed to Section 2.7.7, “Configuring IPv6 Settings”.

2.3.7. Configuring a Wi-Fi Connection Using control-center

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

To configure a mobile broadband (such as 3G) connection, see Section 2.5, “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 icon's menu, 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 2.7.3, “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 the section called “Editing a Connection or Creating a New One”

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 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 2.7.3, “Configuring Wi-Fi Security”.

Editing a Connection or Creating a 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 the section called “Connecting to a Hidden Wi-Fi Network” for more information. If the SSID is in range, the procedure is as follows:

1. Press the Super key to enter the Activities Overview, type 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.

Basic Configuration Options for a Wi-Fi Connection
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:
Figure 2.11. Basic Configuration Options for a Wi-Fi Connection

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 the most connection types:

- **Connect automatically** — Select this box if you want NetworkManager to auto-connect to this connection when it is available. See Section 2.3.3, “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 2.3.5, “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 network connection icon’s menu. See Section 2.3.1, “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 2.7.3, “Configuring Wi-Fi Security”;

- **IPv4** settings for the connection, click **IPv4** and proceed to Section 2.7.6, “Configuring IPv4 Settings”; or,

- **IPv6** settings for the connection, click **IPv6** and proceed to Section 2.7.7, “Configuring IPv6 Settings”.

### 2.4. ESTABLISHING A VPN CONNECTION

IPsec, provided by Libreswan, is the preferred method for creating a VPN in Red Hat Enterprise Linux 7. For more information, see Section 2.2.7, “Configuring a VPN”.

The GNOME graphical user interface tool described below requires the NetworkManager-libreswan-gnome package. To install the package, enter the following command as **root**:

```
~$ yum install NetworkManager-libreswan-gnome
```

See *Red Hat Enterprise Linux System Administrator’s Guide* for more information on how to install new packages in Red Hat Enterprise Linux.

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 Using control-center

You can configure a new IPsec VPN connection using NetworkManager by opening the Network window and selecting the plus button below the menu.

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

2. Click the plus button at the bottom of the window.

![Image of Network window](image)

Figure 2.12. Adding a connection

3. For manually configuration, select IPsec based VPN.

![Image of Network Manager](image)

Figure 2.13. Configuring VPN on IPsec mode
4. In the Identity configuration form, you can specify the fields in the General and Advanced sections:

![Add Network Connection Form](image)

**Figure 2.14. General and Advanced sections**

- In the General section, you can specify:

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

  **User name**
  If required, enter the user name used to authenticate with the VPN for the user's identity.

  **User password**
  If required, enter the user name associated with the VPN user's identity for authentication.

  **Group name**
The name of a VPN group configured on the remote gateway. In case it is blank, the IKEv1 Main mode is used instead of the default Aggressive mode.

**Secret**

It is a pre-shared key which is used to initialize the encryption before the user's authentication. If required, enter the password associated with the group name.

- The following configuration settings are available under the Advanced section:

**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.

---

**NOTE**

Configuring an IPsec VPN without using NetworkManager, see Section 2.2.7, “Configuring a VPN”.

**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.

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.

3. Click the Configure button.
4. Select the **Identity** menu entry on the left, and specify the fields in the **General** section:
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 or select it in the network connection icon's menu. See Section 2.3.1, “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 2.7.6, “Configuring IPv4 Settings”.

  or

- IPv6 settings for the connection, click the IPv6 Settings tab and proceed to Section 2.7.7, “Configuring IPv6 Settings”.

2.5. 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 Using nm-connection-editor

1. Enter nm-connection-editor in a terminal:

   ~]$ nm-connection-editor

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 the section called “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. Enter `nm-connection-editor` in a terminal:

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

2. Select the connection you want to edit and click the Edit button. See Section 2.3.4, “Common Configuration Options in nm-connection-editor” for more information.

3. Edit the mobile broadband-specific settings by referring to the section called “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 Save button to save your customized configuration.

Then, to configure:

- Point-to-point settings for the connection, click the PPP Settings tab and proceed to Section 2.7.5, “Configuring PPP (Point-to-Point) Settings”;

- IPv4 settings for the connection, click the IPv4 Settings tab and proceed to Section 2.7.6, “Configuring IPv4 Settings”; or,

- IPv6 settings for the connection, click the IPv6 Settings tab and proceed to Section 2.7.7, “Configuring IPv6 Settings”.

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 Using nm-connection-editor” 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;
- 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.

### 2.6. 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 Using nm-connection-editor

1. Enter `nm-connection-editor` in a terminal:
   ```
   ~]$ nm-connection-editor
   ```

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 window appears.

Procedure 2.8. Editing an Existing DSL Connection

1. Enter `nm-connection-editor` in a terminal:
   ```
   ~]$ nm-connection-editor
   ```

2. Select the connection you want to edit and click the Edit button. See Section 2.3.4, “Common Configuration Options in nm-connection-editor” for more information.

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 Save button to save your customized configuration.

Then, to configure:

- **The MAC address and MTU settings**, click the Wired tab and proceed to the section called “Basic Configuration Options”

- **Point-to-point settings for the connection**, click the PPP Settings tab and proceed to Section 2.7.5, “Configuring PPP (Point-to-Point) Settings”;

- **IPv4 settings for the connection**, click the IPv4 Settings tab and proceed to Section 2.7.6, “Configuring IPv4 Settings”.

Networking Guide 64
2.7. CONFIGURING CONNECTION SETTINGS

This section describes various configurations of the 802.3 link settings and shows how to configure them by using NetworkManager.

2.7.1. Configuring 802.3 Link Settings

You can configure the 802.3 link settings of an Ethernet connection by modifying the following configuration parameters:

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

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

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

**Ignoring link negotiation**

In this case, NetworkManager ignores link configuration for an ethernet connection, keeping the already configuration on the device.

To ignore link negotiation, set the following parameters:

```
802-3-ethernet.auto-negotiate = no
802-3-ethernet.speed = 0
802-3-ethernet.duplex = NULL
```

**IMPORTANT**

If the `auto-negotiate` parameter is set to no, but the `speed` and `duplex` values are not set, that does not mean that auto-negotiation is disabled.

**Enforcing auto-negotiation activation**

In this case, NetworkManager enforces auto-negotiation on a device.

To enforce auto-negotiation activation, set the following options:

```
802-3-ethernet.auto-negotiate = yes
802-3-ethernet.speed = 0
802-3-ethernet.duplex = NULL
```

**Manually setting the link speed and duplex**

In this case, you can manually configure the `speed` and `duplex` settings on the link.

To manually set the `speed` and `duplex` link settings, set the aforementioned parameters as follows:

```
802-3-ethernet.auto-negotiate = no
```
IMPORTANT

Make sure to set both the *speed* and the *duplex* values, otherwise NetworkManager does not update the link configuration.

To configure 802.3 link settings, you can use:

- The `nmcli` tool
- The `nm-connection-editor` utility

**Procedure 2.9. Configuring 802.3 Link Settings by Using the nmcli Tool**

1. Create a new ethernet connection for the `eth0` device.

2. Set the 802.3 link setting to a configuration of your choice. For details, see Section 2.7.1, “Configuring 802.3 Link Settings”

   For example, to manually set the *speed* option 100 *Mbit/s* and *duplex* to *full*:

   ```
   nmcli connection add con-name MyEthernet type ethernet ifname eth0 \
   802-3-ethernet.auto-negotiate no \ 
   802-3-ethernet.speed 100 \ 
   802-3-ethernet.duplex full
   ```

**Procedure 2.10. Configuring 802.3 Link Settings by Using nm-connection-editor**

1. Enter `nm-connection-editor` in a terminal:

   ```
   ~]$ nm-connection-editor
   ```

2. Select the ethernet connection you want to edit and click **Edit**. See Section 2.3.4, “Common Configuration Options in nm-connection-editor” for more information.

3. Select the link negotiation of your choice.
   - **Ignore**: link configuration is skipped (default).
   - **Automatic**: link auto-negotiation is enforced on the device.
   - **Manual**: the *Speed* and *Duplex* options can be specified to enforce the link negotiation.
2.7.2. 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.

Configuring Connection Settings 802.1X Security Using a GUI

You can configure 802.1X security for a wired or wireless connection type by opening the Network window (see Section 2.3.1, “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 2.11, “For a Wired Connection” or Procedure 2.12, “For a Wireless Connection”:

Procedure 2.11. 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 2.7.2.1, “Configuring Transport Layer Security (TLS) Settings”

Procedure 2.12. 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.


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

5. See Section 2.7.2.1, “Configuring Transport Layer Security (TLS) Settings” for descriptions of which extensible authentication protocol (EAP) types correspond to your selection in the Security drop-down menu.

Configuring Connection Settings 802.1X Security Using the nmcli tool

To configure a wireless connection using the nmcli tool, follow the procedure below:

1. Set the authenticated key-mgmt (key management) protocol. It configures the keying mechanism for a secure wifi connection. See the nm-settings(5) man page for more details on properties.

2. Configure the 802-1x authentication settings. For the Transport Layer Security (TLS) authentication, see Section 2.7.2.1, “Configuring Transport Layer Security (TLS) Settings” and Section 2.7.2.2, “Configuring TLS Settings” for descriptions of relevant properties:

Table 2.1. The 802-1x authentication settings

<table>
<thead>
<tr>
<th>802-1x authentication setting</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>802-1x.identity</td>
<td>Identity</td>
</tr>
<tr>
<td>802-1x.ca-cert</td>
<td>CA certificate</td>
</tr>
</tbody>
</table>
For example, to configure WPA2 Enterprise using the EAP-TLS authentication method, apply the following settings:

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

**NOTE**

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

### 2.7.2.1. Configuring Transport Layer Security (TLS) 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 2.7.2.2, "Configuring TLS Settings";
Select FAST for Flexible Authentication via Secure Tunneling and proceed to Section 2.7.2.3, “Configuring FAST Settings”;

Select Tunneled TLS for Tunneled Transport Layer Security, otherwise known as TTLS, or EAP-TTLS and proceed to Section 2.7.2.4, “Configuring Tunneled TLS Settings”;

Select Protected EAP (PEAP) for Protected Extensible Authentication Protocol and proceed to Section 2.7.2.5, “Configuring Protected EAP (PEAP) Settings”.

2.7.2.2. 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.

2.7.2.3. 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.

Username
Enter the user name to be used in the authentication process.
Password
Enter the password to be used in the authentication process.

2.7.2.4. 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.
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.

2.7.2.5. 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
MD5 – Message Digest 5, a cryptographic hash function.
GTC – Generic Token Card.

Username
Enter the user name to be used in the authentication process.
2.7.3. 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 2.7.2.1, “Configuring Transport Layer Security (TLS) 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 2.7.2.1, “Configuring Transport Layer Security (TLS) Settings”

Password

Enter the password to be used in the authentication process.

2.7.4. Using MACsec with wpa_supplicant and NetworkManager

Media Access Control Security (MACsec, IEEE 802.1AE) encrypts and authenticates all traffic in LANs with the GCM-AES-128 algorithm. MACsec can protect not only IP but also Address Resolution Protocol (ARP), Neighbor Discovery (ND), or DHCP. While IPsec operates on the network layer (layer 3) and SSL or TLS on the transport layer (layer 4), MACsec operates in the data link layer (layer 2). Combine MACsec with security protocols for other networking layers to take advantage of different security features that these standards provide.

To enable MACsec with a switch that performs authentication using a pre-shared Connectivity Association Key/CAK Name (CAK/CKN) pair, perform the following steps:

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

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

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

```
ctrl_interface=/var/run/wpa_supplicant
eapol_version=3
```
ap_scan=0  
fast_reauth=1  

network={  
  key_mgmt=NONE  
eapol_flags=0  
macsec_policy=1  
  mka_cak=0011... # 16 bytes hexadecimal  
mka_ckn=2233... # 32 bytes hexadecimal  
}  

Use the values from the previous step to complete the mka_cak and mka_ckn lines in the `wpa_supplicant.conf` configuration file.

See the `wpa_supplicant.conf(5)` man page for more information.

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

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

Instead of creating and editing the `wpa_supplicant.conf` file, Red Hat recommends using the `nmcli` command to configure `wpa_supplicant` equivalently as in the previous steps. The following example assumes that you already have a 16-byte hexadecimal CAK (`$MKA_CAK`) and a 32-byte hexadecimal CKN (`$MKA_CKN`):

```bash
~]# nmcli connection add type macsec  
con-name test-macsec+ ifname macsec0  
connection.autoconnect no  
macsec.parent eth0 macsec.mode psk  
macsec.mka-cak $MKA_CAK  
macsec.mka-cak-flags 0  
macsec.mka-ckn $MKA_CKN

~]# nmcli connection up test-macsec+
```

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

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

### 2.7.5. Configuring PPP (Point-to-Point) Settings

**Authentication Methods**

In most cases, the provider’s PPP servers supports all the allowed authentication methods. If a connection fails, the user should disable support for some methods, depending on the PPP server configuration.

**Use point-to-point encryption (MPPE)**

Microsoft Point-To-Point Encryption protocol ([RFC 3078](https://tools.ietf.org/rfc/rfc3078.txt)).
Allow BSD data compression

Allow Deflate data compression

Use TCP header compression
Compressing TCP/IP Headers for Low-Speed Serial Links (RFC 1144).

Send PPP echo packets
LCP Echo-Request and Echo-Reply Codes for loopback tests (RFC 1661).

NOTE
Since the PPP support in NetworkManager is optional, to configure PPP settings, make sure that the NetworkManager-ppp package is already installed.

2.7.6. 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 2.7.7, “Configuring IPv6 Settings”. If you need to configure static routes, click the Routes button and proceed to Section 2.7.8, “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.

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 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 2.7.8, “Configuring Routes”.

2.7.7. 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.

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 with prefix FE80::/64.
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 2.7.8, “Configuring Routes”.

2.7.8. 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 2.3.1, “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.

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.

2.8. ADDITIONAL RESOURCES
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.

Online Documentation

*Red Hat Enterprise Linux 7 Security Guide*

Describes IPsec based VPN and its configuration. Describes the use of authenticated DNS queries using DNSSEC.

*RFC 1518 — Classless Inter-Domain Routing (CIDR)*

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

*RFC 1918 — Address Allocation for Private Internets*

Describes the range of IPv4 addresses reserved for private use.

*RFC 3330 — Special-Use IPv4 Addresses*

Describes the global and other specialized IPv4 address blocks that have been assigned by the Internet Assigned Numbers Authority (IANA).
CHAPTER 3. 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 lowercase 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 for more information on this issue.

3.2. CONFIGURING HOST NAMES USING TEXT USER INTERFACE, NMTUI

The text user interface tool nmtui can be used to configure a host name in a terminal window. Issue the following command to start the tool:

```
$ nmtui
```

The text user interface appears. Any invalid command prints a usage message.
Figure 3.1. The NetworkManager Text User Interface starting menu

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

See Section 1.5, “Network Configuration Using a Text User Interface (nmtui)” for information on installing nmtui.

The NetworkManager text user interface tool nmtui can be used to query and set the static host name in the /etc/hostname file.

**IMPORTANT**

In Red Hat Enterprise Linux, NetworkManager uses the systemd-hostnamed service to read and write the static host name, which is stored in the /etc/hostname file. Due to this, manual modifications done to the /etc/hostname file are no longer picked up automatically by NetworkManager; you should change the system host name through the hostnamectl utility. Also, the use of the HOSTNAME variable in the /etc/sysconfig/network file is now deprecated.

### 3.3. 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.

#### 3.3.1. View All the Host Names

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

```
$ hostnamectl status
```

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

#### 3.3.2. 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.

### 3.3.3. 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

### 3.3.4. 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.

### 3.3.5. 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.

### 3.4. 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.

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
```

### 3.5. ADDITIONAL RESOURCES

- **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*. 


CHAPTER 4. CONFIGURE NETWORK BONDING

Red Hat Enterprise Linux 7 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. See the Red Hat Knowledgebase article *Why is bonding in not supported with direct connection using crossover cables?* for more information.

**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](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. CONFIGURE BONDING USING THE TEXT USER INTERFACE, NMTUI

The text user interface tool *nmtui* can be used to configure bonding in a terminal window. Issue the following command to start the tool:
The text user interface appears. Any invalid command prints a usage message.

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

1. From the starting menu, select **Edit a connection**. Select **Add**, the **New Connection** screen opens.

2. Select **Bond** and then **Create**; the **Edit connection** screen for the bond will open.
3. At this point slave interfaces will need to be added to the bond; to add these select Add, the New Connection screen opens. Once the type of Connection has been chosen select the Create button.
4. The slave's \textit{Edit Connection} display appears; enter the required slave's device name or MAC address in the \textit{Device} section. If required, enter a clone MAC address to be used as the bond's MAC address by selecting \textit{Show} to the right of the \textit{Ethernet} label. Select the \textit{OK} button to save the slave.

\textbf{NOTE}

If the device is specified without a MAC address the \textit{Device} section will be automatically populated once the \textit{Edit Connection} window is reloaded, but only if it successfully finds the device.
5. The name of the bond slave appears in the Slaves section. Repeat the above steps to add further slave connections.

6. Review and confirm the settings before selecting the OK button.
4.3. NETWORK BONDING USING THE NETWORKMANAGER COMMAND LINE TOOL, NMCLI

**NOTE**

See Section 2.1.6, “Using the NetworkManager Command Line Tool, `nmcli`” for an introduction to `nmcli`.

To create a bond connection with the `nmcli` tool, issue the following command:

```
~]$ nmcli con add type bond ifname mybond0
Connection 'bond-mybond0' (5f739690-47e8-444b-9620-1895316a28ba) successfully added.
```

Note that as no `con-name` was given for the bond, the connection name was derived from the interface name by prepending the type.

*NetworkManager* supports most of the bonding options provided by the kernel. For example:

See Section 4.6.1.1, “Configuring the Bond Tab” for definitions of the bond terms.

See Section 1.5, “Network Configuration Using a Text User Interface (nmtui)” for information on installing `nmtui`.

Figure 4.5. The NetworkManager Text User Interface Completed Bond
To add a slave interface:

1. Create a new connection, see the section called “Creating and Modifying a Connection Profile” for details.

2. Set the master property to the bond interface name, or to the name of the master connection:

   ```
   -]$ nmcli con add type ethernet ifname ens3 master mybond0
   Connection 'bond-slave-ens3' (220f99c6-ee0a-42a1-820e-454cbabc2618) successfully added.
   ```

To add a new slave interface, repeat the previous command with the new interface. For example:

   ```
   -]$ nmcli con add type ethernet ifname ens7 master mybond0
   Connection 'bond-slave-ens7' (ecc24c75-1c89-401f-90c8-9706531e0231) successfully added.
   ```

To activate the slaves, issue a command 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)
   ```

When you activate a slave, the master connection also starts. You can see Section 4.1, “Understanding the Default Behavior of Master and Slave Interfaces” for more information. In this case, it is not necessary to manually activate the master connection.

It is possible to change the active_slave option and the primary option of the bond at runtime, without deactivating the connection. For example to change the active_slave option, issue the following command:

   ```
   -]$ nmcli dev mod bond0 +bond.options "active_slave=ens7"
   Connection successfully reapplied to device 'bond0'.
   ```

or to change the primary option:

   ```
   -]$ nmcli dev mod bond0 +bond.options "primary=ens3"
   Connection successfully reapplied to device 'bond0'.
   ```
NOTE

The active_slave option sets the currently active slave whereas the primary option of the bond specifies the active slave to be automatically selected by kernel when a new slave is added or a failure of the active slave occurs.

4.4. 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.

4.4.1. Check if Bonding Kernel Module is Installed

In Red Hat Enterprise Linux 7, the bonding module is not loaded by default. You can load the module by issuing the following command as root:

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

This activation will not persist across system restarts. See the Red Hat Enterprise Linux System Administrator's Guide 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:

```sh
~$ modinfo bonding
```

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

4.4.2. 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).

**IMPORTANT**

Parameters for the bonding kernel module must be specified as a space-separated list in the BONDING_OPTS="bonding parameters" directive in the ifcfg-bondN 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-bondN 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 4.5, “Using Channel Bonding”.

### 4.4.3. 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.

### 4.4.4. Activating a Channel Bond

To activate a bond, open all the slaves. As root, issue the following commands:

```
~]# ifup ifcfg-eth0
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/7)
```
Note that if editing interface files for interfaces which are currently “up”, set them down first as follows:

```bash
ifdown ethN
```

Then when complete, open all the slaves, which will open the bond (provided it was not set “down”).

