Oracle® Linux 8

Setting Up Networking

F21491-10
April 2021

Abstract

Oracle® Linux 8: Setting Up Networking provides an introduction and tasks for setting up the network on Oracle Linux 8 systems.

Document generated on: 2021-04-27 (revision: 11835)

Table of Contents

Preface

Oracle® Linux 8: Setting Up Networking provides information about configuring networking for Oracle Linux 8 systems.

Audience

This document is intended for administrators who need to configure and administer Oracle Linux networking. It is assumed that readers are familiar with web technologies and have a general understanding of using the Linux operating system, including knowledge of how to use a text editor such as `emacs` or `vim`, essential commands such as `cd`, `chmod`, `chown`, `ls`, `mkdir`, `mv`, `ps`, `pwd`, and `rm`, and using the `man` command to view manual pages.

Document Organization

The document is organized into the following chapters:

- Chapter 1, *Configuring the System's Network* describes how to configure a system's network interfaces and network routing.
- Chapter 2, *Configuring Network Servers for High Availability* describes how to configure a system's network to ensure high availability.
- Chapter 3, *Configuring Network Addressing* describes how to configure a DHCP server, DHCP client, and Network Address Translation (NAT).
- Chapter 4, *Configuring the Name Service* describes how to use BIND to set up a DNS name server.
- Chapter 5, *Configuring Network Time* describes how to configure the Chrony, Network Time Protocol (NTP), or Precision Time Protocol (PTP) daemons for setting the system time.

Related Documents

The documentation for this product is available at:

https://docs.oracle.com/en/operating-systems/linux.html.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Chapter 1 Configuring the System's Network

Table of Contents

Systems connect to the network through different network components, particularly network interface cards (NICs) and multiple configuration files. Network configuration files define how these interfaces function in the system, as well as how the interfaces interact with other devices and systems on the wider network.

This chapter describes how to configure a system's network interfaces and network routing.

1.1 About Network Interface Names

Traditionally, early kernel versions assigned names to network interface devices by assigning a prefix, which is typically based on the device driver, and a number, such as `eth0`. With the availability of different types of devices, this naming schema is no longer efficient. The names do not necessarily correspond to the chassis labels and the names themselves might be inconsistent across existing network interfaces. The inconsistency would affect embedded adapters on the system as well as add-in adapters. Consequently, server platforms with multiple network adapters could encounter problems managing these interfaces.

Oracle Linux implements a consistent naming scheme for all network interfaces through the `udev` device manager. The scheme offers the following advantages:

- The names of the devices are predictable.
- Device names persist across system reboots or after changes are made to the hardware.
- Defective hardware can easily be identified and thus replaced.

The feature that imlements consistent naming on devices is enabled in Oracle Linux 8 by default. Network interface names are based on information that is derived from the system BIOS; or, alternatively from a device's firmware, system path, or MAC address.

Network interfaces are identified by a name that combines a prefix and a suffix. The prefix depends on the type of network interface:

- Ethernet network interfaces: `en`
- Wireless local area network (LAN) interfaces: `wl`
- Wireless wide area network (WAN) interfaces: `ww`

The suffix contains any of the following information:

- An on-board index number `o`*`n`*, and thus, `eno0` .
- A hot-plug slot index number `s`*`n`*, and thus, `ens1`.

This naming schema can also include `ffunction` and `ddevice-id` that are added to the suffix.

- The bus and slot number `pbussn`, and thus. `enp0s8`.

  This naming schema can also include `ffunction` and `ddevice-id` that are added to the suffix.

- The MAC address `xMAC-addr`, and thus, `enx0217b08b`.

  Note that this naming format is not used by Oracle Linux by default. However, administrators can implement it as an option.

1.2 About Network Configuration Files

The following are additional network configuration files that you might need to configure on a system.

1.2.1 About the /etc/hosts File

The `/etc/hosts` file is a database that contains host names and their corresponding IP addresses. The system uses the file as one of the sources to perform name and address resolution. If the network uses DNS (Domain Name Service), then the file would contain at least IP address information of the loopback interface.

For more information, see the `hosts(5)` manual page.