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

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

Alternatively, to reload all interfaces:

```bash
~# 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:

```bash
~# 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
```

### 4.4.5. Creating Multiple Bonds

In Red Hat Enterprise Linux, 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-bondN` 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=bondN 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=bond0 and the next two using MASTER=bond1.

4.5. USING CHANNEL BONDING

To enhance performance, adjust available module options to ascertain what combination works best. Pay particular attention to the mimon or arp_interval and the arp_ip_target parameters. See Section 4.5.1, “Bonding Module Directives” for a list of available options and how to quickly determine the best ones for your bonded interface.

4.5.1. 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 4.4.2, “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, open the bond you created by running ifup bondN 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/bondN/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 bondN`. 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-bondN` 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

CHAPTER 4. CONFIGURE NETWORK BONDING
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_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](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 by 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 through 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.
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 through 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-mlb (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_inmilliseconds

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 maintains 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.

### 4.6. 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.

#### 4.6.1. Establishing a Bond Connection

**Procedure 4.1. Adding a New Bond Connection Using nm-connection-editor**

Follow the below steps to create a new bond connection.

1. Enter `nm-connection-editor` in a terminal:

   ```
   ~]$ nm-connection-editor
   ```

2. Click the Add button. The Choose a Connection Type window appears. Select Bond and click Create. 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 4.6.1.1, “Configuring the Bond Tab”** below.

**Procedure 4.2. Editing an Existing Bond Connection**

Follow these steps to edit an existing bond connection.

1. Enter `nm-connection-editor` in a terminal:

   ```
   ~]$ nm-connection-editor
   ```

2. Select the connection you want to edit and click the **Edit** 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 2.3.3, “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 2.3.5, “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. See the Red Hat Enterprise Linux 7 Security Guide for more information on firewall zones.

5. Edit the bond-specific settings by referring to Section 4.6.1.1, “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.

Then, to configure:

- IPv4 settings for the connection, click the IPv4 Settings tab and proceed to Section 2.7.6, “Configuring IPv4 Settings”

  or

- IPv6 settings for the connection, click the IPv6 Settings tab and proceed to Section 2.7.7, “Configuring IPv6 Settings”.

4.6.1.1. Configuring the Bond Tab
If you have already added a new bond connection (see Procedure 4.1, “Adding a New Bond Connection_Using nm-connection-editor” 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 through 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.

### 4.7. ADDITIONAL RESOURCES

**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.

**Online Documentation**

*Red Hat Enterprise Linux System Administrator's Guide*

Explains the use of kernel module capabilities.

[https://access.redhat.com/site/node/28421/Configuring_VLAN_devices_over_a_bonded_interface](https://access.redhat.com/site/node/28421/Configuring_VLAN_devices_over_a_bonded_interface)

A Red Hat Knowledgebase article about Configuring VLAN devices over a bonded interface.
CHAPTER 5. CONFIGURE NETWORK TEAMING

5.1. UNDERSTANDING NETWORK TEAMING

The combining or aggregating of network links to provide a logical link with higher throughput, or to provide redundancy, is known by many names, for example channel bonding, Ethernet bonding, port trunking, channel teaming, NIC teaming, or link aggregation. 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 Red Hat Enterprise Linux 7.

NOTE

Regarding the Mode 4 Link Aggregation Control Protocol (LACP) teaming mode, requires configuring the switch to aggregate the links. For more details, see https://www.kernel.org/doc/Documentation/networking/bonding.txt

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 the libteam library is also available. 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 through teamd, and even teamd is optional, users can write their own application to use libteam.

The teamdctl utility is available to control a running instance of teamd using D-bus. teamdctl 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 utility 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 libteam user-space library 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.

NOTE

In the context of network teaming, the term port is also known as slave. Port is preferred when using teamd directly while slave is used when using NetworkManager to refer to interfaces which create a team.
5.2. 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. See the Red Hat Knowledgebase article *Why is bonding not supported with direct connection using crossover cables?* for more information.

5.3. COMPARISON OF NETWORK TEAMING TO BONDING

Table 5.1. A Comparison of Features in Bonding and Team

<table>
<thead>
<tr>
<th>Feature</th>
<th>Bonding</th>
<th>Team</th>
</tr>
</thead>
<tbody>
<tr>
<td>broadcast Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>round-robin Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>active-backup Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LACP (802.3ad) support</td>
<td>Yes (active only)</td>
<td>Yes</td>
</tr>
<tr>
<td>Hash-based Tx policy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User can set hash function</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Feature</td>
<td>Bonding</td>
<td>Team</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>Tx load-balancing support (TLB)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LACP hash port select</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>load-balancing for LACP support</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ethtool link monitoring</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ARP link monitoring</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NS/NA (IPv6) link monitoring</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>ports up/down delays</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>port priorities and stickiness (&quot;primary&quot; option enhancement)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>separate per-port link monitoring setup</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>multiple link monitoring setup</td>
<td>Limited</td>
<td>Yes</td>
</tr>
<tr>
<td>lockless Tx/Rx path</td>
<td>No (rwlock)</td>
<td>Yes (RCU)</td>
</tr>
<tr>
<td>VLAN support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>user-space runtime control</td>
<td>Limited</td>
<td>Full</td>
</tr>
<tr>
<td>Logic in user-space</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Extensibility</td>
<td>Hard</td>
<td>Easy</td>
</tr>
<tr>
<td>Modular design</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Performance overhead</td>
<td>Low</td>
<td>Very Low</td>
</tr>
<tr>
<td>D-Bus interface</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>multiple device stacking</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>zero config using LLDP</td>
<td>No</td>
<td>(in planning)</td>
</tr>
<tr>
<td>NetworkManager support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
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:

```bash
~]# yum 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 4.5, “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”

5.8. SELECTING NETWORK TEAM CONFIGURATION METHODS

To configure a network team using NetworkManager’s text user interface tool, nmtui, proceed to Section 5.9, “Configure a Network Team Using the Text User Interface, nmtui”.

To create a network team using the command-line tool, nmcli, proceed to Section 5.10.1, “Configure Network Teaming Using nmcli”.

To create a network team using the Team daemon, teamd, proceed to Section 5.10.2, “Creating a Network Team Using teamd”.

To create a network team using configuration files, proceed to Section 5.10.3, “Creating a Network Team Using ifcfg Files”.

To configure a network team using a graphical user interface, see Section 5.13, “Creating a Network Team Using a GUI”.

5.9. CONFIGURE A NETWORK TEAM USING THE TEXT USER INTERFACE, NMTUI

The text user interface tool nmtui can be used to configure teaming in a terminal window. Issue the following command to start the tool:

```
➜ $ nmtui
```

The text user interface appears. Any invalid command prints a usage message.

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

1. From the starting menu, select Edit a connection. Select Add, the New Connection screen opens.
2. Select Team, the Edit connection screen opens.
3. To add port interfaces to the team select Add, the New Connection screen opens. Once the type of Connection has been chosen select the Create button to cause the team's Edit Connection display to appear.
4. Enter the required slave's device name or MAC address in the Device section. If required, enter a clone MAC address to be used as the team's MAC address by selecting Show to the right of the Ethernet label. Select the OK button.

**NOTE**

If the device is specified without a MAC address the Device section will be automatically populated once the Edit Connection window is reloaded, but only if it successfully finds the device.
5. The name of the teamed slave appears in the Slaves section. Repeat the above steps to add further slave connections.

6. If custom port settings are to be applied select the Edit button under the JSON configuration section. This will launch a vim console where changes may be applied. Once finished write the changes from vim and then confirm that the displayed JSON string under JSON configuration matches what is intended.

7. Review and confirm the settings before selecting the OK button.
Figure 5.5. The NetworkManager Text User Interface Configuring a Team Connection menu

See Section 5.12, “Configure teamd Runners” for examples of JSON strings. Note that only the relevant sections from the example strings should be used for a team or port configuration using nmtui. Do not specify the “Device” as part of the JSON string. For example, only the JSON string after “device” but before “port” should be used in the Team JSON configuration field. All JSON strings relevant to a port must only be added in the port configuration field.

See Section 1.5, “Network Configuration Using a Text User Interface (nmtui)” for information on installing nmtui.

5.10. CONFIGURE A NETWORK TEAM USING THE COMMAND LINE

5.10.1. Configure Network Teaming Using nmcli

To view the connections available on the system:


dir$ nmcli connection show

<table>
<thead>
<tr>
<th>NAME</th>
<th>UUID</th>
<th>TYPE</th>
<th>DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>eth1</td>
<td>0e8185a1-f0fd-4802-99fb-bedbb31c689b</td>
<td>802-3-ethernet</td>
<td>--</td>
</tr>
<tr>
<td>eth0</td>
<td>dfe1f57b-419d-4d1c-aaf5-245deab82487</td>
<td>802-3-ethernet</td>
<td>--</td>
</tr>
</tbody>
</table>

To view the devices available on the system:
To create a new team interface, with name ServerA:

```bash
$ nmcli connection add type team ifname 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 1.9, “Network Configuration Using sysconfig Files”_ for more information on using configuration files.

To view the other values assigned:

```bash
$ 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
connection.id:                          team-ServerA
connection.uuid:                        b954c62f-5fdd-4339-97b0-40efac734c50
connection.interface-name:              ServerA
connection.type:                        team
connection.autoconnect:                 yes
...ipv4.method:                           auto
```

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:

```bash
$ 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, enter a command as follows:

```bash
$ nmcli con show
NAME                UUID                                  TYPE
DEVICE              TYPE
team-ServerA        b954c62f-5fdd-4339-97b0-40efac734c50  team
ServerA             --
eth1                0e8185a1-f0fd-4802-99f6-4204-8ff0-6d96a91218a7 802-3-ethernet
eth0                dfe1f57b-419d-4d1c-aaf5-245deab82487 802-3-ethernet
```

---

Networking Guide 116
To change the name assigned to a team, enter 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:

```
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:

```
nmcli con show team-name | grep team.config
```

When the `team.config` property is set, all the other team properties are updated accordingly.

It is also possible a more flexible way of exposing and setting particular team options without modifying directly the corresponding JSON string. You can do this by using the other available team properties to set the related team options one by one to the required values. As a result, the `team.config` property is updated to match the new values.

For example, to set the `team.link-watchers` property which allows to specify one or multiple `link-watchers`, enter a command in the following format:

```
nmcli connection modify team-name team.link-watchers "name=ethtool delay-up=5, name=nsna_ping target-host=target.host"
```

The required `link-watchers` are separated by comma and the attributes which belong to the same `link-watcher` are separated by space.

To set the `team.runner` and the `team.link-watchers` properties, enter a command in the following format:

```
nmcli connection modify team-name team.runner activebackup team.link-watchers "name=ethtool delay-up=5, name=nsna_ping target-host=target.host"
```

This is equivalent to set the `team.config` property to the corresponding JSON string:

```
nmcli connection modify team-name team.config '{"runner": {"name": "activebackup"}, "link_watch": [{"name": "ethtool", "delay_up": 5}, {"name": "nsna_ping", "target_host": "target.host"}]}'
```

To add an interface `eth0` to `Team0`, with the name `Team0-port1`, issue a command as follows:

```
-]$$ nmcli con add type ethernet con-name Team0-port1 ifname eth0 master Team0
```
Similarly, to add another interface, `eth1`, with the name `Team0-port2`, issue a command as follows:

```shell
[~]$ nmcli con add type team-slave con-name Team0-port2 ifname eth1 master Team0
Connection 'Team0-port2' (a89ccff8-8202-411e-8ca6-2953b7db52dd) successfully added.
```

**nmcli** only supports Ethernet ports.

To open a team, the ports must be brought up first as follows:

```shell
[~]$ 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:

```shell
[~]$ 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:ff
```

Alternatively, issue a command to open the team as follows:

```shell
[~]$ nmcli connection up Team0
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/4)
```

See Section 2.1.6, “Using the NetworkManager Command Line Tool, **nmcli**” for an introduction to **nmcli**.

### 5.10.2. Creating a Network Team Using teamd

**NOTE**

Configurations created using teamd are not persistent, and as such it may be necessary to create a team using the steps defined in Section 5.10.1, “Configure Network Teaming Using **nmcli**” or Section 5.10.3, “Creating a Network Team Using ifcfg Files”.

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:

```shell
[~]$ ls /usr/share/doc/teamd-*/*example_configs/
```
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
```
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:

```bash
~# 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):

```bash
~# cd /home/user/teamd_working_configs
```

Then issue a command in the following format:

```bash
~# teamd -g -f activebackup_ethtool_1.conf -d
```

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`:

```bash
~# teamdctl team0 state
```

```bash
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.

### 5.10.3. 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
```
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 open the network team, issue the following command as `root`:

```
~ ]# ifup team0
```

To view the network team, issue the following command:

```
~ ]$ ip link show
```

### 5.10.4. 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.

### 5.10.5. 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
```

### 5.10.6. Configuring Options of a Team Using `teamnl`

Networking Guide
To view or list all currently available options, using the `teamnl` utility, issue the following command as `root`:

```bash
~]# teamnl team0 options
```

To configure a team to use active backup mode, issue the following command as `root`:

```bash
~]# teamnl team0 setoption mode activebackup
```

### 5.10.7. 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`:

```bash
~]# ip addr add 192.168.252.2/24 dev team0
```

### 5.10.8. open an Interface to a Network Team Using iputils

To activate or “open” an interface to a network team, `team0`, using the `ip` utility, issue the following command as `root`:

```bash
~]# ip link set team0 up
```

### 5.10.9. 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`:

```bash
~]# teamnl team0 getoption activeport 0
```

### 5.10.10. 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`:

```bash
~]# teamnl team0 setoption activeport 5
```

To check the change in team port options, issue the following command as `root`:

```bash
~]# 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`:

```bash
~]# 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
```

**IMPORTANT**

If using `teamdctl` directly to enslave a port, the slave port must be set to `down`. Otherwise the `teamdctl team0 port add em1` command will fail.

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 round-robin 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 Round-robin Runner

To configure the round-robin runner, using an editor as `root`, add the following to the team JSON format configuration file:

```
{
```

A very basic configuration for round-robin.

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:

```json
{
    "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.

```json
{
    "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 active-backup 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:

```json
{
    "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:

```json
{
    "device": "team0",
    "runner": {
        "name": "loadbalance",
        "tx_hash": ["eth", "ipv4", "ipv6"]
    }
}
```
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:

```json
{
    "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:

```json
{
    "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. Note that only `ethtool` can be used for link monitoring 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:

```json
"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:

```json
    "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:

```json
    "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:

```json
{
    "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**:

```
~/> teamdctl 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

```json
{
  "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:

```json
"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. If more or less queues are required, 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:

```json
{
  "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. CREATING A NETWORK TEAM USING A GUI

#### 5.13.1. Establishing a Team Connection

You can use `nm-connection-editor` 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 Using nm-connection-editor**

Follow the below steps to add a new team connection.

1. Enter `nm-connection-editor` in a terminal:

   ```
   ~]$ nm-connection-editor
   ```

2. Click the Add button. The Choose a Connection Type window appears. Select Team and click Create. The Editing Team connection 1 window appears.
Figure 5.6. The NetworkManager Graphical User Interface Add a menu

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 slave 1 window appears.
Figure 5.7. The NetworkManager Graphical User Interface Add a Slave Connection

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 5.13.1.1, “Configuring the Team Tab” below.

Procedure 5.2. Editing an Existing Team Connection

Follow the below steps to edit an existing team connection.

1. Enter `nm-connection-editor` in a terminal:

   ```bash
   ~]$ nm-connection-editor
   ```

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

3. Select the General tab.

4. Five settings in the Editing dialog are common to the most 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 2.3.3, “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 2.3.5, “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. See the Red Hat Enterprise Linux 7 Security Guide for more information on firewall zones.

5. Edit the team-specific settings by referring to Section 5.13.1.1, “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.

Then, to configure:

- **IPv4 settings** for the connection, click the **IPv4 Settings** tab and proceed to Section 2.7.6, “Configuring IPv4 Settings”

  or

- **IPv6 settings** for the connection, click the **IPv6 Settings** tab and proceed to Section 2.7.7, “Configuring IPv6 Settings”.

### 5.13.1.1. 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 Using nm-connection-editor” for instructions on how to add a new team.

### 5.14. ADDITIONAL RESOURCES

**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.
- `teamn1(8)` man page – Describes the `teamd` Netlink library.
- `bond2team(1)` man page – Describes a tool to convert bonding options to team.

**Online Documentation**

[http://www.w3schools.com/js/js_json_syntax.asp](http://www.w3schools.com/js/js_json_syntax.asp)

An explanation of JSON syntax.
CHAPTER 6. 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.

6.1. CONFIGURE BRIDGING USING THE TEXT USER INTERFACE, NMTUI

The text user interface tool `nmtui` can be used to configure bridging in a terminal window. Issue the following command to start the tool:

```bash
~ ]$ nmtui
```

The text user interface appears. Any invalid command prints a usage message.

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

1. From the starting menu, select **Edit a connection**. Select **Add**, the **New Connection** screen opens.

![Add a Bridge Connection menu](image)

2. Select **Bridge**, the **Edit connection** screen opens.

![Edit connection screen](image)
3. To add slave interfaces to the bridge select Add, the **New Connection** screen opens. Once the type of Connection has been chosen select the **Create** button to cause the bridge's **Edit Connection** display to appear.

![Diagram of NetworkManager Text User Interface Adding a new Bridge Slave Connection menu](networkmanager_diagram.png)

Figure 6.2. The NetworkManager Text User Interface Adding a new Bridge Slave Connection menu

4. Enter the required slave's device name or MAC address in the **Device** section. If required, enter a clone MAC address to be used as the bridge's MAC address by selecting **Show** to the right of the **Ethernet** label. Select the **OK** button.

**NOTE**

If the device is specified without a MAC address the **Device** section will be automatically populated once the **Edit Connection** window is reloaded, but only if it successfully finds the device.
5. The name of the bridge slave appears in the **Slaves** section. Repeat the above steps to add further slave connections.

6. Review and confirm the settings before selecting the **OK** button.
6.2. USING THE NETWORKMANAGER COMMAND LINE TOOL, NMCLI

To create a bridge, named bridge-br0, issue a command as follows as `root`:

```bash
-][# nmcli con add type bridge ifname br0
Connection 'bridge-br0' (6ad5bba6-98a0-4f20-839d-c997ba7668ad) 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:

```bash
-][$ nmcli con show
NAME        UUID                                  TYPE
DEVICE
bridge-br0  79cf6a3e-0310-4a78-b759-bda1cc3eef8d  bridge        br0
eth0        4d5c449a-a6c5-451c-8206-3c9a4ec88bca  802-3-ethernet eth0
```
Spanning tree protocol (STP) is enabled by default. The values used are from the IEEE 802.1D-1998 standard. To disable STP for this bridge, issue a command as follows as `root`:

```
~# nmcli con modify bridge-br0 bridge.stp no
```

To re-enable 802.1D STP for this bridge, issue a command as follows as `root`:

```
~# nmcli con modify bridge-br0 bridge.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.

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.

To configure a bridge connection to forward group addresses in the range from `01:80:C2:00:00:00` to `01:80:C2:00:00:0F`, change the `group-forward-mask` property. This property is a mask of 16 bits. Each bit corresponds to a group address in the above-mentioned range that must be forwarded. For example:

```
~$ nmcli connection modify bridge-br5 bridge.group-forward-mask 8
```

**IMPORTANT**

The `group-forward-mask` property cannot have any of the 0, 1, 2 bits set to 1 because those addresses are used for Spanning tree protocol (STP), Link Aggregation Control Protocol (LACP) and Ethernet MAC pause frames.

To view the bridge settings, issue the following command:

```
~$ nmcli -f bridge con show bridge-br0
```

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 ethernet ifname eth1 master bridge-br0
Connection 'bridge-slave-eth1' (70ffae80-7428-4d9c-8cbd-2e35de72476e) successfully added.
```
To enslave an existing connection to a bridge, proceed as follows:

1. Change its master and slave-type properties. For example to enslave an existing VLAN connection named `vlan100`:

   ```bash
   ~$ nmcli connection modify vlan100 master bridge-br0 slave-type bridge
   ```

2. Reactivate the connection to apply the changes:

   ```bash
   ~$ nmcli connection up vlan100
   ```

To change a value using interactive mode, issue the following command:

```bash
~$ 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 2.1.6, “Using the NetworkManager Command Line Tool, nmcli” for an introduction to nmcli.

### 6.3. USING THE COMMAND LINE INTERFACE (CLI)

#### 6.3.1. Check if Bridging Kernel Module is Installed

In Red Hat Enterprise Linux 7, 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:

```bash
~# modprobe --first-time bridge
```

```
modprobe: ERROR: could not insert 'bridge': Module already in kernel
```

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

```bash
~$ modinfo bridge
```

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

#### 6.3.2. 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**. 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 15 seconds is not needed if no routing loops are possible.

**Example 6.1. Example ifcfg-br0 Interface Configuration File**

The following is an example of a bridge interface configuration file using a static **IP** address:

```bash
DEVICE=br0
TYPE=Bridge
IPADDR=192.168.1.1
PREFIX=24
BOOTPROTO=None
ONBOOT=yes
DELAY=0
```

To complete the bridge another interface is created, or an existing interface is modified, and pointed to the bridge interface.

**Example 6.2. Example ifcfg-ethX Interface Configuration File**

The following is an example of an 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:

```bash
DEVICE=ethX
TYPE=Ethernet
HWADDR=AA:BB:CC:DD:EE:FF
BOOTPROTO=None
ONBOOT=yes
BRIDGE=br0
```

Optionally specify a name using the **NAME** directive. If no name is specified, the **NetworkManager** plug-in, `ifcfg-rh`, will create a name for the connection profile in the form “**Type Interface**”. In this example, this means the bridge will be named **Bridge br0**. Alternately, if **NAME=bridge-br0** is added to the `ifcfg-br0` file the connection profile will be named **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 its name explicitly matches a different interface configuration file).

The directives are case sensitive.

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, 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.

To open 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.

6.3.3. 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, see Section 4.4.2, “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 `ethX` 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 4.5, “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 open 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.

6.4. 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.

6.4.1. Establishing a Bridge Connection with a GUI

Procedure 6.1. Adding a New Bridge Connection Using nm-connection-editor

Follow the below instructions to create a new bridge connection:

1. Enter nm-connection-editor in a terminal:

   ~]$ nm-connection-editor

2. Click the Add button. The Choose a Connection Type window appears. Select Bridge and click Create. The Editing Bridge connection 1 window appears.
Figure 6.5. Editing Bridge Connection 1

3. Add slave devices by referring to Procedure 6.3, “Adding a Slave Interface to a Bridge” below.

Procedure 6.2. Editing an Existing Bridge Connection

1. Enter `nm-connection-editor` in a terminal:

```bash
~]$ nm-connection-editor
```
2. Select the Bridge connection you want to edit.

3. 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 2.3.3, “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 2.3.5, “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. See the Red Hat Enterprise Linux 7 Security Guide for more information on Firewall Zones.

6.4.1.1. 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 IGMP snooping**
If required, select the check box to enable IGMP snooping on the device.

**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.

Group forward mask
This property is a mask of group addresses that allows group addresses to be forwarded. In most cases, group addresses in the range from 01:80:C2:00:00:00 to 01:80:C2:00:00:0F are not forwarded by the bridge device. This property is a mask of 16 bits, each corresponding to a group address in the above range, that must be forwarded. Note that the Group forward mask property cannot have any of the 0, 1, 2 bits set to 1 because those addresses are used for Spanning tree protocol (STP), Link Aggregation Control Protocol (LACP) and Ethernet MAC pause frames.

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 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.
Figure 6.6. The NetworkManager Graphical User Interface Add a Bridge Connection

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 the section called "Basic Configuration Options," or;
- A Bond slave, click the **Bond** tab and proceed to Section 4.6.1.1, "Configuring the Bond Tab," or;
- A Team slave, click the **Team** tab and proceed to Section 5.13.1.1, "Configuring the Team Tab," or;
- An VLAN slave, click the **VLAN** tab and proceed to Section 7.5.1.1, "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 or select it in the network connection icon's menu. See Section 2.3.1, "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 2.7.6, “Configuring IPv4 Settings”, or;

IPv6 settings for the connection, click the IPv6 Settings tab and proceed to Section 2.7.7, “Configuring IPv6 Settings”.

Once saved the Bridge will appear in the Network settings tool with each slave showing in the display.

![NetworkManager Graphical User Interface with Bridge](image)

Figure 6.8. The NetworkManager Graphical User Interface with Bridge

6.5. ETHERNET BRIDGE CONFIGURATION USING IPROUTE

The iproute package can be used as an alternative to the bridge-utils. It allows to set bridge port options such as priority, cost or state.

To set port options for an interface eth0 enslaved in a bridge device, using the ip utility, issue the following command as root:

```bash
~ ] # ip link set eth0 type bridge_slave option
```

To select the available options, using the ip utility, issue the following command as root:

```bash
~ ] # ip link help bridge_slave
Usage: ... bridge_slave [ state STATE ] [ priority PRIO ] [cost COST ]
          [ guard {on | off} ]
          [ hairpin {on | off} ]
          [ fastleave {on | off} ]
          [ root_block {on | off} ]
          [ learning {on | off} ]
          [ flood {on | off} ]
```

For more details on the port options, see the ip-link(8) man page.
6.6. ADDITIONAL RESOURCES

- **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.
- **ip-link(8) man page** – Description of the bridge port options.
CHAPTER 7. CONFIGURE 802.1Q VLAN TAGGING

To create a VLAN, an interface is created on top of 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 does not need to be an Ethernet interface. An 802.1Q VLAN tagging interface can be created on top of 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 opening the VLAN interface. 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 the switch to be properly configured. For example, ports on Cisco switches must be assigned to one VLAN or be configured as trunk ports to accept tagged packets from multiple VLANs. Some vendor switches allow untagged frames of the native VLAN to be processed by a trunk port. Some devices allow you to enable or disable the native VLAN, other devices have it disabled by default. Consequence of this disparity may result in native VLAN misconfiguration between two different switches, posing a security risk. For example:

  One switch uses native VLAN 1 while the other uses native VLAN 10. If the frames are allowed to pass without the tag being inserted, an attacker is able to jump VLANs - this common network penetration technique is also known as VLAN hopping.

To minimize security risks, configure your interface as follows:

**Switches**

- Unless you need them, disable trunk ports.
- If you need trunk ports, disable native VLAN so that untagged frames are not allowed.

**Red Hat Enterprise Linux server**

- Use the nftables or ebtables utilities to drop untagged frames in ingress filtering.

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.

**NOTE**

Bonding on top of VLAN is not supported by Red Hat. See the Red Hat Knowledgebase article Whether configuring bond on top of VLAN as slave interfaces is a valid configuration? for more information.

7.1. SELECTING VLAN INTERFACE CONFIGURATION METHODS
To configure a VLAN interface using NetworkManager's text user interface tool, nmtui, proceed to Section 7.2, “Configure 802.1Q VLAN tagging Using the Text User Interface, nmtui”

To configure a VLAN interface using NetworkManager's command-line tool, nmcli, proceed to Section 7.3, “Configure 802.1Q VLAN Tagging Using the Command Line Tool, nmcli”

To configure a network interface manually, see Section 7.4, “Configure 802.1Q VLAN Tagging Using the Command Line”.

To configure a network using graphical user interface tools, proceed to Section 7.5, “Configure 802.1Q VLAN Tagging Using a GUI”

### 7.2. CONFIGURE 802.1Q VLAN TAGGING USING THE TEXT USER INTERFACE, NMTUI

The text user interface tool nmtui can be used to configure 802.1Q VLANs in a terminal window. Issue the following command to start the tool:

```
~]$ nmtui
```

The text user interface appears. Any invalid command prints a usage message.

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

From the starting menu, select Edit a connection. Select Add, the New Connection screen opens.

![Figure 7.1. The NetworkManager Text User Interface Add a VLAN Connection menu](image)

To configure a VLAN interface using NetworkManager's text user interface tool, nmtui, proceed to Section 7.2, “Configure 802.1Q VLAN tagging Using the Text User Interface, nmtui”

To configure a VLAN interface using NetworkManager's command-line tool, nmcli, proceed to Section 7.3, “Configure 802.1Q VLAN Tagging Using the Command Line Tool, nmcli”

To configure a network interface manually, see Section 7.4, “Configure 802.1Q VLAN Tagging Using the Command Line”.

To configure a network using graphical user interface tools, proceed to Section 7.5, “Configure 802.1Q VLAN Tagging Using a GUI”
Select VLAN, the Edit connection screen opens. Follow the on-screen prompts to complete the configuration.

![Edit connection](image)

Figure 7.2. The NetworkManager Text User Interface Configuring a VLAN Connection menu

See Section 7.5.1.1, “Configuring the VLAN Tab” for definitions of the VLAN terms.

See Section 1.5, “Network Configuration Using a Text User Interface (nmtui)” for information on installing nmtui.

7.3. CONFIGURE 802.1Q VLAN TAGGING USING THE COMMAND LINE TOOL, NMCLI

To view the available interfaces on the system, issue a command as follows:

```bash
$ nmcli con show
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>UUID</th>
<th>TYPE</th>
<th>DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>System eth1</td>
<td>9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04</td>
<td>802-3-ethernet</td>
<td>eth1</td>
</tr>
<tr>
<td>System eth0</td>
<td>5fb06bd0-0bb0-7ff8-45f1-d6edd65f3eb03</td>
<td>802-3-ethernet</td>
<td>eth0</td>
</tr>
</tbody>
</table>

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 firewallld.

To create an 802.1Q VLAN interface on Ethernet interface eth0, with VLAN interface VLAN10 and ID 10, issue a command as follows:

```bash
$ 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:

```bash
$ 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:

```bash
$ nmcli con add type vlan con-name VLAN20 dev eth0 id 20 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:

```bash
$ nmcli con add type vlan con-name VLAN30 dev eth0 id 30
```

See Section 2.1.8, “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:

```bash
$ nmcli con show
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>UUID</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLAN12</td>
<td>4129a37d-4feb-4be5-ac17-14a193821755</td>
<td>vlan</td>
</tr>
<tr>
<td>eth0.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System eth1</td>
<td>9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04</td>
<td>802-3-ethernet</td>
</tr>
<tr>
<td>System eth0</td>
<td>5fb06bd0-0bb0-7ff8-45f1-d6edd65f3e03</td>
<td>802-3-ethernet</td>
</tr>
<tr>
<td>vlan-VLAN10</td>
<td>1be91581-11c2-461a-b40d-893d42fed4f4</td>
<td>vlan</td>
</tr>
</tbody>
</table>

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

```bash
$ 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
...```

---

CHAPTER 7. CONFIGURE 802.1Q VLAN TAGGING
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 *VLAN1* and ID 13, issue a command as follows:

```
[-]$ nmcli con add type vlan con-name VLAN1 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 vlan-VLAN10
```

To change the MTU, issue a command as follows:

```
[-]$ nmcli connection modify vlan-VLAN10 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 2.1.6, “Using the NetworkManager Command Line Tool, nmcli” for an introduction to nmcli.

### 7.4. CONFIGURE 802.1Q VLAN TAGGING USING THE COMMAND LINE

In Red Hat Enterprise Linux 7, 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.

#### 7.4.1. 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
```

### 7.4.2. 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
```

To use multiple interfaces belonging to multiple VLANs, create locally eth0.1 and eth0.2 with the appropriate VLAN ID on top of a physical interface eth0:

```
~# ip link add link eth0 name eth0.1 type vlan id 1
ip link set dev eth0.1 up
~# ip link add link eth0 name eth0.2 type vlan id 2
ip link set dev eth0.2 up
```
Note that running a network sniffer on a physical device, you can capture the tagged frames reaching the physical device, even if no VLAN device is configured on top of eth0. For example:

```
tcpdump -nnei eth0 -vvv
```

**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 7.4.1, “Setting Up 802.1Q VLAN Tagging Using ifcfg Files”

### 7.5. CONFIGURE 802.1Q VLAN TAGGING USING A GUI

#### 7.5.1. Establishing a VLAN Connection

You can use `nm-connection-editor` to create a VLAN using an existing interface as the parent interface. 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 Using nm-connection-editor**

1. Enter `nm-connection-editor` in a terminal:

   ````
   ~]$ nm-connection-editor
   ```

2. Click the Add button. The Choose a Connection Type window appears. Select VLAN and click Create. 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 7.5.1.1, “Configuring the VLAN Tab”.

Procedure 7.2. Editing an Existing VLAN Connection

Follow these steps to edit an existing VLAN connection.

1. Enter `nm-connection-editor` in a terminal:

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

2. Select the connection you want to edit and click the **Edit** 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. See Section 2.3.3, “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. See Section 2.3.5, “System-wide and Private Connection Profiles” for details.

5. To edit the VLAN-specific settings see Section 7.5.1.1, “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.

Then, to configure:

- IPv4 settings for the connection, click the **IPv4 Settings** tab and proceed to Section 2.7.6, “Configuring IPv4 Settings”.

  or

- IPv6 settings for the connection, click the **IPv6 Settings** tab and proceed to Section 2.7.7, “Configuring IPv6 Settings”.

### 7.5.1.1. Configuring the VLAN Tab

If you have already added a new VLAN connection (see Procedure 7.1, “Adding a New VLAN Connection Using nm-connection-editor” 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.

### 7.6. VLAN ON BOND AND BRIDGE USING IP COMMANDS

To use VLANs over bonds and bridges, proceed as follows:

1. Add a bond device as root:

   ```bash
   # ip link add bond0 type bond
   # ip link set bond0 type bond miimon 100 mode active-backup
   # ip link set em1 down
   # ip link set em1 master bond0
   # ip link set em2 down
   # ip link set em2 master bond0
   # ip link set bond0 up
   ```
2. Set VLAN on the bond device:

   # ip link add link bond0 name bond0.2 type vlan id 2
   # ip link set bond0.2 up

3. Add the bridge device and attach VLAN to it:

   # ip link add br0 type bridge
   # ip link set bond0.2 master br0
   # ip link set br0 up

7.7. VLAN ON BOND AND BRIDGE USING THE NETWORKMANAGER COMMAND LINE TOOL, NMCLI

To use VLANs over bonds and bridges, proceed as follows:

1. Add a bond device:

   ~$ nmcli connection add type bond con-name Bond0 ifname bond0
      bond.options "mode=active-backup,miimon=100" ipv4.method disabled
      ipv6.method ignore

   Note that in this case a bond connection serves only as a "lower interface" for VLAN, and does
   not get any IP address. Therefore, the ipv4.method disabled and ipv6.method ignore
   parameters have been added on the command line.

2. Add slaves to the bond device:

   ~$ nmcli connection add type ethernet con-name Slave1 ifname em1
      master bond0 slave-type bond
   ~$ nmcli connection add type ethernet con-name Slave2 ifname em2
      master bond0 slave-type bond

3. Add a bridge device:

   ~$ nmcli connection add type bridge con-name Bridge0 ifname br0 ip4
      192.0.2.1/24

4. Add a VLAN interface on top of bond, enslaved to the bridge device:

   ~$ nmcli connection add type vlan con-name Vlan2 ifname bond0.2 dev
      bond0 id 2 master br0 slave-type bridge

5. View the created connections:

   ~$ nmcli connection show
   NAME      UUID                                  TYPE
   DEVICE    
   Bond0     f05806fa-72c3-4803-8743-2377f0c10bed  bond
   bond0     
   Bridge0   22d3c0de-d79a-4779-80eb-10718c2bed61  bridge
             br0
   Slave1    e59e13cb-d749-4df2-ae6-de3bfaec698c  802-3-ethernet
             em1
7.8. CONFIGURING VLAN SWITCHPORT MODE

Red Hat Enterprise Linux machines are often used as routers and enable an advanced VLAN configuration on their network interfaces. You need to set switchport mode when the Ethernet interface is connected to a switch and there are VLANs running over the physical interface. A Red Hat Enterprise Linux server or workstation is usually connected to only one VLAN, which makes switchport mode access suitable, and the default setting.

In certain scenarios, multiple tagged VLANs use the same physical link, that is Ethernet between the switch and Red Hat Enterprise Linux machine, which requires switchport mode trunk to be configured on both ends.

For example, when a Red Hat Enterprise Linux machine is used as a router, the machine needs to forward tagged packets from the various VLANs behind the router to the switch over the same physical Ethernet, and maintain separation between those VLANs.

With the setup described, for example, in Section 7.3, “Configure 802.1Q VLAN Tagging Using the Command Line Tool, nmcli”, use the Cisco switchport mode trunk. If you only set an IP address on an interface, use Cisco switchport mode access.

7.9. ADDITIONAL RESOURCES

- **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.
- **nm-settings-ifcfg-rh(5) man page** – Description of ifcfg-rh settings in the /etc/sysconfig/network-scripts/ifcfg-* files.
CHAPTER 8. CONSISTENT NETWORK DEVICE NAMING

Red Hat Enterprise Linux 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 Red Hat Enterprise Linux, 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.

WARNING

Do not disable consistent network device naming because it allows the system using ethX style names, where X is a unique number corresponding to a specific interface and may have different names of network interfaces during the boot process. For more details, see Section 8.10, “Troubleshooting Network Device Naming”.

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 kernel 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:

- `o<index>`
  - on-board device index number

- `s<slot>[f<function>][d<dev_id>]`
  - hotplug slot index number. All multi-function PCI devices will carry the `[f<function>]` number in the device name, including the function 0 device.
MAC address

[\text{P<domain>}][\text{p<bus>}[s<slot>][f<function>][d<dev_id>]]

PCI geographical location. In PCI geographical location, the \([\text{P<domain>}]\) number is only mentioned if the value is not 0. For example:

\(\text{ID\_NET\_NAME\_PATH=P1enp5s0}\)

[\text{P<domain>}][\text{p<bus>}[s<slot>][f<function>][u<port>][..][c<config>][i<interface>]]

USB port number chain. 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. If there are multiple USB devices in the chain, the default values for USB configuration descriptors (c1) and USB interface descriptors (i0) are suppressed.

### 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:

\(\text{enccw0.0.1234}\)

CTC network devices of device type SLIP are named as follows:

\(\text{slccw0.0.1234}\)

Use the \text{znetconf -c} command or the \text{lscss -a} command to display available network devices and their bus-IDs.

Table 8.1. Device Name Types for Linux on System z

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enccwbus-\text{ID}</td>
<td>device type Ethernet</td>
</tr>
<tr>
<td>slccwbus-\text{ID}</td>
<td>CTC network devices of device type SLIP</td>
</tr>
</tbody>
</table>

### 8.5. NAMING SCHEME FOR VLAN INTERFACES

Traditionally, VLAN interface names in the format: \text{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 Red Hat Enterprise Linux 7, 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 through 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 8.2, “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 8.2. The biosdevname Naming Convention**

<table>
<thead>
<tr>
<th>Device</th>
<th>Old Name</th>
<th>New Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded network interface (LOM)</td>
<td><code>eth[0123...]</code></td>
<td><code>em[1234...]\textsuperscript{[a]}</code></td>
</tr>
<tr>
<td>PCI card network interface</td>
<td><code>eth[0123...]</code></td>
<td><code>p&lt;slot&gt;p&lt;ethernet port&gt;\textsuperscript{[b]}</code></td>
</tr>
<tr>
<td>Virtual function</td>
<td><code>eth[0123...]</code></td>
<td><code>p&lt;slot&gt;p&lt;ethernet port&gt;_&lt;virtual interface&gt;\textsuperscript{[c]}</code></td>
</tr>
</tbody>
</table>

\textsuperscript{[a]} New enumeration starts at 1.

\textsuperscript{[b]} For example: `p3p4`

\textsuperscript{[c]} 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:

```
~]# yum 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 through 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, is only recommended for special scenarios. See Chapter 8, Consistent Network Device Naming and Section 8.10, “Troubleshooting Network Device Naming” for more information.

To disable consistent network device naming, choose from one of the following:

- Disable the assignment of fixed names by "masking" udev's rule file for the default policy. This can be done by creating a symbolic link to /dev/null. As root, enter the following command:

```
~# 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 the new file above 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:

```
```

Edit the file in the /etc/udev/rules.d/ directory and change the lines as necessary.

- Open the /etc/default/grub/ file and find the GRUB_CMDLINE_LINUX variable.

**NOTE**

GRUB_CMDLINE_LINUX is a variable that includes entries which are added to the kernel command line. It might already contain additional configuration depending on your system settings.

Add both net.ifnames=0 and biosdevname=0 as kernel parameter values to the GRUB_CMDLINE_LINUX variable:

```
~# cat /etc/default/grub
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
```
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel_7/swap rd.luks.uuid=luks-cc387312-6da6-469a-8e49-b40cd58ad67a crashkernel=auto\n\nvconsole.keymap=us vconsole.font=latarcyrheb-sun16\nrd.lvm.lv=rhel_7/root rhgb quiet net.ifnames=0 biosdevname=0\nGRUB_DISABLE_RECOVERY="true"

Rebuild the /boot/grub2/grub.cfg/ file by running the grub2-mkconfig command:

~]$ grub2-mkconfig -o /boot/grub2/grub.cfg/

NOTE
For systems booted using UEFI:

~]$ grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg

View the current device name. For example, eno1:

~]$ nmcli connection show

<table>
<thead>
<tr>
<th>NAME</th>
<th>UUID</th>
<th>TYPE</th>
<th>DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wired</td>
<td>63cba8b2-60f7-4317-bc80-949e800a23cb</td>
<td>802-3-ethernet</td>
<td>eno1</td>
</tr>
</tbody>
</table>

Modify the device name to eth0, and reboot the system:

~]$ nmcli connection modify Wired connection.interface-name eth0

~]$ reboot

The grubby utility is used for updating and displaying information about the configuration files for the grub boot loader. See the grubby(8) man page for more details. For more information about working with GRUB 2, see the Red Hat Enterprise Linux System Administrator's Guide.

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,
From the above list of rule files it can be seen that if interface naming is done through 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 (through 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.

Example 8.3. Using Naming Scheme in ifcfg Files with ethX Names Does Not Work

Red Hat Enterprise Linux does not provide a way to consistently apply the ethX naming convention except under very specific circumstances.

The udev rules, which set an interface to a specific name, fail if the requested name is already in use by some other interface. This includes the functionality provided by the /usr/lib/udev/rules.d/60-net.rules file.

Kernel uses the ethX naming convention at boot time when it enumerates network devices. The ethx names are inconsistent across various reboots, and thus they are unpredictable. Consequently, attempting to use udev to rename an interface to an ethX name or to reorder the unpredictable ethX names given by the kernel fails.

Using the ethX names works correctly for the following scenarios:

- The system has only one network interface.
- When used for virtio NICs in Red Hat Enterprise Linux 7 virtual machine guests. See the KVM Paravirtualized (virtio) Drivers and Network Configuration chapters in the Virtualization Deployment and Administration Guide

Example 8.4. Setting net.ifnames=0 Results in Inconsistent ethX Names
If both `systemd` predictable interface naming (net.ifnames) and `biosdevname` naming schemes are disabled, network interfaces continue to use the unpredictable and potentially inconsistent ethX name originally given by the kernel.

Kernel always uses the ethX naming convention at boot when it enumerates network devices. Due to parallelization, the order of the kernel interface enumeration is expected to vary across reboots. Red Hat Enterprise Linux relies on either `systemd` predictable interface naming scheme or the `biosdevname` naming scheme to rename the kernel unpredictable ethX interfaces in a predictable way to a name which is always consistent across reboots.

For more information about network adapter naming conventions, see the Is it safe to set net.ifnames=0 in RHEL7? Knowledge Centered Support article on the Red Hat Customer Portal.

8.11. ADDITIONAL RESOURCES

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.

Online Documentation

- The IBM Knowledge Center Publication SC34-2710-00 Device Drivers, Features, and Commands on Red Hat Enterprise Linux 7 includes information on “Predictable network device names” for IBM System z devices and attachments.
PART II. INFINIBAND AND RDMA NETWORKING

This part discusses how to set up RDMA, InfiniBand, and IP over InfiniBand network connections.
CHAPTER 9. CONFIGURE INFINIBAND AND RDMA NETWORKS

9.1. UNDERSTANDING INFINIBAND AND RDMA TECHNOLOGIES

InfiniBand refers to two distinctly different things. The first is a physical link-layer protocol for InfiniBand networks. The second is a higher level programming API called the InfiniBand Verbs API. The InfiniBand Verbs API is an implementation of a remote direct memory access (RDMA) technology.