1.2.2 About the /etc/nsswitch.conf File

The `/etc/nsswitch.conf` file configures how the system uses various databases and name resolution mechanisms. One significant parameter in the file is `hosts`, which refers to the sources for resolving names and addresses.

```
hosts:      files dns nis
```

In the example, the system uses 3 sources for name and address resolution. `files` refers to the `/etc/hosts` file. `dns` and `nis` refer to servers. With this definition, the system resolves names and IP addresses by first querying `files`. If unsuccessful, the system then queries a DNS server and a finally NIS server.

Do not modify `/etc/nsswitch.conf` directly. To revise the file, do the following:

1. Enter the changes in `/etc/authselect/user-nsswitch.conf`.
2. Run `authselect apply-changes`.

For more information, see the `nsswitch.conf(5)` and the `authselect(8)` manual pages. See also *Oracle® Linux 8: Setting Up System Users and Authentication*.

1.2.3 About the /etc/resolv.conf File

The `/etc/resolv.conf` file defines how the system uses DNS to resolve host names and IP addresses. This file contains a line specifying the search domains and up to three lines that specify the IP addresses of DNS server, for example:

```
search us.mydomain.com mydomain.com
nameserver 192.168.154.3
nameserver 192.168.154.4
nameserver 10.216.106.3
```

If your system obtains its IP address from a DHCP server, the file would also contain information obtained from DHCP.

For more information, see the `resolv.conf(5)` manual page.

1.2.4 About the /etc/sysconfig/network File

The `/etc/sysconfig/network` file specifies additional information that is valid to all network interfaces on the system, as shown in the following example:

```
NETWORKING=yes
NETWORKING_IPV6=no
HOSTNAME=host20.mydomain.com
GATEWAY=192.168.1.1
```

For more information, see `/usr/share/doc/initscripts*/sysconfig.txt`.

1.3 About Network Interface Configuration Files

Each physical and virtual network device on an Oracle Linux system has an associated configuration file named `ifcfg-`*`interface`* in the `/etc/sysconfig/network-scripts` directory, for example:

```
# cd /etc/sysconfig/network-scripts
# ls ifcfg-*
ifcfg-en0  ifcfg-en1
```

In this example, the configuration files for `en0` and `en1` are `ifcfg-en0` and `ifcfg-en1`, respectively. The system reads these configuration files at boot time to configure the network interfaces.

Note

In Oracle Linux 8, network scripts are deprecated. Previous `ifup` and `ifdown` scripts have been modified to work through the `NetworkManager` utility.

For an interface `en1` that uses the Dynamic Host Configuration Protocol (DHCP) to obtain its IP address, the corresponding `ifcfg-en1` file might appear as follows:

```
DEVICE="en1"
NM_CONTROLLED="yes"
ONBOOT=yes
USERCTL=no
TYPE=Ethernet
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
NAME="System en1"
UUID=5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03
HWADDR=08:00:27:16:C3:33
PEERDNS=yes
PEERROUTES=yes
```

If the interface is configured with a static IP address, the file would contain entries similar to the following:

```
DEVICE="en1"
NM_CONTROLLED="yes"
ONBOOT=yes
USERCTL=no
TYPE=Ethernet
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
NAME="System en1"
UUID=5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03
HWADDR=08:00:27:16:C3:33
IPADDR=192.168.1.101
```

```
NETMASK=255.255.255.0
BROADCAST=192.168.1.255
PEERDNS=yes
PEERROUTES=yes
```

The following selected configuration parameters are typically used in interface configuration files:

`BOOTPROTO`	How the interface obtains its IP address. Its values can be `dhcp` for DHCP-assigned addresses or `none` for static addresses.
`BROADCAST`	IPv4 broadcast address.
`DEFROUTE`	Whether this interface is the default route.
`DEVICE`	Name of the physical network interface device (or a PPP logical device).
`GATEWAY`*N*	IPv4 gateway address for the interface. As an interface can be associated with several combinations of IP address, network mask prefix length, and gateway address, these are numbered starting from 0.
`HWADDR`	Media access control (MAC) address of an Ethernet device.
`IPADDR`*N*	IPv4 address of the interface.
`IPV4_FAILURE_FATAL`	Whether the device is disabled if IPv4 configuration fails.
`IPV6_DEFAULTGW`	IPv6 gateway address for the interface.
`IPV6_FAILURE_FATAL`	Whether the device is disabled if IPv6 configuration fails.
`IPV6ADDR`	IPv6 address of the interface in CIDR notation, including the network mask prefix length.
`IPV6INIT`	Whether to enable IPv6 for the interface.
`MASTER`	Specifies the name of the primary bonded interface, of which this interface is a backup.
`NAME`	Name of the interface as displayed in the Network Connections GUI.
`NETWORK`	IPV4 address of the network.
`NM_CONTROLLED`	Whether the network interface device is controlled by the network management daemon, `NetworkManager`.
`ONBOOT`	Whether the interface is activated at boot time.
`PEERDNS`	Whether the `/etc/resolv.conf` file used for DNS resolution contains information obtained from the DHCP server.
`PEERROUTES`	Whether the information for the routing table entry that defines the default gateway for the interface is obtained from the DHCP server.
`PREFIX`*N*	Length of the IPv4 network mask prefix for the interface.
`SLAVE`	Specifies that this interface is a backup of a bonded interface.
`TYPE`	Interface type.
`USERCTL`	Whether users other than `root` can control the state of this interface.
`UUID`	Universally unique identifier for the network interface device.

1.4 Network Configuration Tools

Different tools are available to configure the network. All of them generally perform the same functions. You can select any tool or a combination of tools to manage the network.

- Cockpit is a web-based configuration tool for managing network configuration, including network interfaces, bonds, teams, bridges, virtual VLANs and the firewall.
- Networks Connection Editor is a graphical user interface (GUI) that is GNOME-based. The Networks Connection Editor is a subset of the GNOME settings application and is launched by using the `nm-connection-editor` command.

Note

The GNOME settings application enables you to perform other system configurations, aside from networking. To access, select the network icon at the top right of the desktop, then select Settings. Alternatively, you can click Activities, select Show Applications, and then select Settings.

- `NetworkManager` is a text-based user interface (TUI) that is launched with the `nmtui` command. While it performs the same functions as the other tools, you navigate the TUI by using the keyboard rather than the mouse device.
- `NetworkManager` command line, which consists of the `nmcli` command and its options. Other commands such as `ip` and `ethtool` complement `nmcli` in configuring and managing networks.

  For more information, see the `nmcli(1)`, `ip(8)`, and `ethtool(8)` man pages.

Note

The `NetworkManager` service and the `nmcli` command are included in the `NetworkManager` package. The Network Connections editor is included in the `nm-connection-editor` package. Depending on the type of environment you selected when you installed Oracle Linux 8, you might need to manually install the `NetworkManager` package by running the `dnf install NetworkManager` command.

1.5 Configuring Network Interfaces

The following information describes how to configure a NIC by using the tools that were previously described.

Because using Cockpit is intuitive, the following tasks focus on the other tools that are used to configure NICs on an Oracle Linux system. For a tutorial on using Cockpit to configure network interfaces, see https://docs.oracle.com/en/operating-systems/oracle-linux/8/obe-cockpit-network/index.html.

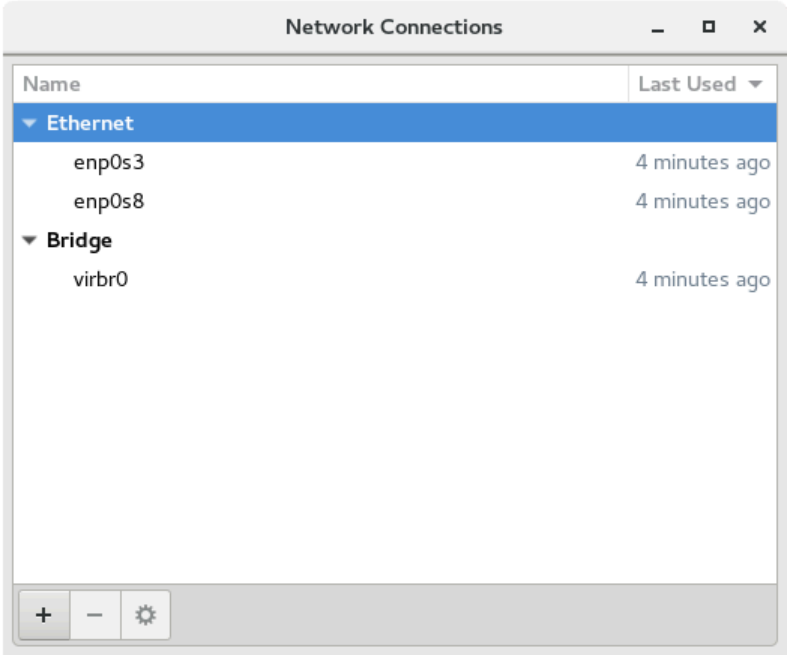
How to Use the Network Connections Editor