RDMA provides direct access from the memory of one computer to the memory of another without involving either computer’s operating system. This technology enables high-throughput, low-latency networking with low CPU utilization, which is especially useful in massively parallel computer clusters.

In a typical IP data transfer, application X on machine A sends some data to application Y on machine B. As part of the transfer, the kernel on machine B must first receive the data, decode the packet headers, determine that the data belongs to application Y, wake up application Y, wait for application Y to perform a read syscall into the kernel, then it must manually copy the data from the kernel’s own internal memory space into the buffer provided by application Y. This process means that most network traffic must be copied across the system’s main memory bus at least twice (once when the host adapter uses DMA to put the data into the kernel-provided memory buffer, and again when the kernel moves the data to the application’s memory buffer) and it also means the computer must execute a number of context switches to switch between kernel context and application Y context. Both of these things impose extremely high CPU loads on the system when network traffic is flowing at very high rates and can make other tasks slow down.

RDMA communications differ from normal IP communications because they bypass kernel intervention in the communication process, and in the process greatly reduce the CPU overhead normally needed to process network communications. The RDMA protocol allows the host adapter in the machine to know when a packet comes in from the network, which application should receive that packet, and where in the application’s memory space it should go. Instead of sending the packet to the kernel to be processed and then copied into the user application’s memory, it places the contents of the packet directly in the application’s buffer without any further intervention necessary. However, it cannot be accomplished using the standard Berkeley Sockets API that most IP networking applications are built upon, so it must provide its own API, the InfiniBand Verbs API, and applications must be ported to this API before they can use RDMA technology directly.

Red Hat Enterprise Linux 7 supports both the InfiniBand hardware and the InfiniBand Verbs API. In addition, there are two additional supported technologies that allow the InfiniBand Verbs API to be utilized on non-InfiniBand hardware:

- The Internet Wide Area RDMA Protocol (iWARP)
  
  iWARP is a computer networking protocol that implements remote direct memory access (RDMA) for efficient data transfer over Internet Protocol (IP) networks.

- The RDMA over Converged Ethernet (RoCE) protocol, which later renamed to InfiniBand over Ethernet (IBoE).
  
  RoCE is a network protocol that allows remote direct memory access (RDMA) over an Ethernet network.

Prerequisites
Both iWARP and RoCE technologies have a normal IP network link layer as their underlying technology, and so the majority of their configuration is actually covered in Chapter 2, Configure IP Networking. For the most part, once their IP networking features are properly configured, their RDMA features are all automatic and will show up as long as the proper drivers for the hardware are
installed. The kernel drivers are always included with each kernel Red Hat provides, however the user-space drivers must be installed manually if the InfiniBand package group was not selected at machine install time.

Since Red Hat Enterprise Linux 7.4, all RDMA user-space drivers are merged into the rdma-core package. To install all supported iWARP, RoCE or InfiniBand user-space drivers, enter as root:

```bash
~]# yum install libibverbs
```

If you are using Priority Flow Control (PFC) and mlx4-based cards, then edit `/etc/modprobe.d/mlx4.conf` to instruct the driver which packet priority is configured for the “no-drop” service on the Ethernet switches the cards are plugged into and rebuild the initramfs to include the modified file. Newer mlx5-based cards auto-negotiate PFC settings with the switch and do not need any module option to inform them of the “no-drop” priority or priorities.

To set the Mellanox cards to use one or both ports in Ethernet mode, see Section 9.5.4, “Configuring Mellanox cards for Ethernet operation”.

With these driver packages installed (in addition to the normal RDMA packages typically installed for any InfiniBand installation), a user should be able to utilize most of the normal RDMA applications to test and see RDMA protocol communication taking place on their adapters. However, not all of the programs included in Red Hat Enterprise Linux 7 properly support iWARP or RoCE/IBoE devices. This is because the connection establishment protocol on iWARP in particular is different than it is on real InfiniBand link-layer connections. If the program in question uses the librdmacm connection management library, it handles the differences between iWARP and InfiniBand silently and the program should work. If the application tries to do its own connection management, then it must specifically support iWARP or else it does not work.

### 9.2. TRANSFERRING DATA USING ROCE

RDMA over Converged Ethernet (RoCE) is a network protocol that enables remote direct memory access (RDMA) over an Ethernet network. There are two RoCE versions, RoCE v1 and RoCE v2, depending on the network adapter used.

**RoCE v1**

The RoCE v1 protocol is an Ethernet link layer protocol with ethertype 0x8915 that enables communication between any two hosts in the same Ethernet broadcast domain. RoCE v1 is the default version for RDMA Connection Manager (RDMA_CM) when using the ConnectX-3 network adapter.

**RoCE v2**

The RoCE v2 protocol exists on top of either the UDP over IPv4 or the UDP over IPv6 protocol. The UDP destination port number 4791 has been reserved for RoCE v2. Since Red Hat Enterprise Linux 7.5, RoCE v2 is the default version for RDMA_CM when using the ConnectX-3 Pro, ConnectX-4, ConnectX-4 Lx and ConnectX-5 network adapters. Hardware supports both RoCE v1 and RoCE v2.

RDMA Connection Manager (RDMA_CM) is used to set up a reliable connection between a client and a server for transferring data. RDMA_CM provides an RDMA transport-neutral interface for establishing connections. The communication is over a specific RDMA device, and data transfers are message-based.
Prerequisites
An RDMA_CM session requires one of the following:

- Both client and server support the same RoCE mode.
- A client supports RoCE v1 and a server RoCE v2.

Since a client determines the mode of the connection, the following cases are possible:

A successful connection:
If a client is in RoCE v1 or in RoCE v2 mode depending on the network card and the driver used, the corresponding server must have the same version to create a connection. Also, the connection is successful if a client is in RoCE v1 and a server in RoCE v2 mode.

A failed connection:
If a client is in RoCE v2 and the corresponding server is in RoCE v1, no connection can be established. In this case, update the driver or the network adapter of the corresponding server, see Section 9.2, “Transferring Data Using RoCE”

Table 9.1. RoCE Version Defaults Using RDMA_CM

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
<th>Default setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoCE v1</td>
<td>RoCE v1</td>
<td>Connection</td>
</tr>
<tr>
<td>RoCE v1</td>
<td>RoCE v2</td>
<td>Connection</td>
</tr>
<tr>
<td>RoCE v2</td>
<td>RoCE v2</td>
<td>Connection</td>
</tr>
<tr>
<td>RoCE v2</td>
<td>RoCE v1</td>
<td>No connection</td>
</tr>
</tbody>
</table>

That RoCE v2 on the client and RoCE v1 on the server are not compatible. To resolve this issue, force both the server and client-side environment to communicate over RoCE v1. This means to force hardware that supports RoCE v2 to use RoCE v1:

Procedure 9.1. Changing the Default RoCE Mode When the Hardware Is Already Running in RoCE v2

1. Change into the `/sys/kernel/config/rdma_cm` directory to set the RoCE mode:

   ```bash
   ~]$ cd /sys/kernel/config/rdma_cm
   ```

2. Enter the `ibstat` command with an Ethernet network device to display the status. For example, for `mlx5_0`:

   ```bash
   ~]$ ibstat mlx5_0
   CA 'mlx5_0'
   CA type: MT4115
   Number of ports: 1
   Firmware version: 12.17.1010
   Hardware version: 0
   Node GUID: 0x248a0703004bf0a4
   ```
3. Create a directory for the mlx5_0 device:

   ~
   mkdir mlx5_0

4. Display the RoCE mode in the default_roce_mode file in the tree format:

   ~
   cd mlx5_0

   ~
   tree
   ├── ports
   │    └── 1
   │           └── default_roce_mode
   │                   └── default_roce_tos

   ~
   cat /sys/kernel/config/rdma_cm/mlx5_0/ports/1/default_roce_mode
   RoCE v2

5. Change the default RoCE mode:

   ~
   echo "RoCE v1" >
   /sys/kernel/config/rdma_cm/mlx5_0/ports/1/default_roce_mode

6. View the changes:

   ~
   cat /sys/kernel/config/rdma_cm/mlx5_0/ports/1/default_roce_mode
   RoCE v1

9.3. CONFIGURING SOFT-ROCE

RoCE can be implemented both in the hardware and in the software. Soft-RoCE is the software implementation of the RDMA transport.

Prerequisites
Since Red Hat Enterprise Linux 7.4, the Soft-RoCE driver is already merged into the kernel. The userspace driver also is merged into the rdma-core package. Soft-RoCE is also known as RXE. To start, stop and configure RXE, use the rxe_cfg script. To view options for rxe_cfg, enter rxe_cfg help.
1. As the root user, enter the following command to display the current configuration status of RXE:

```
~# rxe_cfg
rdma_rxe module not loaded
Name   Link  Driver   Speed   NMTU  IPv4_addr  RDEV  RMTU
igb_1  yes   igb
mlx4_1  no    mlx4_en
mlx4_2  no    mlx4_en
```

2. To load the RXE kernel module and start RXE, enter as root:

```
~# rxe_cfg start
Name   Link  Driver   Speed   NMTU  IPv4_addr  RDEV  RMTU
igb_1  yes   igb
mlx4_1  no    mlx4_en
mlx4_2  no    mlx4_en
```

Optionally, to verify that the RXE kernel module is loaded, enter:

```
~# lsmod |grep rdma_rxe
rdma_rxe              111129  0
ip6_udp_tunnel         12755  1 rdma_rxe
udp_tunnel             14423  1 rdma_rxe
ib_core               236827  15 rdma_cm,ib_cm,iw_cm,rpcrdma,mlx4_ib,ib_srp,ib_ucm,ib_iser,ib_srpt,ib _umad,ib_uverbs,rdma_rxe,rdma_ucm,ib_ipoib,ib_isert
```

3. Before adding a new RXE device over an Ethernet interface, the corresponding interface should be opened and has a valid IP address assigned. To add a new RXE device, for example igb_1:

```
~# rxe_cfg add igb_1
```

```
~# rxe_cfg status
Name   Link  Driver   Speed   NMTU  IPv4_addr  RDEV  RMTU
igb_1  yes   igb
mlx4_1  no    mlx4_en
mlx4_2  no    mlx4_en
(3)
```

The rxe0 in the RDEV column indicates that rx e is enabled for the igb_1 device.

4. To verify the status of an RXE device, use the ibv_devices command:

```
~# ibv_devices
device                         node GUID
------                         ----------------
mlx4_0                         0002c90300b3cffe0
rxe0                            a2369fff018294
```

Alternatively, enter the ibstat for a detailed status:

```
~# ibstat rxe0
```
Removing an RXE device
If you want to remove an RXE device, enter:

```bash
~# rxe_cfg remove igb_1
```

Verifying Connectivity of an RXE device
The following examples show how to verify connectivity of an RXE device on the server and client side.

**Example 9.1. Verifying Connectivity of an RXE device on the Server Side**

```bash
~$ ibv_rc_pingpong -d rxe0 -g 0
local address: LID 0x0000, QPN 0x000012, PSN 0xe2965f, GID fe80::290:faff:fe29:486a
remote address: LID 0x0000, QPN 0x000011, PSN 0x4bf206, GID fe80::290:faff:fe29:470a
8192000 bytes in 0.05 seconds = 1244.06 Mbit/sec
1000 iters in 0.05 seconds = 52.68 usec/iter
```

**Example 9.2. Verifying Connectivity of an RXE device on the Client Side**

```bash
~$ ibv_rc_pingpong -d rxe0 -g 0 172.31.40.4
local address: LID 0x0000, QPN 0x000011, PSN 0x4bf206, GID fe80::290:faff:fe29:470a
remote address: LID 0x0000, QPN 0x000012, PSN 0xe2965f, GID fe80::290:faff:fe29:486a
8192000 bytes in 0.05 seconds = 1245.72 Mbit/sec
1000 iters in 0.05 seconds = 52.61 usec/iter
```

**9.4. INFINIBAND AND RDMA RELATED SOFTWARE PACKAGES**

Because RDMA applications are so different from Berkeley Sockets based applications, and from normal IP networking, most applications that are used on an IP network cannot be used directly on an RDMA network. Red Hat Enterprise Linux 7 comes with a number of different software packages for
RDMA network administration, testing and debugging, high level software development APIs, and performance analysis.

In order to utilize these networks, some or all of these packages need to be installed (this list is not exhaustive, but does cover the most important packages related to RDMA).

**Required packages:**

- **rdma** – responsible for kernel initialization of the RDMA stack.
- **libibverbs** – provides the InfiniBand Verbs API.
- **opensm** – subnet manager (only required on one machine, and only if there is no subnet manager active on the fabric).
- **user space driver for installed hardware** – one or more of: infinipath-psm, libcxgb3, libcxgb4, libehca, libipathverbs, libmthca, libmlx4, libmlx5, libnes, and libocrdma. Note that libehca is only available for IBM Power Systems servers.

**Recommended packages:**

- **librdmacm, librdmacm-utils, and ibacm** – Connection management library that is aware of the differences between InfiniBand, iWARP, and RoCE and is able to properly open connections across all of these hardware types, some simple test programs for verifying the operation of the network, and a caching daemon that integrates with the library to make remote host resolution faster in large clusters.
- **libibverbs-utils** – Simple Verbs based programs for querying the installed hardware and verifying communications over the fabric.
- **infiniband-diags and ibutils** – Provide a number of useful debugging tools for InfiniBand fabric management. These provide only very limited functionality on iWARP or RoCE as most of the tools work at the InfiniBand link layer, not the Verbs API layer.
- **perftest and qperf** – Performance testing applications for various types of RDMA communications.

**Optional packages:**

These packages are available in the Optional channel. Before installing packages from the Optional channel, see Scope of Coverage Details. Information on subscribing to the Optional channel can be found in the Red Hat Knowledgebase solution How to access Optional and Supplementary channels.

- **dapl, dapl-devel, and dapl-utils** – Provide a different API for RDMA than the Verbs API. There is both a runtime component and a development component to these packages.
- **openmpi, mvapich2, and mvapich2-psm** – MPI stacks that have the ability to use RDMA communications. User-space applications writing to these stacks are not necessarily aware that RDMA communications are taking place.

### 9.5. CONFIGURING THE BASE RDMA SUBSYSTEM

Startup of the rdma service is automatic. When RDMA capable hardware, whether InfiniBand or iWARP or RoCE/IBoE is detected, udev instructs systemd to start the rdma service.
Users need not enable the rdma service, but they can if they want to force it on all the time. To do that, enter the following command as root:

```
-]# systemctl enable rdma
```

### 9.5.1. Configuration of the rdma.conf file

The `rdma` service reads `/etc/rdma/rdma.conf` to find out which kernel-level and user-level RDMA protocols the administrator wants to be loaded by default. Users should edit this file to turn various drivers on or off.

The various drivers that can be enabled and disabled are:

- **IPoIB** – This is an IP network emulation layer that allows IP applications to run over InfiniBand networks.

- **SRP** – This is the SCSI Request Protocol. It allows a machine to mount a remote drive or drive array that is exported through the SRP protocol on the machine as though it were a local hard disk.

- **SRPT** – This is the target mode, or server mode, of the SRP protocol. This loads the kernel support necessary for exporting a drive or drive array for other machines to mount as though it were local on their machine. Further configuration of the target mode support is required before any devices will actually be exported. See the documentation in the `targetd` and `targetcli` packages for further information.

- **ISER** – This is a low-level driver for the general iSCSI layer of the Linux kernel that provides transport over InfiniBand networks for iSCSI devices.

- **RDS** – This is the Reliable Datagram Service in the Linux kernel. It is not enabled in Red Hat Enterprise Linux 7 kernels and so cannot be loaded.

### 9.5.2. Usage of 70-persistent-ipoib.rules

The `rdma` package provides the file `/etc/udev.d/rules.d/70-persistent-ipoib.rules`. This udev rules file is used to rename IPoIB devices from their default names (such as `ib0` and `ib1`) to more descriptive names. Users must edit this file to change how their devices are named. First, find out the GUID address for the device to be renamed:

```
-]$ ip link show ib0
8: ib0: >BROADCAST,MULTICAST,UP,LOWER_UP< mtu 65520 qdisc pfifo_fast state UP mode DEFAULT qlen 256
   link/infiniband
 80:00:02:00:fe:80:00:00:00:00:00:00:00:f4:52:14:03:00:7b:cb:a1 brd
00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff
```

Immediately after `link/infiniband` is the 20 byte hardware address for the IPoIB interface. The
final 8 bytes of the address, marked in bold above, is all that is required to make a new name. Users can make up whatever naming scheme suits them. For example, use a `device_fabric` naming convention such as `mlx4_ib0` if a mlx4 device is connected to an `ib0` subnet fabric. The only thing that is not recommended is to use the standard names, like `ib0` or `ib1`, as these can conflict with the kernel assigned automatic names. Next, add an entry in the rules file. Copy the existing example in the rules file, replace the 8 bytes in the `ATTR{address}` entry with the highlighted 8 bytes from the device to be renamed, and enter the new name to be used in the `NAME` field.

9.5.3. Relaxing memlock restrictions for users

RDMA communications require that physical memory in the computer be pinned (meaning that the kernel is not allowed to swap that memory out to a paging file in the event that the overall computer starts running short on available memory). Pinning memory is normally a very privileged operation. In order to allow users other than `root` to run large RDMA applications, it will likely be necessary to increase the amount of memory that non-`root` users are allowed to pin in the system. This is done by adding a file in the `/etc/security/limits.d/` directory with contents such as the following:

```bash
-]$ more /etc/security/limits.d/rdma.conf
# configuration for rdma tuning
* soft memlock unlimited
* hard memlock unlimited
# rdma tuning end
```

9.5.4. Configuring Mellanox cards for Ethernet operation

Certain hardware from Mellanox is capable of running in either InfiniBand or Ethernet mode. These cards generally default to InfiniBand. Users can set the cards to Ethernet mode. There is currently support for setting the mode only on ConnectX family hardware (which uses either the mlx5 or mlx4 driver).

To configure Mellanox mlx5 cards, use the `mstconfig` program from the `mstflint` package. For more details, see the Configuring Mellanox mlx5 cards in Red Hat Enterprise Linux 7 Knowledge Base article on the Red Hat Customer Portal.

To configure Mellanox mlx4 cards, use `mstconfig` to set the port types on the card as described in the Knowledge Base article. If `mstconfig` does not support your card, edit the `/etc/rdma/mlx4.conf` file and follow the instructions in that file to set the port types properly for RoCE/IBoE usage. In this case is also necessary to rebuild the `initramfs` to make sure the updated port settings are copied into the `initramfs`.

Once the port type has been set, if one or both ports are set to Ethernet and `mstconfig` was not used to set the port types, then users might see this message in their logs:

```bash
mlx4_core 0000:05:00.0: Requested port type for port 1 is not supported on this HCA
```

This is normal and does not affect operation. The script responsible for setting the port type has no way of knowing when the driver has finished switching port 2 to the requested type internally, and from the time that the script issues a request for port 2 to switch until that switch is complete, the attempts to set port 1 to a different type get rejected. The script retries until the command succeeds or until a timeout has passed indicating that the port switch never completed.

9.5.5. Connecting to a Remote Linux SRP Target
The SCSI RDMA Protocol (SRP) is a network protocol that enables a system to use RDMA to access
SCSI devices that are attached to another system. To allow an SRP initiator to connect an SRP target
on the SRP target side, you must add an access control list (ACL) entry for the host channel adapter
(HCA) port used in the initiator.

ACL IDs for HCA ports are not unique. The ACL IDs depend on the GID format of the HCAs. HCAs that
use the same driver, for example `ib_qib`, can have different format of GIDs. The ACL ID also depends
on how you initiate the connection request.

Connecting to a Remote Linux SRP Target: High-Level Overview

1. Prepare the target side:
   1. Create storage back end. For example get the `/dev/sdc1` partition:
      ```
      /backstores/block create vol1 /dev/sdc1
      ```
   2. Create an SRP target:
      ```
      /srpt create 0xfe80000000000000001175000077dd7e
      ```
   3. Create a LUN based on the back end created in step a:
      ```
      /srpt/ib.fe80000000000000001175000077dd7e/luns create
      /backstores/block/vol1
      ```
   4. Create a Node ACL for the remote SRP client:
      ```
      /srpt/ib.fe80000000000000001175000077dd7e/acls create
      0x7edd770000751100001175000077d708
      ```
      Note that the Node ACL is different for `srp_daemon` and `ibsrpdm`.

2. Initiate an SRP connection with `srp_daemon` or `ibsrpdm` for the client side:
   ```
   [root@initiator]# srp_daemon -e -n -i qib0 -p 1 -R 60 &
   [root@initiator]# ibsrpdm -c -d /dev/infiniband/umad0 >
   /sys/class/infiniband_srp/srp-qib0-1/add_target
   ```
   Optional. It is recommended to verify the SRP connection with different tools, such as `lsscsi`
or `dmesg`.