1. Launch the editor:

```
# nm-connection-editor
```

   `NetworkManager` detects the network devices that are on the system and lists them as well as their current states:

Figure 1.1 Network Connections



2. To add or remove a connection, use the plus (+) or minus (-) buttons located at the bottom of the editor window.

   If you add a connection, a window that prompts you for the connection type opens. Select a type, such as Ethernet, from the drop-down list then click **Create**. The Interface Editor window opens.

Note

The same window opens if you edit an existing connection.

Figure 1.2 Interface Editor

3. On each tab, enter the necessary information about the interface.
4. Click **Save** after you have completed the configuration.

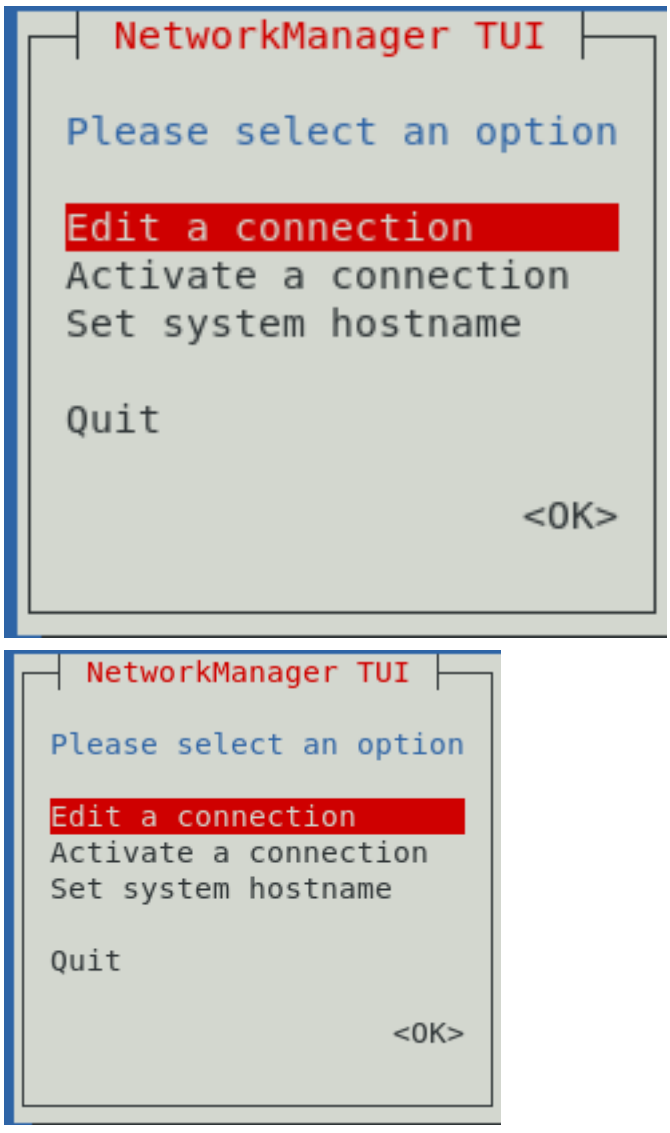
   You must specify all of the required information. Otherwise, the settings cannot be saved and the editor's background terminal window would display messages that indicate the errors.

How to Use the Text Based User Interface

1. Open the text-based user interface.

```
# nmtui
```

Figure 1.3 TUI Main Menu



The figure shows a window containing the main menu of text-based network configuration tool that enables you to edit or activate a connection or set the system hostname.