Procedure 9.3. Connecting to a Remote Linux SRP Target with `srp_daemon` or `ibsrpdm`

1. Use the `ibstat` command on the target to determine the `State` and `Port GUID` values. The
   HCA must be in `Active` state. The ACL ID is based on the `Port GUID`:
   ```
   [root@target]# ibstat
   CA 'qib0'
   CA type: InfiniPath_QLE7342
   Number of ports: 1
   Firmware version:
   ```
 Hardware version: 2
Node GUID: 0x001175000077dd7e
System image GUID: 0x001175000077dd7e
Port 1:
 State: Active
 Physical state: LinkUp
 Rate: 40
 Base lid: 1
 LMC: 0
 SM lid: 1
 Capability mask: 0x0769086a
 Port GUID: 0x001175000077dd7e
 Link layer: InfiniBand

2. Get the SRP target ID, which is based on the GUID of the HCA port. Note that you need a dedicated disk partition as a back end for a SRP target, for example /dev/sdc1. The following command replaces the default prefix of fe80, removes the colon, and adds the new prefix to the remainder of the string:

```
[root@target]# ibstatus | grep '<default-gid>' | sed -e 's/<default-gid>://g' -e 's/://g' | grep 001175000077dd7e
```

3. Use the `targetcli` tool to create the LUN vol1 on the block device, create an SRP target, and export the LUN:

```
[root@target]# targetcli
```

```
/> /backstores/block create vol1 /dev/sdc1
Created block storage object vol1 using /dev/sdc1.
/> /srpt create 0xfe80000000000000001175000077dd7e
Created target ib.fe80000000000000001175000077dd7e.
/> /srpt/ib.fe80000000000000001175000077dd7e/luns create
/backstores/block/vol1
Created LUN 0.
/> ls /
 o- /
 ... .........................................................
 o- backstores
 ... .........................................................
 [....]  o- block ...........................................
 [Storage Objects: 1]
 | o- vol1 ........................................... [/dev/sdc1 (77.8GiB)
 write-thru activated]
 | o- fileio ...........................................
 [Storage Objects: 0]
 | o- pscsi ...........................................
 [Storage Objects: 0]
 | o- ramdisk .......................................
 [Storage Objects: 0]
 | o- iscsi
 ..........................................................
```

Networking Guide

186
4. Use the `ibstat` command on the initiator to check if the state is **Active** and determine the **Port GUID**:

```
[root@initiator]# ibstat
CA 'qib0'
   CA type: InfiniPath_QLE7342
   Number of ports: 1
   Firmware version: 
   Hardware version: 2
   Node GUID: 0x001175000077d708
   System image GUID: 0x001175000077d708
   Port 1:
      State: Active
      Physical state: LinkUp
      Rate: 40
      Base lid: 2
      LMC: 0
      SM lid: 1
      Capability mask: 0x07690868
      Port GUID: 0x001175000077d708
      Link layer: InfiniBand
```

5. Use the following command to scan without connecting to a remote SRP target. The target GUID shows that the initiator had found remote target. The ID string shows that the remote target is a Linux software target (`ib_srpt.ko`).

```
[root@initiator]# srp_daemon -a -o
IO Unit Info:
   port LID:     0001
   port GID:      fe800000000000001175000077dd7e
   change ID:     0001
   max controllers: 0x10

   controller[ 1]
      GUID: 001175000077dd7e
      vendor ID: 000011
```
device ID: 007322
IO class: 0100
ID: Linux SRP target
service entries: 1
  service[ 0]: 001175000077dd7e / SRP.T10:001175000077dd7e

6. To verify the SRP connection, use the lsscsi command to list SCSI devices and compare the lsscsi output before and after the initiator connects to target.

[root@initiator]# lsscsi
[0:0:10:0] disk IBM-ESXS ST9146803SS B53C /dev/sda

7. To connect to a remote target without configuring a valid ACL for the initiator port, which is expected to fail, use the following commands for srp_daemon or ibsrpdm:

[root@initiator]# srp_daemon -e -n -i qib0 -p 1 -R 60 &
[1] 4184
[root@initiator]# ibsrpdm -c -d /dev/infiniband/umad0 > /sys/class/infiniband_srp/srp-qib0-1/add_target

8. The output of the dmesg shows why the SRP connection operation failed. In a later step, the dmesg command on the target side is used to make the situation clear.

[root@initiator]# dmesg -c
[ 1230.059652] scsi host5: ib_srp: REJ received
[ 1230.073792] scsi host5: ib_srp: Connection 0/2 failed
[ 1230.078848] scsi host5: ib_srp: Sending CM DREQ failed

9. Because of failed LOGIN, the output of the lsscsi command is the same as in the earlier step.

[root@initiator]# lsscsi
[0:0:10:0] disk IBM-ESXS ST9146803SS B53C /dev/sda

10. Using the dmesg on the target side (ib_srpt.ko) provides an explanation of why LOGIN failed. Also, the output contains the valid ACL ID provided by srp_daemon:
0x7edd770000751100001175000077d708.

[root@target]# dmesg
[ 1200.303001] ib_srpt Received SRP_LOGIN_REQ with i_port_id 0x7edd770000751100:0x1175000077d708, t_port_id 0x1175000077dd7e:0x1175000077dd7e and it_iu_len 260 on port 1 (guid=0xfe80000000000000:0x1175000077dd7e)
[ 1200.322207] ib_srpt Rejected login because no ACL has been configured yet for initiator 0x7edd770000751100001175000077d708.

11. Use the targetcli tool to add a valid ACL:
12. Verify the SRP LOGIN operation:

a. Wait for 60 seconds to allow srp_daemon to re-try logging in:

```
[root@initiator]# sleep 60
```

b. Verify the SRP LOGIN operation:

```
[root@initiator]# lsscsi
[0:0:0:0]   disk    IBM-ESXS ST9146803SS      B53C  /dev/sda
[7:0:0:0] disk LIO-ORG vol1 4.0 /dev/sdb
```

c. For a kernel log of SRP target discovery, use:

```
[root@initiator]# dmesg -c
[ 1354.182072] scsi host7: SRP.T10:001175000077DD7E
[ 1354.187258] scsi 7:0:0:0: Direct-Access     LIO-ORG vol1 4.0  PQ: 0 ANSI: 5
[ 1354.208688] scsi 7:0:0:0: alua: supports implicit and explicit TPGS
[ 1354.215698] scsi 7:0:0:0: alua: port group 00 rel port 01
[ 1354.221409] scsi 7:0:0:0: alua: port group 00 state A non-preferred supports TOlUSNA
[ 1354.229147] scsi 7:0:0:0: alua: Attached
[ 1354.233402] sd 7:0:0:0: Attached scsi generic sg1 type 0
[ 1354.235127] sd 7:0:0:0: Write Protect is off
[ 1354.235550] sd 7:0:0:0: [sdb] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[ 1354.255491] sd 7:0:0:0: [sdb] Attached SCSI disk
```

9.6. CONFIGURING THE SUBNET MANAGER

9.6.1. Determining Necessity

Most InfiniBand switches come with an embedded subnet manager. However, if a more up to date
subnet manager is required than the one in the switch firmware, or if more complete control than the
switch manager allows is required, Red Hat Enterprise Linux 7 includes the opensm subnet
manager. All InfiniBand networks must have a subnet manager running for the network to
function. This is true even when doing a simple network of two machines with no switch and the cards
are plugged in back to back, a subnet manager is required for the link on the cards to come up. It is
possible to have more than one, in which case one will act as master, and any other subnet managers
will act as slaves that will take over should the master subnet manager fail.

9.6.2. Configuring the opensm master configuration file

The opensm program keeps its master configuration file in /etc/rdma/opensm.conf. Users may
edit this file at any time and edits will be kept on upgrade. There is extensive documentation of the
options in the file itself. However, for the two most common edits needed, setting the GUID to bind to
and the PRIORITY to run with, it is highly recommended that the opensm.conf file is not edited but
instead edit /etc/sysconfig/opensm. If there are no edits to the base /etc/rdma/opensm.conf
file, it will get upgraded whenever the opensm package is upgraded. As new options are added to this
file regularly, this makes it easier to keep the current configuration up to date. If the opensm.conf file
has been changed, then on upgrade, it might be necessary to merge new options into the edited file.

9.6.3. Configuring the opensm startup options

The options in the /etc/sysconfig/opensm file control how the subnet manager is actually started,
as well as how many copies of the subnet manager are started. For example, a dual port InfiniBand
card, with each port plugged into physically separate networks, will need a copy of the subnet manager
running on each port. The opensm subnet manager will only manage one subnet per instance of the
application and must be started once for each subnet that needs to be managed. In addition, if there is
more than one opensm server, then set the priorities on each server to control which are to be slaves
and which are to be master.

The file /etc/sysconfig/opensm is used to provide a simple means to set the priority of the subnet
manager and to control which GUID the subnet manager binds to. There is an extensive explanation of
the options in the /etc/sysconfig/opensm file itself. Users need only read and follow the directions
in the file itself to enable failover and multifabric operation of opensm.

9.6.4. Creating a P_Key definition

By default, opensm.conf looks for the file /etc/rdma/partitions.conf to get a list of partitions
to create on the fabric. All fabrics must contain the 0x7fff subnet, and all switches and all hosts must
belong to that fabric. Any other partition can be created in addition to that, and all hosts and all
switches do not have to be members of these additional partitions. This allows an administrator to
create subnets akin to Ethernet’s VLANs on InfiniBand fabrics. If a partition is defined with a given
speed, such as 40 Gbps, and there is a host on the network unable to do 40 Gbps, then that host will be
unable to join the partition even if it has permission to do so as it will be unable to match the speed
requirements, therefore it is recommended that the speed of a partition be set to the slowest speed of
any host with permission to join the partition. If a faster partition for some subset of hosts is required,
create a different partition with the higher speed.

The following partition file would result in a default 0x7fff partition at a reduced speed of 10 Gbps,
and a partition of 0x0002 with a speed of 40 Gbps:

```
-]$ more /etc/rdma/partitions.conf
# For reference:
# IPv4 IANA reserved multicast addresses:
# http://www.iana.org/assignments/multicast-addresses/multicast-
addresses.txt

# IPv6 IANA reserved multicast addresses:
# http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xml
#
# mtu =
# 1 = 256
# 2 = 512
# 3 = 1024
# 4 = 2048
# 5 = 4096
#
# rate =
# 2 = 2.5 GBit/s
# 3 = 10 GBit/s
# 4 = 30 GBit/s
# 5 = 5 GBit/s
# 6 = 20 GBit/s
# 7 = 40 GBit/s
# 8 = 60 GBit/s
# 9 = 80 GBit/s
# 10 = 120 GBit/s

Default=0x7fff, rate=3, mtu=4, scope=2, defmember=full:
   ALL, ALL_SWITCHES=full;
Default=0x7fff, ipoib, rate=3, mtu=4, scope=2:
   mgid=ff12:401b::ffff:ffff       # IPv4 Broadcast address
   mgid=ff12:401b::1               # IPv4 All Hosts group
   mgid=ff12:401b::2               # IPv4 All Routers group
   mgid=ff12:401b::16              # IPv4 IGMP group
   mgid=ff12:401b::fb              # IPv4 mDNS group
   mgid=ff12:401b::fc              # IPv4 Multicast Link Local Name
Resolution group
   mgid=ff12:401b::101             # IPv4 NTP group
   mgid=ff12:401b::202             # IPv4 Sun RPC
   mgid=ff12:601b::1               # IPv6 All Hosts group
   mgid=ff12:601b::2               # IPv6 All Routers group
   mgid=ff12:601b::16              # IPv6 MLDv2-capable Routers
           group
   mgid=ff12:601b::fb              # IPv6 mDNS group
   mgid=ff12:601b::101             # IPv6 NTP group
   mgid=ff12:601b::202             # IPv6 Sun RPC group
   mgid=ff12:601b::1:3             # IPv6 Multicast Link Local Name
Resolution group
   ALL=full, ALL_SWITCHES=full;

ib0_2=0x0002, rate=7, mtu=4, scope=2, defmember=full:
   ALL, ALL_SWITCHES=full;
ib0_2=0x0002, ipoib, rate=7, mtu=4, scope=2:
   mgid=ff12:401b::ffff:ffff       # IPv4 Broadcast address
   mgid=ff12:401b::1               # IPv4 All Hosts group
   mgid=ff12:401b::2               # IPv4 All Routers group
   mgid=ff12:401b::16              # IPv4 IGMP group
   mgid=ff12:401b::fb              # IPv4 mDNS group
   mgid=ff12:401b::fc              # IPv4 Multicast Link Local Name
Resolution group
9.6.5. Enabling opensm

Users need to enable the opensm service as it is not enabled by default when installed. Issue the following command as root:

```
[~]# systemctl enable opensm
```

9.7. TESTING EARLY INFINIBAND RDMA OPERATION

**NOTE**

This section applies only to InfiniBand devices. Since iWARP and RoCE/IBoE devices are IP based devices, users should proceed to the section on testing RDMA operations once IPoIB has been configured and the devices have IP addresses.

Once the rdma service is enabled, and the opensm service (if needed) is enabled, and the proper user-space library for the specific hardware has been installed, user space rdma operation should be possible. Simple test programs from the libibverbs-utils package are helpful in determining that RDMA operations are working properly. The ibv_devices program will show which devices are present in the system and the ibv_devinfo command will give detailed information about each device. For example:

```
[~]# ibv_devices
device node GUID
------ ----------------
mlx4_0 0002c903003178f0
mlx4_1 f4521403007bcba0
```

```
[~]# ibv_devinfo -d mlx4_1
hca_id: mlx4_1
  transport: InfiniBand (0)
  fw_ver: 2.30.8000
  node_guid: f452:1403:007b:cb0
  sys_image_guid: f452:1403:007b:cb3
  vendor_id: 0x02c9
  vendor_part_id: 4099
  hw_ver: 0x0
  board_id: MT_1090120019
  phys_port_cnt: 2
  port: 1
    state: PORT_ACTIVE (4)
    max_mtu: 4096 (5)
```
The `ibv_devinfo` and `ibstat` commands output slightly different information (such as port MTU exists in `ibv_devinfo` but not in `ibstat` output, and the Port GUID exists in `ibstat` output but not in `ibv_devinfo` output), and a few things are named differently (for example, the Base local identifier (LID) in `ibstat` output is the same as the `port_lid` output of `ibv_devinfo`).

Simple ping programs, such as `ibping` from the `infiniband-diags` package, can be used to test RDMA connectivity. The `ibping` program uses a client-server model. You must first start an `ibping` server on one machine, then run `ibping` as a client on another machine and tell it to connect to the `ibping` server. Since we are wanting to test the base RDMA capability, we need to use an RDMA specific address resolution method instead of IP addresses for specifying the server.

On the server machine, the user can use the `ibv_devinfo` and `ibstat` commands to print out the `port_lid` (or Base lid) and the Port GUID of the port they want to test (assuming port 1 of the above
interface, the `port_lid/Base LID` is 2 and Port GUID is `0xf4521403007bcba1`). Then start `ibping` with the necessary options to bind specifically to the card and port to be tested, and also specifying `ibping` should run in server mode. You can see the available options to `ibping` by passing `-?` or `--help`, but in this instance we will need either the `-S` or `--Server` option and for binding to the specific card and port we will need either `-C` or `--Ca` and `-P` or `--Port`. Note: port in this instance does not denote a network port number, but denotes the physical port number on the card when using a multi-port card. To test connectivity to the RDMA fabric using, for example, the second port of a multi-port card, requires telling `ibping` to bind to port 2 on the card. When using a single port card, or testing the first port on a card, this option is not needed. For example:

```
~$ ibping -S -C mlx4_1 -P 1
```

Then change to the client machine and run `ibping`. Make note of either the port GUID of the port the server `ibping` program is bound to, or the `local identifier` (LID) of the port the server `ibping` program is bound to. Also, take note which card and port in the client machine is physically connected to the same network as the card and port that was bound to on the server. For example, if the second port of the first card on the server was bound to, and that port is connected to a secondary RDMA fabric, then on the client specify whichever card and port are necessary to also be connected to that secondary fabric. Once these things are known, run the `ibping` program as a client and connect to the server using either the port LID or GUID that was collected on the server as the address to connect to. For example:

```
~$ ibping -c 10000 -f -C mlx4_0 -P 1 -L 2
--- rdma-host.example.com.(none) (Lid 2) ibping statistics ---
10000 packets transmitted, 10000 received, 0% packet loss, time 816 ms
rtt min/avg/max = 0.032/0.081/0.446 ms
```

or

```
~$ ibping -c 10000 -f -C mlx4_0 -P 1 -G 0xf4521403007bcba1 \
--- rdma-host.example.com.(none) (Lid 2) ibping statistics ---
10000 packets transmitted, 10000 received, 0% packet loss, time 769 ms
rtt min/avg/max = 0.027/0.076/0.278 ms
```

This outcome verifies that end to end RDMA communications are working for user space applications.

The following error may be encountered:

```
~$ ibv_devinfo
libibverbs: Warning: no userspace device-specific driver found for
/sys/class/infiniband_verbs/uverbs0
No IB devices found
```

This error indicates that the necessary user-space library is not installed. The administrator will need to install one of the user-space libraries (as appropriate for their hardware) listed in section Section 9.4, “InfiniBand and RDMA related software packages”. On rare occasions, this can happen if a user installs the wrong arch type for the driver or for `libibverbs`. For example, if `libibverbs` is of arch `x86_64`, and `libmlx4` is installed but is of type `i686`, then this error can result.
NOTE

Many sample applications prefer to use host names or addresses instead of LIDs to open communication between the server and client. For those applications, it is necessary to set up IPoIB before attempting to test end-to-end RDMA communications. The `ibping` application is unusual in that it will accept simple LIDs as a form of addressing, and this allows it to be a simple test that eliminates possible problems with IPoIB addressing from the test scenario and therefore gives us a more isolated view of whether or not simple RDMA communications are working.

9.8. CONFIGURING IPOIB

9.8.1. Understanding the role of IPoIB

As mentioned in Section 1.2, “IP Networks versus non-IP Networks”, most networks are IP networks. InfiniBand is not. The role of IPoIB is to provide an IP network emulation layer on top of InfiniBand RDMA networks. This allows existing applications to run over InfiniBand networks unmodified. However, the performance of those applications is considerably lower than if the application were written to use RDMA communication natively. Since most InfiniBand networks have some set of applications that really must get all of the performance they can out of the network, and then some other applications for which a degraded rate of performance is acceptable if it means that the application does not need to be modified to use RDMA communications, IPoIB is there to allow those less critical applications to run on the network as they are.

Because both iWARP and RoCE/IBoE networks are actually IP networks with RDMA layered on top of their IP link layer, they have no need of IPoIB. As a result, the kernel will refuse to create any IPoIB devices on top of iWARP or RoCE/IBoE RDMA devices.

9.8.2. Understanding IPoIB communication modes

IPoIB devices can be configured to run in either datagram or connected mode. The difference is in what type of queue pair the IPoIB layer attempts to open with the machine at the other end of the communication. For datagram mode, an unreliable, disconnected queue pair is opened. For connected mode, a reliable, connected queue pair is opened.

When using datagram mode, the unreliable, disconnected queue pair type does not allow any packets larger than the InfiniBand link-layer’s MTU. The IPoIB layer adds a 4 byte IPoIB header on top of the IP packet being transmitted. As a result, the IPoIB MTU must be 4 bytes less than the InfiniBand link-layer MTU. As 2048 is a common InfiniBand link-layer MTU, the common IPoIB device MTU in datagram mode is 2044.

When using connected mode, the reliable, connected queue pair type allows messages that are larger than the InfiniBand link-layer MTU and the host adapter handles packet segmentation and reassembly at each end. As a result, there is no size limit imposed on the size of IPoIB messages that can be sent by the InfiniBand adapters in connected mode. However, there is still the limitation that an IP packet only has a 16 bit size field, and is therefore limited to 65535 as the maximum byte count. The maximum allowed MTU is actually smaller than that because we have to account for various TCP/IP headers that must also fit in that size. As a result, the IPoIB MTU in connected mode is capped at 65520 in order to make sure there is sufficient room for all needed TCP headers.

The connected mode option generally has higher performance, but it also consumes more kernel memory. Because most systems care more about performance than memory consumption, connected mode is the most commonly used mode.
However, if a system is configured for connected mode, it must still send multicast traffic in datagram mode (the InfiniBand switches and fabric cannot pass multicast traffic in connected mode) and it will also fall back to datagram mode when communicating with any hosts not configured for connected mode. Administrators should be aware that if they intend to run programs that send multicast data, and those programs try to send multicast data up to the maximum MTU on the interface, then it is necessary to configure the interface for datagram operation or find some way to configure the multicast application to cap their packet send size at a size that will fit in datagram sized packets.

9.8.3. Understanding IPoIB hardware addresses

IPoIB devices have a 20 byte hardware addresses. The deprecated utility ifconfig is unable to read all 20 bytes and should never be used to try and find the correct hardware address for an IPoIB device. The ip utilities from the iproute package work properly.

The first 4 bytes of the IPoIB hardware address are flags and the queue pair number. The next 8 bytes are the subnet prefix. When the IPoIB device is first created, it will have the default subnet prefix of 0xfe : 80 : 00 : 00 : 00 : 00 : 00 : 00. The device will use the default subnet prefix (0xfe00000000000000) until it makes contact with the subnet manager, at which point it will reset the subnet prefix to match what the subnet manager has configured it to be. The final 8 bytes are the GUID address of the InfiniBand port that the IPoIB device is attached to. Because both the first 4 bytes and the next 8 bytes can change from time to time, they are not used or matched against when specifying the hardware address for an IPoIB interface. Section Section 9.5.2, “Usage of 70-persistent-ipoib.rules” explains how to derive the address by leaving the first 12 bytes out of the ATTR{address} field in the udev rules file so that device matching will happen reliably. When configuring IPoIB interfaces, the HWADDR field of the configuration file can contain all 20 bytes, but only the last 8 bytes are actually used to match against and find the hardware specified by a configuration file. However, if the TYPE=InfiniBand entry is not spelled correctly in the device configuration file, and ifup-ib is not the actual script used to open the IPoIB interface, then an error about the system being unable to find the hardware specified by the configuration will be issued. For IPoIB interfaces, the TYPE= field of the configuration file must be either InfiniBand or infiniband (the entry is case sensitive, but the scripts will accept these two specific spellings).

9.8.4. Understanding InfiniBand P_Key subnets

An InfiniBand fabric can be logically segmented into virtual subnets by the use of different P_Key subnets. This is highly analogous to using VLANs on Ethernet interfaces. All switches and hosts must be a member of the default P_Key subnet, but administrators can create additional subnets and limit members of those subnets to subsets of the hosts or switches in the fabric. A P_Key subnet must be defined by the subnet manager before a host can use it. See section Section 9.6.4, “Creating a P_Key definition” for information on how to define a P_Key subnet using the opensm subnet manager. For IPoIB interfaces, once a P_Key subnet has been created, we can create additional IPoIB configuration files specifically for those P_Key subnets. Just like VLAN interfaces on Ethernet devices, each IPoIB interface will behave as though it were on a completely different fabric from other IPoIB interfaces that share the same link but have different P_Key values.

There are special requirements for the names of IPoIB P_Key interfaces. All IPoIB P_Keys range from 0x0000 to 0x7fff, and the high bit, 0x8000, denotes that membership in a P_Key is full membership instead of partial membership. The Linux kernel’s IPoIB driver only supports full membership in P_Key subnets, so for any subnet that Linux can connect to, the high bit of the P_Key number will always be set. That means that if a Linux computer joins P_Key 0x0002, its actual P_Key number once joined will be 0x8002, denoting that we are full members of P_Key 0x0002. For this reason, when creating a P_Key definition in an opensm partitions.conf file as depicted in section Section 9.6.4, “Creating a P_Key definition”, it is required to specify a P_Key value without 0x8000, but when defining the P_Key IPoIB interfaces on the Linux clients, add the 0x8000 value to the base P_Key value.
9.9. CONFIGURE INFINIBAND USING THE TEXT USER INTERFACE, NMTUI

The text user interface tool nmtui can be used to configure InfiniBand in a terminal window. Issue the following command to start the tool:

```
~]$ nmtui
```

The text user interface appears. Any invalid command prints a usage message.

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

From the starting menu, select **Edit a connection**. Select **Add**, the **New Connection** screen opens.

![Figure 9.1. The NetworkManager Text User Interface Add an InfiniBand Connection menu](image)

Select **InfiniBand**, the **Edit connection** screen opens. Follow the on-screen prompts to complete the configuration.
9.10. CONFIGURE IPOIB USING THE COMMAND-LINE TOOL, NMCLI

First determine if renaming the default IPoIB device(s) is required, and if so, follow the instructions in section Section 9.5.2, “Usage of 70-persistent-ipoib.rules” to rename the devices using udev renaming rules. Users can force the IPoIB interfaces to be renamed without performing a reboot by removing the ib_ipoib kernel module and then reloading it as follows:

```bash
$ rmmod ib_ipoib
$ modprobe ib_ipoib
```

Once the devices have the name required, use the nmcli tool to create the IPoIB interface(s). The following examples display two ways:

**Example 9.3. Creating and modifying IPoIB in two separate commands.**

```bash
$ nmcli con add type infiniband con-name mlx4_ib0 ifname mlx4_ib0 transport-mode connected mtu 65520
Connection 'mlx4_ib0' (8029a0d7-8b05-49ff-a826-2a6d722025cc) successfully added.
$ nmcli con edit mlx4_ib0
===| nmcli interactive connection editor |===
Editing existing 'infiniband' connection: 'mlx4_ib0'
Type 'help' or '?' for available commands.
```
Type 'describe [>setting<.>prop<]' for detailed property description.

You may edit the following settings: connection, infiniband, ipv4, ipv6

```
nmcli> set infiniband.mac-address 80:00:02:00:fe:80:00:00:00:00:00:00:f4:52:14:03:00:7b:cb:a3
nmcli> save
Connection 'mlx4_ib3' (8029a0d7-8b05-49ff-a826-2a6d722025cc) successfully updated.
nmcli> quit
```

or you can run `nmcli c add` and `nmcli c modify` in one command, as follows:

```
Example 9.4. Creating and modifying IPoIB in one command.

```

```
nmcli con add type infiniband con-name mlx4_ib0 ifname mlx4_ib0 transport-mode connected mtu 65520 infiniband.mac-address 80:00:02:00:fe:80:00:00:00:00:00:00:f4:52:14:03:00:7b:cb:a3
```

At these points, an IPoIB interface named `mlx4_ib0` has been created and set to use connected mode, with the maximum connected mode MTU, DHCP for IPv4 and IPv6. If using IPoIB interfaces for cluster traffic and an Ethernet interface for out-of-cluster communications, it is likely that disabling default routes and any default name server on the IPoIB interfaces will be required. This can be done as follows:

```
~]$ nmcli con edit mlx4_ib0

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

Editing existing 'infiniband' connection: 'mlx4_ib0'

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

You may edit the following settings: connection, infiniband, ipv4, ipv6

```
nmcli> set ipv4.ignore-auto-dns yes
nmcli> set ipv4.ignore-auto-routes yes
nmcli> set ipv4.never-default true
nmcli> set ipv6.ignore-auto-dns yes
nmcli> set ipv6.ignore-auto-routes yes
nmcli> set ipv6.never-default true
nmcli> save
Connection 'mlx4_ib0' (8029a0d7-8b05-49ff-a826-2a6d722025cc) successfully updated.
nmcli> quit
```

If a P-Key interface is required, create one using `nmcli` as follows:

```
~]$ nmcli con add type infiniband con-name mlx4_ib0.8002 ifname mlx4_ib0.8002 parent mlx4_ib0 p-key 0x8002
Connection 'mlx4_ib0.8002' (4a9f5509-7bd9-4e89-87e9-77751a1c54b4) successfully added.
~]$ nmcli con modify mlx4_ib0.8002 infiniband.mtu 65520
```

9.11. CONFIGURE IPOIB USING THE COMMAND LINE

First determine if renaming the default IPoIB device(s) is required, and if so, follow the instructions in section Section 9.5.2, “Usage of 70-persistent-ipoib.rules” to rename the devices using udev renaming rules. Users can force the IPoIB interfaces to be renamed without performing a reboot by removing the ib_ipoib kernel module and then reloading it as follows:

```bash
~]$ rmmod ib_ipoib
~]$ modprobe ib_ipoib
```

Once the devices have the name required, administrators can create ifcfg files with their preferred editor to control the devices. A typical IPoIB configuration file with static IPv4 addressing looks as follows:

```bash
~]$ more ifcfg-mlx4_ib0
DEVICE=mlx4_ib0
TYPE=InfiniBand
ONBOOT=yes
HWADDR=80:00:00:4c:fe:80:00:00:00:00:00:00:00:00:00:00:7b:cb:a1
BOOTPROTO=None
IPADDR=172.31.0.254
PREFIX=24
NETWORK=172.31.0.0
BROADCAST=172.31.0.255
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
MTU=65520
CONNECTED_MODE=yes
NAME=mlx4_ib0
```

The DEVICE field must match the custom name created in any udev renaming rules. The NAME entry need not match the device name. If the GUI connection editor is started, the NAME field is what is used to present a name for this connection to the user. The TYPE field must be InfiniBand in order for InfiniBand options to be processed properly. CONNECTED_MODE is either yes or no, where yes will use connected mode and no will use datagram mode for communications (see section Section 9.8.2, “Understanding IPoIB communication modes”).

For P_Key interfaces, this is a typical configuration file:

```bash
~]$ more ifcfg-mlx4_ib0.8002
DEVICE=mlx4_ib0.8002
PHYSDEV=mlx4_ib0
PKEY=yes
PKEY_ID=2
TYPE=InfiniBand
ONBOOT=yes
HWADDR=80:00:00:4c:fe:80:00:00:00:00:00:00:00:00:00:00:7b:cb:a1
BOOTPROTO=None
IPADDR=172.31.2.254
PREFIX=24
```
For all P-Key interface files, the PHYSDEV directive is required and must be the name of the parent device. The PKEY directive must be set to yes, and PKEY_ID must be the number of the interface (either with or without the 0x8000 membership bit added in). The device name, however, must be the four digit hexadecimal representation of the PKEY_ID combined with the 0x8000 membership bit using the logical OR operator as follows:

```
NAME=${PHYSDEV}.${((0x8000 | $PKEY_ID))}
```

By default, the PKEY_ID in the file is treated as a decimal number and converted to hexadecimal and then combined using the logical OR operator with 0x8000 to arrive at the proper name for the device, but users may specify the PKEY_ID in hexadecimal by prepending the standard 0x prefix to the number.

### 9.12. Testing an RDMA Network After IPoIB Is Configured

Once IPoIB is configured, it is possible to use IP addresses to specify RDMA devices. Due to the ubiquitous nature of using IP addresses and host names to specify machines, most RDMA applications use this as their preferred, or in some cases only, way of specifying remote machines or local devices to connect to.

To test the functionality of the IPoIB layer, it is possible to use any standard IP network test tool and provide the IP address of the IPoIB devices to be tested. For example, the ping command between the IP addresses of the IPoIB devices should now work.

There are two different RDMA performance testing packages included with Red Hat Enterprise Linux, qperf and perftest. Either of these may be used to further test the performance of an RDMA network.

However, when using any of the applications that are part of the perftest package, or using the qperf application, there is a special note on address resolution. Even though the remote host is specified using an IP address or host name of the IPoIB device, it is allowed for the test application to actually connect through a different RDMA interface. The reason for this is because the process of converting from the host name or IP address to an RDMA address allows any valid RDMA address pair between the two machines to be used. If there are multiple ways for the client to connect to the server, then the programs might choose to use a different path if there is a problem with the path specified. For example, if there are two ports on each machine connected to the same InfiniBand subnet, and an IP address for the second port on each machine is given, it is likely that the program will find the first port on each machine is a valid connection method and use them instead. In this case, command-line options to any of the perftest programs can be used to tell them which card and port to bind to, as was done with ibping in Section 9.7, “Testing Early InfiniBand RDMA operation”, in order to ensure that testing occurs over the specific ports required to be tested. For qperf, the method of binding to ports is slightly different. The qperf program operates as a server on one machine, listening on all devices (including non-RDMA devices). The client may connect to qperf using any valid IP address or host name for the server. Qperf will first attempt to open a data connection and run the requested test(s) over the IP address or host name given on the client command line, but if there is any problem using
that address, qperf will fall back to attempting to run the test on any valid path between the client and server. For this reason, to force qperf to test over a specific link, use the -loc_id and -rem_id options to the qperf client in order to force the test to run on a specific link.

9.13. CONFIGURE IPOIB USING A GUI

To configure an InfiniBand connection using a graphical tool, use nm-connection-editor

Procedure 9.4. Adding a New InfiniBand Connection Using nm-connection-editor

1. Enter nm-connection-editor in a terminal:

   ~]$ nm-connection-editor

2. Click the Add button. The Choose a Connection Type window appears. Select InfiniBand and click Create. The Editing InfiniBand connection window appears.

3. On the InfiniBand tab, select the transport mode from the drop-down list you want to use for the InfiniBand connection.

4. Enter the InfiniBand MAC address.

5. Review and confirm the settings and then click the Save button.


Procedure 9.5. Editing an Existing InfiniBand Connection

Follow these steps to edit an existing InfiniBand connection.

1. Enter nm-connection-editor in a terminal:

   ~]$ nm-connection-editor

2. Select the connection you want to edit and click the Edit 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 2.3.3, “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 2.3.5, “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. See the Red Hat Enterprise Linux 7 Security Guide for more information on Firewall Zones.

5. Edit the InfiniBand-specific settings by referring to the Section 9.13.1, “Configuring the InfiniBand Tab”.

Saving Your New (or Modified) Connection and Making Further Configurations
Once you have finished editing your InfiniBand connection, click the Save button to save your customized configuration.

Then, to configure:

- IPv4 settings for the connection, click the IPv4 Settings tab and proceed to Section 2.7.6, “Configuring IPv4 Settings”
  or
- IPv6 settings for the connection, click the IPv6 Settings tab and proceed to Section 2.7.7, “Configuring IPv6 Settings”.

9.13.1. Configuring the InfiniBand Tab
If you have already added a new InfiniBand connection (see Procedure 9.4, “Adding a New InfiniBand Connection Using nm-connection-editor” for instructions), you can edit the InfiniBand tab to set the parent interface and the InfiniBand ID.

Transport mode
Datagram or Connected mode can be selected from the drop-down list. Select the same mode the rest of your IPoIB network is using.

Device MAC address
The MAC address of the InfiniBand capable device to be used for the InfiniBand network traffic. This hardware address field will be pre-filled if you have InfiniBand hardware installed.

MTU
Optionally sets a Maximum Transmission Unit (MTU) size to be used for packets to be sent over the InfiniBand connection.

9.14. ADDITIONAL RESOURCES

Installed Documentation


Online Documentation

https://www.kernel.org/doc/Documentation/infiniband/ipib.txt
A description of the IPoIB driver. Includes references to relevant RFCs.
This part discusses how to set up servers normally required for networking.

**NOTE**

To monitor and administer servers through a web browser, see the *Red Hat Enterprise Linux Getting Started with Cockpit*. 


CHAPTER 10. 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.

10.1. 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.

10.2. CONFIGURING A DHCP SERVER

The dhcp package contains an Internet Systems Consortium (ISC) DHCP server. Install the package as root:

```bash
~]$ yum 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:

```bash
~]$ 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-version;/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. See Section 10.2.2, “Lease Database” for more information.

10.2.1. 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.

**IMPORTANT**

If the configuration file is changed, the changes do not take effect until the DHCP daemon is restarted with the command `systemctl restart dhcpd`.

**NOTE**

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 10.1, “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 10.1. 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
    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 10.2, “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 10.2. Range Parameter

```bash
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 10.3, “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 10.3. Static IP Address Using DHCP

```bash
host apex {
    option host-name "apex.example.com";
    hardware ethernet 00:A0:78:8E:9E:AA;
    fixed-address 192.168.1.4;
}
```

Red Hat Enterprise Linux 7 supports assigning static IP addresses to InfiniBand IPoIB interfaces. However, as these interfaces do not have a normal hardware Ethernet address, a different method of specifying a unique identifier for the IPoIB interface must be used. The standard is to use the option dhcp-client-identifier= construct to specify the IPoIB interface’s dhcp-client-identifier field. The DHCP server host construct supports at most one hardware Ethernet and one dhcp-client-identifier entry per host stanza. However, there may be more than one fixed-address entry and the DHCP server will automatically respond with an address that is appropriate for the network that the DHCP request was received on.

Example 10.4. Static IP Address Using DHCP on Multiple Interfaces

If a machine has a complex configuration, for example two InfiniBand interfaces, and P_Key interfaces on each physical interface, plus an Ethernet connection, the following static IP construct could be used to serve this configuration:

```bash
Host apex.0 {
    option host-name "apex.example.com";
    hardware ethernet 00:A0:78:8E:9E:AA;
}
```
In order to find the right `dhcp-client-identifier` for your device, you can usually use the prefix `ff:00:00:00:00:00:02:00:00:02:c9:00` and then add the last 8 bytes of the IPoIB interface (which happens to also be the 8 byte GUID of the InfiniBand port the IPoIB interface is on). On some older controllers, this prefix is not correct. In that case, we recommend using `tcpdump` on the DHCP server to capture the incoming IPoIB DHCP request and gather the right `dhcp-client-identifier` from that capture. For example:

```
$ tcpdump -vv -i mlx4_ib0
```

```
tcpdump: listening on mlx4_ib0, link-type LINUX_SLL (Linux cooked),
capture size 65535 bytes
23:42:44.131447 IP (tos 0x10, ttl 128, id 0, offset 0, flags [none],
proto UDP (17), length 328)
  0.0.0.0.bootpc > 255.255.255.255.bootps: [udp sum ok] BOOTT/DHCP,
  Request, length 300, htype 32, hlen 0, xid 0x975cb024, Flags [Broadcast]
  (0x8000)
    Vendor-rfc1048 Extensions
      Magic Cookie 0x63825363
      DHCP-Message Option 53, length 1: Discover
      Hostname Option 12, length 10: "rdma-qe-03"
      Parameter-Request Option 55, length 18:
        Subnet-Mask, BR, Time-Zone, Classless-Static-Route
        Domain-Name, Domain-Name-Server, Hostname, YD
        YS, NTP, MTU, Option 119
        Default-Gateway, Classless-Static-Route, Classless-
        Static-Route-Microsoft, Static-Route
        Option 252, NTP
        Client-ID Option 61, length 20: hardware-type 255,
        00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:02:00:21:ac:c1
```

The above dump shows the Client-ID field. The hardware-type 255 corresponds to the initial `ff:` of the ID, the rest of the ID is then quoted exactly as it needs to appear in the DHCP configuration file.

All subnets that share the same physical network should be declared within a `shared-network` declaration as shown in Example 10.5, “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 10.5. 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 10.6, “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 10.6. 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;
  }
}

**NOTE**

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.
10.2.2. 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.

10.2.3. Starting and Stopping the Server

IMPORTANT

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. For information on how to configure the daemon to start automatically at boot time, see *Red Hat Enterprise Linux System Administrator's Guide*. 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 cannot connect to the daemon through 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 10.3, “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.

### 10.3. 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 10.3.1, “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 10.3.2, “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`.

### 10.3.1. 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:

```bash
[~]# 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:

```bash
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:

```bash
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:

```bash
[~]# systemctl --system daemon-reload
[~]# systemctl restart dhcrelay
```

### 10.3.2. 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:

```bash
[~]# 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:

```bash
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:
10.4. 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:

```plaintext
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:

```
DHCP: No subnet declaration for eth0 (0.0.0.0).
DHCP: ** Ignoring requests on eth0. If this is not what
DHCP: you want, please write a subnet declaration
DHCP: in your dhcpd.conf file for the network segment
DHCP: to which interface eth1 is attached. **
DHCP:
DHCP:
DHCP: 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.

---

**WARNING**

To avoid misconfiguration when DHCP server gives IP addresses from one IP range to another physical Ethernet segment, make sure you do not enclose more subnets in a shared-network declaration.

---

### 10.4.1. 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:

```plaintext
# Example dhcpd.conf configuration 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;
}
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.

10.5. 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-version/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";
}
```

To assign a fixed-address to a client, based on the MAC address of the network interface card, use the hardware ethernet parameter:

```
host otherclient {
    hardware ethernet 01:00:80:a2:55:67;
    fixed-address6 3ffe:501:ffff:100::4321;
}
```

The configuration options in the shared-network, and group declaration for IPv6 are the same as IPv4. For more details, see the examples as demonstrated in Example 10.5, “Shared-network Declaration”, and Example 10.6, “Group Declaration”.

10.6. CONFIGURING THE RADVD DAEMON FOR IPV6 ROUTERS

The router advertisement daemon (radvd) sends router advertisement messages which are required for IPv6 stateless autoconfiguration. This allows users to automatically configure their addresses, settings, routes and choose a default router based on these advertisements. To configure the radvd daemon:
1. Install the `radvd` daemon:

   ```bash
   ~]# sudo yum install radvd
   ```

2. Set up the `/etc/radvd.conf` file. For example:

   ```plaintext
   interface eth0
   {
   AdvSendAdvert on;
   MinRtrAdvInterval 30;
   MaxRtrAdvInterval 100;
   prefix 2001:db8:1:0::/64
   {
   AdvOnLink on;
   AdvAutonomous on;
   AdvRouterAddr off;
   }
   }
   }
   ```

   **NOTE**

   If you want to additionally advertise DNS resolvers along with the router advertisements, add the `RDNSS <ip> <ip> <ip> { };` option in the `/etc/radvd.conf` file. To configure a DHCPv6 service for your subnets, you can set the `AdvManagedFlag` to `on`, so the router advertisements allow clients to automatically obtain an IPv6 address when a DHCPv6 service is available. For more details on configuring the DHCPv6 service, see Section 10.5, “DHCP for IPv6 (DHCPv6)”

3. Enable the `radvd` daemon:

   ```bash
   ~]# sudo systemctl enable radvd.service
   ```

4. Start the `radvd` daemon immediately:

   ```bash
   ~]# sudo systemctl start radvd.service
   ```

To display the content of router advertisement packages and the configured values sent by the `radvd` daemon, use the `radvdump` command:

```bash
~]# radvdump
Router advertisement from fe80::280:c8ff:feb9:cef9 (hoplimit 255)
   AdvCurHopLimit: 64
   AdvManagedFlag: off
   AdvOtherConfigFlag: off
   AdvHomeAgentFlag: off
   AdvReachableTime: 0
   AdvRetransTimer: 0
   Prefix 2002::0102:0304:f101::/64
      AdvValidLifetime: 30
      AdvPreferredLifetime: 20
      AdvOnLink: off
```
10.7. COMPARISON OF DHCPV6 TO RADVD

Dynamic Host configuration for IPv4 is mainly applied with DHCPv4. However, for IPv6 the following options are available:

- Manually
- Using the radvd daemon
- Using the DHCPv6 server

**Manually**
Manual addressing is always available. You can assign IPv6 addresses to a system using the tools described in Section 2.1.8, “Connecting to a Network Using nmcli”, Section 4.2, “Configure Bonding Using the Text User Interface, nmtui”, Section 2.2.3, “Configuring a Network Interface Using ip Commands”.