To navigate the tool, use the up and down arrow keys, then press **Enter** to make a selection

2. To add a connection, select **Edit a connection**, then click **Add**.

3. After selecting a connection type, the Edit Connection window opens.

Figure 1.4 Edit Connection

4. As an option, specify a preferred profile name, as well as the name of the device.

5. By default, IPv4 and IPv6 configurations are set to Automatic. To change the setting, select the Automatic field and press Enter. From the drop-down list, select the type of IP configuration that you want to implement, such as Manual. Then, select the corresponding **Show** field.

   The fields that are displayed depend on the type of IP configuration that is selected. For example, if you want to manually configure an IP address, selecting Show displays an address field, where you would enter an IP addresses for the interface, as the following figure illustrates.

Figure 1.5 Adding IP Addresses

6. Navigate through all of the fields on the screen to ensure that the required information is specified.
7. After you have edited the connection, select **OK**.

How to Use the Command Line

To illustrate the different uses of the `nmcli` command, this procedure describes an example of adding and configuring a new ethernet connection for the `enp0s2` device. For more information about the command, see the `nmcli(1)` manual page.

Tip

Before adding the connection, prepare the information you would for the configuration, such as the following:

- Connection name, for example, `My Work Connection`. The `nmcli` command works by referring to the connection name rather than the device name. If you do not set a connection name, then the device's name is used as the connection name.
- IP addresses (IPv4 and, if needed, IPv6)
- Gateway addresses
- Other relevant data you want to set for the connection

1. (Optional): Display the network devices on the system.

```
# nmcli device status
DEVICE  TYPE      STATE          CONNECTION
enp0s1  ethernet  connected      enp0s1
enp0s2  ethernet  disconnected    --
lo      loopback  unmanaged
```

The command shows whether a device is connected or disconnected, and whether managed or unmanaged.

2. (Optional) Display the connection information about the network devices.

```
# nmcli con show [--active]
NAME     UUID                                  TYPE      DEVICE
enp0s1   nn-nn-nn-nn-nn  ethernet  enp0s1
virbr0   nn-nn-nn-nn-nn  bridge    virbr0
mybond   nn-nn-nn-nn-nn  bond      bond0
```

The `con` subcommand is the short form of `connection`, and can be further shortened to `c`. Specifying the `--active` option would display only active devices.

Note that in the output, `NAME` represents the connection ID.

3. Add a new connection.

```
# nmcli con add {properties} [IP-info] [gateway-info
```

`properties` — The connection name as specified by the `con-name` argument, the type of connection as specified by the `type` argument, and the interface name as specified by the `ifname` argument.

IP-info	The IPv4 or IPv6 address as specified by either the `ip4` or `ip6` argument. The address must be in the format `address/netmask`. The IPv4 address can be in CIDR form, for example, `1.2.3.4/24`.
gateway-info	The gateway IPv4 or IPv6 address as specified by either the `gw4` or `gw6` argument.

For example, to add the connection with the information at the beginning of this procedure, you would type:

```
# nmcli con add type ethernet ifname enp0s2 \
   con-name "My Work Connection" ip4 192.168.5.10/24 gw4 192.168.5.2
Connection 'My Work Connection' (uuid) successfully added.
```

4. Activate the interface.

```
# nmcli con up "My Work Connection"
```

5. (Optional) Display the configuration properties of the new connection.

```
# nmcli [-o] con show "My Work Connection
connection.id:                My Work Connection
connection.uuid:              nn-nn-nn-nn-nn
connection.type:              802-3-ethernet
connection.interface-name:    enp0s2
...
IP4.ADDRESS[1]:               192.168.5.10
IP4.GATEWAY:                  192.168.5.2
...
```

Specifying the `-o` option displays only properties that have configured values.

After you have created the connection, a corresponding configuration file is created in the `/etc/sysconfig/networking-scripts` directory, for example:

```
# ls -lrt /etc/sysconfig/network-scripts/ifcfg*
-rw-r--r--. 1 root root 254 Aug 15  2017 /etc/sysconfig/network-scripts/ifcfg-lo
-rw-r--r--. 1 root root 266 Aug  6 11:03 /etc/sysconfig/network-scripts/ifcfg-My_Work_Connection
```

1.6 Configuring Network Routing

A system uses its routing table to determine which network interface to use when sending packets to remote systems. For a system with only a single interface, it is sufficient to configure the IP address of a gateway system on the local network that routes packets to other networks.