**Using the radvd Daemon**
A standards-compliant IPv6 network must provide router advertisements, thus IPv6 configuration options can be applied running the router advertisement daemon (radvd). The router advertisements provide the on-link information on which prefix is actually available locally on a physical LAN. On top of router advertisements, you can select either manual IPv6 configuration, automatic IPv6 configuration through router advertisements or the Dynamic Host Configuration Protocol (DHCPv6). For more details on configuring the radvd daemon, see Section 10.6, “Configuring the radvd daemon for IPv6 routers”.

**Using the DHCPv6 Server**
When address management is under central administration, the user can set up a DHCPv6 server. The availability of DHCPv6 is announced by flags in the router advertisement packets.

**Table 10.1. Comparison of DHCPv6 to radvd**

<table>
<thead>
<tr>
<th>DHCPv6</th>
<th>radvd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guarantee random addresses to protect privacy.</td>
<td>Provide information on a default gateway.</td>
</tr>
<tr>
<td>DHCPv6</td>
<td>radvd</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Send further network configuration options to clients. For example,</td>
<td>Map MAC addresses to IPv6 addresses</td>
</tr>
<tr>
<td>Network Time Protocol (NTP) servers, Session Initiation Protocol (SIP)</td>
<td></td>
</tr>
<tr>
<td>servers, Preboot Execution Environment (iPXE) configuration.</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**

To correctly configure a network, use DHCPv6 in conjunction with radvd, as only router advertisements provide information on a default gateway.

### 10.8. ADDITIONAL RESOURCES

- `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-version/` – Contains example files, README files, and release notes for current versions of the DHCP service.
CHAPTER 11. 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 name server, 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.

11.1. 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 name server, it usually connects to port 53. The name server then attempts to resolve the name requested. If the name server 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 name servers, called root name servers, to determine which name servers are authoritative for the name in question, and then queries them to get the requested name. Name servers configured as purely authoritative, with recursion disabled, will not do lookups on behalf of clients.

11.1.1. Name server Zones

In a DNS server, all information is stored in basic data elements called resource records (RR). Resource records are defined in RFC 1034. 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 11.1. 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 name servers through the use of zone files, which contain definitions of the resource records in each zone. Zone files are stored on primary name servers (also called master name servers), where changes are made to the files, and secondary name servers (also called slave name servers), which receive zone definitions from the primary name servers. Both primary and secondary name servers are authoritative for the zone and look the same to clients. Depending on the configuration, any name server 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.

11.1.2. Name server Types

There are two name server configuration types:

authoritative
Authoritative name servers answer to resource records that are part of their zones only. This category includes both primary (master) and secondary (slave) name servers.

recursive
Recursive name servers 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 name server 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.

11.1.3. BIND as a Name server

BIND consists of a set of DNS-related programs. It contains a name server called named, an administration utility called rndc, and a debugging tool called dig. See Red Hat Enterprise Linux System Administrator’s Guide for more information on how to run a service in Red Hat Enterprise Linux.

11.2. BIND

This section covers BIND (Berkeley Internet Name Domain), the DNS server included in Red Hat Enterprise Linux. It focuses on the structure of its configuration files, and describes how to administer it both locally and remotely.

11.2.1. 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.

The number of empty zones created for RFC 1918 prefixes has been increased, and users of BIND 9.9 and above will see the RFC 1918 empty zones both when empty-zones-enable is unspecified (defaults to yes), and when it is explicitly set to yes.

11.2.2. Configuring the named Service

When the named service is started, it reads the configuration from the files as described in Table 11.1, “The named Service Configuration Files”.

222
Table 11.1. The named Service Configuration Files

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/named.conf</td>
<td>The main configuration file.</td>
</tr>
<tr>
<td>/etc/named/</td>
<td>An auxiliary directory for configuration files that are included in the main configuration file.</td>
</tr>
</tbody>
</table>

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;
};
```
NOTE

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:

```bash
~]$ vim -c "set backupcopy=yes" /etc/named.conf
```

11.2.2.1. Installing BIND in a chroot Environment
To install **BIND** to run in a **chroot** environment, issue the following command as **root**:

```
~# yum 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
```

### 11.2.2.2. 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 11.2, “Predefined Access Control Lists”.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>any</td>
<td>Matches every IP address.</td>
</tr>
<tr>
<td>Keyword</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>localhost</td>
<td>Matches any IP address that is in use by the local system.</td>
</tr>
<tr>
<td>localnets</td>
<td>Matches any IP address on any network to which the local system is connected.</td>
</tr>
<tr>
<td>none</td>
<td>Does not match any IP address.</td>
</tr>
</tbody>
</table>

The `acl` statement can be especially useful in conjunction with other statements such as `options`. Example 11.2, “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 11.2. Using acl in Conjunction with Options**

```plaintext
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 11.3. 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:
For a list of frequently used option directives, see Table 11.3, “Commonly Used Configuration Options” below.

### Table 11.3. Commonly Used Configuration Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>allow-query</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>allow-query-cache</code></td>
<td>Specifies which hosts are allowed to query the nameserver for non-authoritative data such as recursive queries. Only <code>localhost</code> and <code>localnets</code> are allowed by default.</td>
</tr>
<tr>
<td><code>blackhole</code></td>
<td>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 <code>none</code>.</td>
</tr>
<tr>
<td><code>directory</code></td>
<td>Specifies a working directory for the named service. The default option is <code>/var/named/</code>.</td>
</tr>
<tr>
<td><code>disable-empty-zone</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>dnssec-enable</code></td>
<td>Specifies whether to return DNSSEC related resource records. The default option is <code>yes</code>.</td>
</tr>
<tr>
<td><code>dnssec-validation</code></td>
<td>Specifies whether to prove that resource records are authentic through DNSSEC. The default option is <code>yes</code>.</td>
</tr>
<tr>
<td><code>empty-zones-enable</code></td>
<td>Controls whether or not empty zones are created. Can be specified only in the options statement.</td>
</tr>
<tr>
<td><code>forwarders</code></td>
<td>Specifies a list of valid IP addresses for nameservers to which the requests should be forwarded for resolution.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| **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.  
  - **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 11.4, “Using the options Statement”.

IMPORTANT

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.

Example 11.4. 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:

```plaintext
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 11.4, “Commonly Used Options in Zone Statements”.

The zone-name attribute is particularly important, as it is the default value assigned for the
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, see Section 11.2.3, “Editing Zone Files”.

Table 11.4. Commonly Used Options in Zone Statements

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allow-query</td>
<td>Specifies which clients are allowed to request information about this zone.</td>
</tr>
<tr>
<td></td>
<td>This option overrides global allow-query option. All query requests are</td>
</tr>
<tr>
<td></td>
<td>allowed by default.</td>
</tr>
<tr>
<td>allow-transfer</td>
<td>Specifies which secondary servers are allowed to request a transfer of the</td>
</tr>
<tr>
<td></td>
<td>zone’s information. All transfer requests are allowed by default.</td>
</tr>
<tr>
<td>allow-update</td>
<td>Specifies which hosts are allowed to dynamically update information in their</td>
</tr>
<tr>
<td></td>
<td>zone. The default option is to deny all dynamic update requests.</td>
</tr>
<tr>
<td></td>
<td>Note that you should be careful when allowing hosts to update information</td>
</tr>
<tr>
<td></td>
<td>about their zone. Do not set IP addresses in this option unless the server is</td>
</tr>
<tr>
<td></td>
<td>in the trusted network. Instead, use TSIG key as described in Section 11.2.6.3, “Transaction SIgnatures (TSIG)”.</td>
</tr>
<tr>
<td>file</td>
<td>Specifies the name of the file in the named working directory that contains</td>
</tr>
<tr>
<td></td>
<td>the zone’s configuration data.</td>
</tr>
<tr>
<td>masters</td>
<td>Specifies from which IP addresses to request authoritative zone information.</td>
</tr>
<tr>
<td></td>
<td>This option is used only if the zone is defined as type slave.</td>
</tr>
<tr>
<td>notify</td>
<td>Specifies whether to notify the secondary nameservers when a zone is</td>
</tr>
<tr>
<td></td>
<td>updated. It accepts the following options:</td>
</tr>
<tr>
<td></td>
<td>• yes – The server will notify all secondary nameservers.</td>
</tr>
<tr>
<td></td>
<td>• no – The server will not notify any secondary nameserver.</td>
</tr>
<tr>
<td></td>
<td>• master-only – The server will notify primary server for the zone only.</td>
</tr>
<tr>
<td></td>
<td>• explicit – The server will notify only the secondary servers that are</td>
</tr>
<tr>
<td></td>
<td>specified in the also-notify list within a zone statement.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>type</strong></td>
<td>Specifies the zone type. It accepts the following options:</td>
</tr>
<tr>
<td>- <strong>delegation-only</strong></td>
<td>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.</td>
</tr>
<tr>
<td>- <strong>forward</strong></td>
<td>Forwards all requests for information about this zone to other nameservers.</td>
</tr>
<tr>
<td>- <strong>hint</strong></td>
<td>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.</td>
</tr>
<tr>
<td>- <strong>master</strong></td>
<td>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.</td>
</tr>
<tr>
<td>- <strong>slave</strong></td>
<td>Designates the nameserver as a slave server for this zone. Master server is specified in masters directive.</td>
</tr>
</tbody>
</table>

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 11.5, “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 11.5. 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 11.6, “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 11.6. A Zone Statement for a Secondary nameserver

```

---

231

---
11.2.2.3. 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.

See Section 11.2.4, “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.

See Section 11.2.4, “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, see the BIND 9 Administrator Reference Manual referenced in Section 11.2.8.1, “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). See Section 11.2.6.4, “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, see Section 11.2.6.1, “Multiple Views”.

11.2.2.4. 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 */
```

11.2.3. Editing Zone Files
As outlined in Section 11.1.1, “Name server 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 11.5. The named Service Zone Files

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/var/named/</td>
<td>The working directory for the named service. The nameserver is not allowed to write to this directory.</td>
</tr>
<tr>
<td>/var/named/slaves/</td>
<td>The directory for secondary zones. This directory is writable by the named service.</td>
</tr>
<tr>
<td>/var/named/dynamic/</td>
<td>The directory for other files, such as dynamic DNS (DDNS) zones or managed DNSSEC keys. This directory is writable by the named service.</td>
</tr>
<tr>
<td>/var/named/data/</td>
<td>The directory for various statistics and debugging files. This directory is writable by the named service.</td>
</tr>
</tbody>
</table>

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.

11.2.3.1. 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 11.7. 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 11.8, “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 11.8. 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 11.9. Using the $TTL Directive

$TTL 1D

11.2.3.2. 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 11.10, “Using the A Resource Record”, the requests for server1.example.com are pointed to 10.0.1.3 or 10.0.1.5.

Example 11.10. 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, and so on). The only exception are DNSSEC related records (RRSIG, NSEC, and so on) 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 11.11, “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 11.11. 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 11.12, “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 11.12. 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 11.13. 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. See Section 11.2.3.4.2, “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 *SOIGIN* directive (or the zone’s name if the *SOIGIN* 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 11.6, “Seconds compared to other time units” shows an amount of time in seconds and the equivalent time in another format.

Table 11.6. Seconds compared to other time units

<table>
<thead>
<tr>
<th>Seconds</th>
<th>Other Time Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>1M</td>
</tr>
<tr>
<td>1800</td>
<td>30M</td>
</tr>
<tr>
<td>3600</td>
<td>1H</td>
</tr>
<tr>
<td>10800</td>
<td>3H</td>
</tr>
<tr>
<td>21600</td>
<td>6H</td>
</tr>
<tr>
<td>43200</td>
<td>12H</td>
</tr>
<tr>
<td>86400</td>
<td>1D</td>
</tr>
<tr>
<td>259200</td>
<td>3D</td>
</tr>
<tr>
<td>604800</td>
<td>1W</td>
</tr>
<tr>
<td>31536000</td>
<td>365D</td>
</tr>
</tbody>
</table>

**Example 11.14. 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
```

### 11.2.3.3. 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:
11.2.3.4. Example Usage

The following examples show the basic usage of zone files.

11.2.3.4.1. A Simple Zone File

Example 11.15, “A simple zone file” demonstrates the use of standard directives and SOA values.

Example 11.15. A simple zone file

```dns
$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 through A records. Since these names do not end in a trailing period, 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:

```plaintext
zone "example.com" IN {
    type master;
    file "example.com.zone";
    allow-update { none; };
};
```

11.2.3.4.2. 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 11.16, “A reverse name resolution zone file”.

**Example 11.16. 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:

```plaintext
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.

### 11.2.4. 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:

```plaintext
rndc [option...] command [command-option]
```

#### 11.2.4.1. 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 11.7. Relevant files**

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/etc/named.conf</code></td>
<td>The default configuration file for the <code>named</code> service.</td>
</tr>
<tr>
<td><code>/etc/rndc.conf</code></td>
<td>The default configuration file for the <code>rndc</code> utility.</td>
</tr>
<tr>
<td><code>/etc/rndc.key</code></td>
<td>The default key location.</td>
</tr>
</tbody>
</table>

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 11.2.2.3, “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, see manual pages and the *BIND 9 Administrator Reference Manual* listed in Section 11.2.8, “Additional Resources”. 
IMPORTANT

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

11.2.4.2. 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

11.2.4.3. 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
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.
```

### 11.2.4.4. 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;
};
```

### 11.2.4.5. 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
```

See the options statement described in Section 11.2.2.2, “Common Statement Types” for information on how to configure this option in /etc/named.conf.

The Red Hat Enterprise Linux 7 Security Guide has a comprehensive section on DNSSEC.

### 11.2.4.6. 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 11.2.4.2, "Checking the Service Status".

### 11.2.5. 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
```

See Section 11.2.3.2, "Common Resource Records" for a list of common values to use for `type`.

#### 11.2.5.1. 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 11.17, "A sample nameserver lookup", the `dig` utility is used to display nameservers for `example.com`.

**Example 11.17. 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.
exmple.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
```

#### 11.2.5.2. 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 11.18, "A sample IP address lookup", the `dig` utility is used to display the IP address of `example.com`.

---

Networking Guide

244
Example 11.18. A sample IP address lookup

```bash
~]$ 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)
;; MSG SIZE  rcvd: 93
```

11.2.5.3. 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 11.19, “A Sample Host Name Lookup”, the `dig` utility is used to display the host name assigned to `192.0.32.10`.

Example 11.19. A Sample Host Name Lookup

```bash
~]$ 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.
```

245
11.2.6. 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.

IMPORTANT

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 11.2.8.1, “Installed Documentation”.

11.2.6.1. 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.

11.2.6.2. 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.
11.2.6.3. 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.

IMPORTANT

When communicating over an insecure network, do not rely on IP address-based authentication only.

11.2.6.4. 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 11.2.5, “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).

11.2.6.5. 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 11.3, “Commonly Used Configuration Options”.

11.2.7. 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.

**WARNING**

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.

### 11.2.8. Additional Resources

The following sources of information provide additional resources regarding BIND.

#### 11.2.8.1. 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.

### 11.2.8.2. Online Resources

**https://access.redhat.com/site/articles/770133**

A Red Hat Knowledgebase article about running BIND in a **chroot** environment, including the differences compared to Red Hat Enterprise Linux 6.


The *Red Hat Enterprise Linux 7 Security Guide* has a comprehensive section on DNSSEC.

**https://www.icann.org/namollision**

The *ICANN FAQ on domain name collision*. 
CHAPTER 12. SQUID

This chapter deals with Squid, a high performance proxy caching server for web clients. In this section, you can read on how to configure Squid, how to authenticate, and block access with Squid.

12.1. INTRODUCTION TO SQUID

Squid is a proxy web server that optimizes website operation by caching pages so that they load more quickly, thereby improving the response time for the pages that users access most frequently. Squid provides proxy and cache services for Hypertext Transport Protocol (HTTP), File Transfer Protocol (FTP), and other popular protocols. Squid is mostly used for speeding up a web server by caching repeated requests, aiding security by filtering traffic, or for limiting user access to specific pages.

Squid supports FTP, gopher, ICAP, ICP, HTCP, and HTTP data objects.

Squid consists of:

- a main server program Squid
- optional programs for custom processing and authentication
- management and client tools

12.2. INSTALLING AND RUNNING SQUID

In Red Hat Enterprise Linux, the squid package provides the Squid Caching Proxy. Run the `rpm -q squid` command to see if the squid package is installed. If not, enter the following command as the root user to install it:

```bash
# yum install squid
```

Run the `systemctl start squid` command as the root user to start Squid:

```bash
# systemctl start squid
```

Squid will start listening on port 3128 (default) on all network interfaces on the machine.

Run the `systemctl status squid` command to confirm whether Squid is running. An example output is attached.

```bash
# systemctl status squid
● squid.service - Squid caching proxy
   Loaded: loaded (/usr/lib/systemd/system/squid.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2016-04-06 13:15:05 CEST; 2min 17s ago
   [output truncated]
```

Run the `ps -eZ | grep squid` command to view the Squid processes:

```bash
# ps -eZ | grep squid
unconfined_u:system_r:squid_t:s0 2522 ? 00:00:00 squid
unconfined_u:system_r:squid_t:s0 2524 ? 00:00:00 squid
```
If you are interested in detailed performance statistics about Squid from a command line environment, use the squidclient tool that can access the Squid service and retrieve its statistics. For example, to get some general performance statistic, enter the following command on the Squid server:

```
~]# squidclient -p squid-port mgr:info
```

To stop Squid, issue this command:

```
~]# systemctl stop squid
```

**Squid Log Files**
The Squid proxy server log files are stored in the `/var/log/squid/` directory. The log file that stores information about proxied requests is the `/var/log/squid/access.log` file.

### 12.3. SQUID CONFIGURATION

To configure Squid, adjust the directives in the configuration file. Squid is normally configured according to the requirements of a given network using the command line and editing the Squid configuration file, located at `/etc/squid/squid.conf`, which contains recommended minimum configuration.

#### 12.3.1. Basic Configuration and `/etc/squid/squid.conf`

**Procedure 12.1. Basic configuration**

1. Backup the original config file.
   ```
   mv /etc/squid/squid.conf /etc/squid/squid.conf.org
   ```

2. Create a new `/etc/squid/squid.conf` file with the following contents. Edit the Access Control List (ACL) line for `mynetwork` to define source network for your local network. This is the network where client systems use the Squid server as their proxy.
   ```
   acl mynetwork src xxx.xxx.xxx.0/24
   http_access allow mynetwork
   ```

   **NOTE**
   The order of the items in the `/etc/squid/squid.conf` configuration file is important as Squid reads it from the beginning.

   ```
   acl localnet src 10.0.0.0/8
   ```
acl localnet src 172.16.0.0/12
acl localnet src 192.168.0.0/16
acl localnet src fc00::/7
acl localnet src fe80::/10
acl SSL_ports port 443
acl Safe_ports port 80
acl Safe_ports port 21
acl Safe_ports port 443
acl Safe_ports port 70
acl Safe_ports port 210
acl Safe_ports port 1025-65535
acl Safe_ports port 280
acl Safe_ports port 488
acl Safe_ports port 591
acl Safe_ports port 777
acl CONNECT method CONNECT
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localnet
http_access allow localhost
http_access deny all
http_port 3128
hierarchy_stoplist cgi-bin ?
coredump_dir /var/spool/squid
refresh_pattern ^ftp: 1440 20% 10080
refresh_pattern ^gopher: 1440 0% 1440
refresh_pattern -i (/cgi-bin/|\?) 0 0% 0
refresh_pattern . 0 20% 4320

3. Start the service and enable it on boot:

   ~# systemctl enable squid
   ~# systemctl start squid

4. If firewall is enabled, allow the Squid port.

   ~# firewall-cmd --add-port=3128/tcp --permanent

5. Configure your web browser to use the proxy. This depends on the browser you use and its version. For example, to configure Firefox version 46.0.0:

   **Procedure 12.2. Configuring Firefox with Proxy**

   1. In the Firefox menu located in the top right corner, select Preferences, from the tabs on the left, select Advanced, and then select Network from the tabs located on the top bar.

   2. In the Connection section, open Settings.

   3. In the new window that opens up, tick Manual proxy configuration and enter the proxy server that you are connecting to in the HTTP Proxy field. If you need to enter a specific port, enter it into the Port field.

   For more information on /etc/squid/squid.conf, see the squid(8) man page.
12.3.2. Configuring Squid as an HTTP proxy server

Procedure 12.3. Configuring Squid as an HTTP proxy server

1. Add the following lines to the top of the /etc/squid/squid.conf file replacing the example IP address:

   ```
   cache_dir ufs /var/spool/squid 500 16 256
   acl my_machine src 192.0.2.21 # Replace with your IP address
   http_access allow my_machine
   ```

2. Create cache directories using the following command:

   ```
   ~ ]# systemctl restart squid
   ```

   Squid now starts listening on port 3128 (default) on all network interfaces on the machine.

3. Configure your browser, for example Firefox, to use Squid as an HTTP proxy server with the host as the IP address of the machine and port 3128: for details, see Procedure 12.2, “Configuring Firefox with Proxy”

12.3.2.1. Setting the HTTP Port

The `http_port` directive is used to specify the port where Squid will listen for client connections. The default behavior is to listen on port 3128 on all the available interfaces on a machine. You can force Squid to listen on multiple interfaces and on different ports, on different interfaces.

Example 12.1. Specifying the HTTP Port

Open /etc/squid/squid.conf and edit the respective line. In this example, Squid is set up to listen on port 8080.

```
# Squid normally listens to port 3128
http_port 8080
```

The Squid server can listen on multiple ports at the same time.

Example 12.2. Specifying Two or More Ports

With the following setting, Squid listens on both port 8080 and port 9090:

```
http_port 8080 9090
```
NOTE

Do not forget to restart Squid server to apply new settings by running:

```
~]# systemctl restart squid
```

You can also specify the IP address and port combination in `/etc/squid/squid.conf`. Normally, this approach is used when you have multiple interfaces on the machine and want Squid to listen only on the interface connected to a local area network (LAN).

**Example 12.3. Setting IP addresses**

The following command instructs Squid to listen on port 3128 on the interface with the IP address 192.0.2.25:

```
http_port 192.0.2.25:3128
```

In addition, you can specify `http_port` by using `host name` and `port` combination. The host name will be translated to an IP address by Squid, which will then listen on port 8080 on that particular IP address.

```
http_port myproxy.example.com:8080
```

Another aspect of the `http_port` directive is that it can take multiple values on separate lines. The following lines will trigger Squid to listen on three different IP addresses and port combinations. This is generally helpful when you have clients in different LANs, which are configured to use different ports for the proxy server. Edit the `/etc/squid/squid.conf` file as follows:

```
http_port 192.0.2.25:8080
http_port lan1.example.com:3128
http_port lan2.example.com:8081
```

**12.3.2.2. ACLs and HTTP access control**

Access Control Lists (ACLs) are the base elements for access control and are normally used in combination with other directives, such as `http_access`, to control access to various Squid components and web resources.

**Example 12.4. Constructing an ACL for a Domain Name**

This example shows how to edit the following general instruction:

```
 acl example_site dstdomain example.com
```

as follows. Name your ACL by replacing `example_site` with any name. The type used here is `dstdomain`, which specifies that the value (the website) is a domain name.

```
 acl FB dstdomain facebook.com
```
If you need to construct an ACL covering a number of websites, you can:

- Write values on a single line:
  
  ```
  acl example_sites dstdomain example.com example.net example.org
  ```

- Write values on multiple lines in case the list of values grows significantly:
  
  ```
  acl example_sites dstdomain example.com example.net
  acl example_sites dstdomain example.org
  ```

- You can put the values in a dedicated file and then instruct Squid to read the values from that file:
  
  ```
  acl example_sites dstdomain '/etc/squid/example_sites.txt'
  ```

  The content of `/etc/squid/example_sites.txt` looks as follows:

  ```
  # Write one value (domain name) per line
  example.net
  example.org  # Temporarily remove example.org from example_sites acl
  example.com
  ```

**IMPORTANT**

ACLs must be combined with access control directives to allow or deny access to various resources. `http_access` is one such directive which is used to grant access to perform HTTP transactions through Squid:

**Controlling HTTP access using ACLs**

To allow or deny access to clients, you need to combine ACLs with the `http_access` directive.

In the `/etc/squid/squid.conf` file, edit the `http_access` directive, where `ACL_NAME` signifies the requests for which the access must be granted or revoked:

```
http_access allow|deny [!]ACL_NAME
```

**Example 12.5. Allowing or denying Access to Clients**

The following configuration setting grants access to localhost:

```
http_access allow localhost
```

This configuration denies access to localhost:

```
http_access deny localhost
```

Some ACL names start with an exclamation mark, in such case include the mark as well:

```
http_access deny !Safe_ports
```
12.4. SQUID AUTHENTICATION

For authentication, the Squid source code connects with a few authentication back ends, also called helpers, such as SMB (SMB server like Windows NT or Samba), DB (an SQL database), or LDAP (Lightweight Directory Access Protocol). Users are authenticated if Squid is configured to use proxy_auth ACLs.

Instruct Squid which authentication helper program to use with the auth_param directive in /etc/squid/squid.conf. Specify the name of the program and any command line options if necessary.

```
auth_param scheme parameter [setting]
```

Example 12.6. Adding proxy_auth ACLs

Add proxy_auth ACL entries to your Squid configuration by specifying individual user names. In this example, users named lisa, sarah, joe, and frank are allowed to use the proxy at all times. Other users are allowed only during daytime hours.

```
acl foo proxy_auth REQUIRED
acl bar proxy_auth lisa sarah frank joe
acl daytime time 08:00-17:00
http_access allow foo daytime
http_access allow bar
http_access deny all
```

12.4.1. Authentication with LDAP

In this setup, Squid uses LDAP to authenticate users before allowing them to surf the Internet. The Squid source code connects to an authentication back end (LDAP) for authentication. Users then need to enter their user name and password before they are allowed to proceed to web pages. Squid makes use of the Squid LDAP authentication helper, squidldapauth, which allows Squid to connect to an LDAP directory to validate the user name and password of basic HTTP authentication.

Edit /etc/squid/squid.conf as follows to connect Squid to ldap.example.com:

```
auth_param basic program /usr/lib64/squid/basic_ldap_auth -b "dc=example,dc=com" -f "uid=%s" -c 2 -t 2 -h ldap.example.com otherldap.example.com
```

In case you wish to authenticate Squid users on an LDAP server through a SSL/TLS secure channel, pass the -ZZ argument to the squidldapauth program.

```
auth_param basic program /usr/lib64/squid/basic_ldap_auth -v 3 -ZZ -b "dc=yourcompany,dc=com" -D uid=some-user,ou=People,dc=yourcompany,dc=com -w password -f uid=%s ldap.yourcompany.com
```

If you want to authenticate against multiple OpenLDAP servers, for example TLS and SSL, you need to specify auth_param in the /etc/squid/squid.conf file:
1. Edit the `/etc/squid/squid.conf` for TLS:

```
auth_param basic program /usr/lib64/squid/basic_ldap_auth -Z -b "dc=example,dc=com" -f "uid=%s" -c 2 -t 2 -h ldap.example.com
```

and for SSL:

```
auth_param basic program /usr/lib64/squid/basic_ldap_auth -b "dc=example,dc=com" -f "uid=%s" -c 2 -t 2 -H ldaps://ldap.example.com
```

Where

- `b` - Specifies the base DN under which the users are located.
- `f` - Specifies LDAP search filter to locate the user DN.
- `c` - Specifies timeout used when connecting to LDAP servers.
- `t` - Specifies time limit on LDAP search operations.
- `h` - Specifies the LDAP server to connect to.
- `H` - Specifies the LDAP server to connect to by LDAP URI

2. Restart the Squid service

```
~]# systemctl restart squid
```

### 12.4.2. Authentication with Kerberos

Follow the procedure to configure Squid proxy on Red Hat Enterprise Linux 7 to use Kerberos authentication. Also, as a prerequisite, first install Samba, Common Internet File System (CIFS) file server for Red Hat Enterprise Linux. For more information on installing Samba, see section Samba in the Red Hat Enterprise Linux 7 System Administrator's Guide.

**Procedure 12.4. Configure Squid on Red Hat Enterprise Linux 7 to use Kerberos authentication**

1. Configure Squid to join an Active Directory (AD) domain.

   1. Edit the `/etc/krb5.conf` file:

```
[libdefaults]
default_realm = EXAMPLE.COM
dns_lookup_kdc = no
dns_lookup_realm = no
default_keytab_name = /etc/krb5.keytab
; for Windows 2003
default_tgs_enctypes = rc4-hmac des-cbc-crc des-cbc-md5
default_tkt_enctypes = rc4-hmac des-cbc-crc des-cbc-md5
permitted_enctypes = rc4-hmac des-cbc-crc des-cbc-md5
; for Windows 2008 with AES
;   default_tgs_enctypes = aes256-cts-hmac-sha1-96 rc4-hmac des-cbc-crc des-cbc-md5
;   default_tkt_enctypes = aes256-cts-hmac-sha1-96 rc4-hmac des-cbc-crc des-cbc-md5
;   permitted_enctypes = aes256-cts-hmac-sha1-96 rc4-hmac des-cbc-crc des-cbc-md5
```
des-cbc-crc des-cbc-md5

[realms]
EXAMPLE.COM = {
    kdc = 192.168.0.1
    admin_server = 192.168.0.1
}

[domain_realm]
example.com = EXAMPLE.COM
.example.com = EXAMPLE.COM

[logging]
kdc = FILE:/var/log/kdc.log
admin_server = FILE:/var/log/kadmin.log
default = FILE:/var/log/krb5lib.log

2. Verify with the kinit command:

~]# kinit testuser1

~]# kinit administrator

3. Edit the /etc/samba/smb.conf file as follows:

[global]
workgroup = EXAMPLE
password server = 192.168.0.1
# Remember to put the realm all in CAPS:
realm = EXAMPLE.COM
security = ads
idmap uid = 16777216-33554431
idmap gid = 16777216-33554431
template shell = /bin/bash
winbind use default domain = true
winbind offline logon = false
winbind enum users = yes
winbind enum groups = yes
encrypt passwords = yes
log file = /var/log/samba/log.%m
max log size = 50
passdb backend = tdbsam
load printers = yes
cups options = raw
kerberos method = system keytab

4. Join the AD domain

~]# net ads join -U Administrator

2. Create keytab for HTTP/fqdn with the net ads keytab command

~]# kinit administrator
~]# export KRB5_KTNAME=FILE:/etc/squid/HTTP.keytab
~]# net ads keytab CREATE
~]# net ads keytab ADD HTTP

and verify the keytab file

~]# klist -k /etc/squid/HTTP.keytab

NOTE

Make sure host name is properly set in /etc/hosts file

3. Make sure the files are included in Squid.

~]# rpm -q squid
    squid-3.5.20-12.el7.x86_64

~]# rpm -ql squid-3.5.20-12.el7.x86_64 | grep kerb
    /usr/lib64/squid/ext_kerberos_ldap_group_acl
    /usr/lib64/squid/negotiate_kerberos_auth
    /usr/lib64/squid/negotiate_kerberos_auth_test
    /usr/share/man/man8/negotiate_kerberos_auth.8.gz

4. Modify /etc/squid/squid.conf as follows

    auth_param negotiate program /usr/lib64/squid/squid_kerb_auth -d -s HTTP/squid.example.com@EXAMPLE.COM
    auth_param negotiate children 10
    auth_param negotiate keep_alive on
    acl kerb_auth proxy_auth REQUIRED
    (content truncated)

    http_access allow kerb_auth
    http_access allow manager localhost
    http_access deny manager
    http_access deny !Safe_ports
    http_access deny CONNECT !SSL_ports
    http_access allow localnet
    http_access allow localhost
    http_access deny all
    (content truncated)

5. Set the .keytab file readable by the Squid process owner:

~]# chgrp squid /etc/squid/HTTP.keytab
~]# chmod g+r /etc/squid/HTTP.keytab

6. Add the below lines to the /etc/sysconfig/squid file:
7. Start the Squid service

~]# service squid start

8. Configure a Kerberos client, and configure your web browser to use the Squid proxy. Get a Kerberos ticket from Key Distribution Center (KDC).

~]# kinit testuser1

Try to access any website. The web browser should not prompt for any user name or password.

12.5. USING SQUID FOR RESTRICTING ACCESS

Mostly, Squid is used for blocking access to certain web content. Typically, either certain ports are blocked or particular web sites.

12.5.1. Restricting Access by Blocking a Port

By this method, also called port filtering, you can block a specific port number with the Squid proxy server. Doing so, you can restrict the use of some protocols, services, websites, applications. For example, to block FTP traffic, it is enough to block port 21/TCP. In the same way you can block all HTTPS sites by blocking port 443/TCP.

Procedure 12.5. Blocking Port Numbers

1. Log in as the root user and open the Squid configuration file:

~]# vi /etc/squid/squid.conf

2. Block ports using ACLs.

   acl Bad_ports port 443           #(create acl for port 443/tcp)

3. Save the changes.

4. Restart Squid to apply the new configuration:

~]# service squid reload

The Squid configuration file contains lines with acl Safe_ports port. By default, these port numbers are added as "Safe_Ports" and open for browsing.

   acl Safe_ports port 80
   acl Safe_ports port 21
   acl Safe_ports port 443
   acl Safe_ports port 70
   acl Safe_ports port 210
   acl Safe_ports port 1025-65535
acl Safe_ports port 280
acl Safe_ports port 488
acl Safe_ports port 591
acl Safe_ports port 777

You can disable each of the lines in /etc/squid/squid.conf to block the appropriate ports.

Example 12.7. Blocking port 777/tcp

To block port 777/tcp, add a hash sign in front of the respective line as follows:

```
#acl Safe_ports port 777         # multiling http
```

12.5.2. Restricting Access by Blocking Specific Sites or Addresses

Configure Squid for your network to disable access to specific sites.

Procedure 12.6. Blocking a Specific Website

1. Enable access to Squid on your network. Open the /etc/squid/squid.conf file and search for "Access Controls". Scroll down to INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS. Make sure you adapt the list to your internal IP networks from where browsing should be allowed. In this example, ACL allows access from the local networks 192.168.1.0/24 and 192.168.2.0/24.

```
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
acl our_networks src 192.168.1.0/24 192.168.2.0/24
http_access allow our_networks
```

2. Create a file containing a list of sites you want to block. Name the files, for example, /usr/local/etc/allowed-sites.squid and /usr/local/etc/restricted-sites.squid.

```
~# cat /usr/local/etc/allowed-sites.squid
www.redhat.com
fedoraproject.org

~# cat /usr/local/etc/restricted-sites.squid
www.badsites.com
illegal.com
```

These can then be used to block the restricted sites.

```
~# vi /etc/squid/squid.conf

acl our_networks src 192.168.1.0/24 192.168.2.0/24
acl GoodSites dstdomain "/usr/local/etc/allowed-sites.squid"
acl BadSites dstdomain "/usr/local/etc/restricted-sites.squid"
```
http_access allow our_networks
http_access deny BadSites
http_access allow home_network business_hours GoodSites

Save and close the file.

3. Restart the Squid proxy server:

   ~]# systemctl restart squid

4. Configure your web browser to use the DNS name or IP address of your Squid server and match the running port.

12.6. ADDITIONAL RESOURCES

- squid(8)
- squidclient(1)
- basic_ldap_auth(8)
- ext_ldap_group_acl(8)
- ext_session_acl(8)
- ext_unix_group_acl(8)
- negotiate_kerberos_auth(8)
APPENDIX A. RED HAT CUSTOMER PORTAL LABS RELEVANT TO NETWORKING

Red Hat Customer Portal Labs are tools designed to help you improve performance, troubleshoot issues, identify security problems, and optimize configuration. This appendix provides an overview of Red Hat Customer Portal Labs relevant to networking. All Red Hat Customer Portal Labs are available at https://access.redhat.com/labs/.

BRIDGE CONFIGURATION
The Bridge Configuration is designed to configure a bridged network interface for applications such as KVM using Red Hat Enterprise Linux 5.4 or later.

NETWORK BONDING HELPER
The Network Bonding Helper allows administrators to bind multiple Network Interface Controllers together into a single channel using the bonding kernel module and the bonding network interface.

Use the Network Bonding Helper to enable two or more network interfaces to act as one bonding interface.

PACKET CAPTURE SYNTAX GENERATOR
The Packet capture syntax generator helps you to capture network packets.

Use the Packet capture syntax generator to generate the tcpdump command that selects an interface and then prints information to the console. You need root access to enter the command.
## APPENDIX B. REVISION HISTORY

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Author</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10-03</td>
<td>Thu 22 Mar 2018</td>
<td>Ioanna Gkioka</td>
<td>Preparing document for 7.5 GA publication.</td>
</tr>
<tr>
<td>0.10-02</td>
<td>Mon 14 Aug 2017</td>
<td>Ioanna Gkioka</td>
<td>Async release with misc.updates</td>
</tr>
<tr>
<td>0.10-01</td>
<td>Tue 25 Jul 2017</td>
<td>Mirek Jahoda</td>
<td>Version for 7.4 GA publication.</td>
</tr>
<tr>
<td>0.9-30</td>
<td>Tue 18 Oct 2016</td>
<td>Mirek Jahoda</td>
<td>Version for 7.3 GA publication.</td>
</tr>
<tr>
<td>0.9-25</td>
<td>Wed 11 Nov 2015</td>
<td>Jana Heves</td>
<td>Version for 7.2 GA release.</td>
</tr>
<tr>
<td>0.9-15</td>
<td>Tue 17 Feb 2015</td>
<td>Christian Huffman</td>
<td>Version for 7.1 GA release.</td>
</tr>
<tr>
<td>0.9-14</td>
<td>Fri Dec 05 2014</td>
<td>Christian Huffman</td>
<td>Updated the nmtui and NetworkManager GUI sections.</td>
</tr>
<tr>
<td>0.9-12</td>
<td>Wed Nov 05 2014</td>
<td>Stephen Wadeley</td>
<td>Improved IP Networking, 802.1Q VLAN tagging, and Teaming.</td>
</tr>
<tr>
<td>0.9-11</td>
<td>Tues Oct 21 2014</td>
<td>Stephen Wadeley</td>
<td>Improved Bonding, Bridging, and Teaming.</td>
</tr>
<tr>
<td>0.9-9</td>
<td>Tue Sep 2 2014</td>
<td>Stephen Wadeley</td>
<td>Improved Bonding and Consistent Network Device Naming.</td>
</tr>
<tr>
<td>0.9-8</td>
<td>Tue July 8 2014</td>
<td>Stephen Wadeley</td>
<td>Red Hat Enterprise Linux 7.0 GA release of the Networking Guide.</td>
</tr>
<tr>
<td>0.0-0</td>
<td>Wed Dec 12 2012</td>
<td>Stephen Wadeley</td>
<td>Initialization of the Red Hat Enterprise Linux 7 Networking Guide.</td>
</tr>
</tbody>
</table>

### B.1. ACKNOWLEDGMENTS

INDEX

Symbols

/etc/named.conf (see BIND)

A

authoritative name server (see BIND)

B

Berkeley Internet Name Domain (see BIND)

BIND

additional resources, Online Resources
installed documentation, Installed Documentation

common mistakes, Common Mistakes to Avoid
configuration

acl statement, Common Statement Types
comment tags, Comment Tags
controls statement, Other Statement Types
include statement, Common Statement Types
key statement, Other Statement Types
logging statement, Other Statement Types
options statement, Common Statement Types
server statement, Other Statement Types
trusted-keys statement, Other Statement Types
view statement, Other Statement Types
zone statement, Common Statement Types

directories

/etc/named/, Configuring the named Service
/var/named/, Editing Zone Files
/var/named/data/, Editing Zone Files
/var/named/dynamic/, Editing Zone Files
/var/named/slaves/, Editing Zone Files

features

Automatic Zone Transfer (AXFR), Incremental Zone Transfers (IXFR)
DNS Security Extensions (DNSSEC), DNS Security Extensions (DNSSEC)
Incremental Zone Transfer (IXFR), Incremental Zone Transfers (IXFR)
Internet Protocol version 6 (IPv6), Internet Protocol version 6 (IPv6)
multiple views, Multiple Views
Transaction SIGnature (TSIG), Transaction SIGnatures (TSIG)
files
/etc/named.conf, Configuring the named Service, Configuring the Utility
/etc/rndc.conf, Configuring the Utility
/etc/rndc.key, Configuring the Utility

resource record, Name server Zones
types
  authoritative name server, Name server Types
  primary (master) name server, Name server Zones, Name server Types
  recursive name server, Name server Types
  secondary (slave) name server, Name server Zones, Name server Types

utilities
dig, BIND as a Name server, Using the dig Utility, DNS Security Extensions (DNSSEC)
named, BIND as a Name server, Configuring the named Service
rndc, BIND as a Name server, Using the rndc Utility

zones
$INCLUDE directive, Common Directives
$ORIGIN directive, Common Directives
$TTL directive, Common Directives
A (Address) resource record, Common Resource Records
CNAME (Canonical Name) resource record, Common Resource Records
comment tags, Comment Tags
description, Name server Zones
example usage, A Simple Zone File, A Reverse Name Resolution Zone File
MX (Mail Exchange) resource record, Common Resource Records
NS (Nameserver) resource record, Common Resource Records
PTR (Pointer) resource record, Common Resource Records
SOA (Start of Authority) resource record, Common Resource Records

bonding (see channel bonding)

C

channel bonding
  configuration, Using Channel Bonding
description, Using Channel Bonding
  parameters to bonded interfaces, Bonding Module Directives

channel bonding interface (see kernel module)

D
default gateway, Static Routes and the Default Gateway
DHCP, DHCP Servers
  additional resources, Additional Resources
  command-line options, Starting and Stopping the Server
dhcpd.conf, Configuration File
dhcpd.leases, Starting and Stopping the Server
dhcpd6.conf, DHCP for IPv6 (DHCPv6)
DHCPv6, DHCP for IPv6 (DHCPv6)
dhcrelay, DHCP Relay Agent
global parameters, Configuration File
group, Configuration File
options, Configuration File
reasons for using, Why Use DHCP?
Relay Agent, DHCP Relay Agent
server configuration, Configuring a DHCP Server
shared-network, Configuration File
starting the server, Starting and Stopping the Server
stopping the server, Starting and Stopping the Server
subnet, Configuration File
dhcpd.conf, Configuration File
dhcpd.leases, Starting and Stopping the Server
dhcrelay, DHCP Relay Agent
dig (see BIND)
DNS
  definition, DNS Servers
  (see also BIND)

Dynamic Host Configuration Protocol (see DHCP)

K

kernel module
  bonding module, Using Channel Bonding
    description, Using Channel Bonding
    parameters to bonded interfaces, Bonding Module Directives

module parameters
  bonding module parameters, Bonding Module Directives

M

Multihomed DHCP
  host configuration, Host Configuration
server configuration, Configuring a Multihomed DHCP Server

N
name server (see DNS)
named (see BIND)

NIC
binding into single channel, Using Channel Bonding

P
primary name server (see BIND)

R
recursive name server (see BIND)
resource record (see BIND)
rndc (see BIND)
root name server (see BIND)

S
secondary name server (see BIND)
Squid, Squid
static route, Static Routes and the Default Gateway