To create a default route for IPv4 network packets, include an entry for `GATEWAY` in the `/etc/sysconfig/network` file. For example, the following entry configures the IP address of the gateway system:

```
GATEWAY=192.0.2.1
```

If your system has more than one network interface, you can specify which interface should be used as follows:

```
GATEWAY=192.0.2.1
GATEWAYDEV=eth1
```

A single statement is usually sufficient to define the gateway for IPv6 packets, for example:

```
IPV6_DEFAULTGW="2001:db8:1e10:115b::2%en1"
```

Any changes that you make to `/etc/sysconfig/network` do not take effect until you restart the network service:

```
# systemctl restart NetworkManager
```

Display the routing table.

```
# ip route show
10.0.2.0/24 dev en1  proto kernel  scope link  src 10.0.2.15
default via 10.0.2.2 dev en1  proto static
```

The output in the previous example shows that packets destined for the local network (`10.0.2.0/24`) do not use the gateway. The default entry means that any packets destined for addresses outside the local network are routed through the gateway (`10.0.2.2`).

Note

If you previously used the `route` command to configure routing, note that this command is considered obsolete. Using the `ip` command is preferred over the `route` command.

You can also use the `netstat -rn` command to display this information, for example:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 en1
0.0.0.0         10.0.2.2        0.0.0.0         UG        0 0          0 en1
```

To add or delete a route from the table, use the `ip route add` or `ip route del` commands, as shown in the following example, where the entry for the static default route is replaced:

```
# ip route del default
# ip route show
10.0.2.0/24 dev en1  proto kernel  scope link  src 10.0.2.15
# ip route add default via 10.0.2.1 dev en1 proto static
# ip route show
10.0.2.0/24 dev en1  proto kernel  scope link  src 10.0.2.15
default via 10.0.2.1 dev en1  proto static
```

For example, you would use the following command to add a route to the network `10.0.3.0/24` via `10.0.3.1`, over the interface `en2`, and then delete that route:

```
# ip route add 10.0.4.0/24 via 10.0.2.1 dev en2
# ip route show
10.0.2.0/24 dev en1  proto kernel  scope link  src 10.0.2.15
10.0.3.0/24 via 10.0.3.1 dev en2
default via 10.0.2.2 dev en1  proto static
# ip route del 10.0.3.0/24
# ip route show
10.0.2.0/24 dev en1  proto kernel  scope link  src 10.0.2.15
default via 10.0.2.2 dev en1  proto static
```

The `ip route get` command is useful for querying on which route the system will send packets to reach a specified IP address, for example:

```
# ip route get 23.6.118.140
23.6.118.140 via 10.0.2.2 dev en1  src 10.0.2.15
    cache  mtu 1500 advmss 1460 hoplimit 64
```

In this example, packets to `23.6.118.140` are sent out of the `en1` interface via the gateway `10.0.2.2`.

Note that any changes you make to the routing table by using the `ip route` command do not persist across system reboots. To configure static routes that persist over system rebotts, you can create a

`route-interface` file in`/etc/sysconfig/network-scripts` for the interface. For example, you would configure a static route for the `en1` interface in a file named `route-en1`. An entry in these files can take the same format as the arguments to the `ip route add` command.

For example, you would define a default gateway entry for `en1` by creating an entry similar to the following in `route-en1`:

```
default via 10.0.2.1 dev en1
```

The following entry in `route-en2` defines a route to `10.0.3.0/24` via `10.0.3.1` over `en2`:

```
10.0.3.0/24 via 10.0.3.1 dev en2
```

Any changes that you make to a `route-interface` file do not take effect until you restart either the network service or the interface.

For more information, see the `ip(8)` and `netstat(8)` manual pages.

Chapter 2 Configuring Network Servers for High Availability

Table of Contents

For systems that provide network services to clients inside the network, network availability becomes a priority to ensure that the services are continuous and interruptions are prevented. This chapter describes features such as bonding and teaming that address the issue of high availability of the network. With virtual local area networks (VLANs), you can also organize your network such that systems with similar functions are grouped together as though the belong to their own virtual networks. This feature facilitates network management and administration.

To avail the system of these features, it must have multiple NICs. The more NICs, the better assurances of the network availability that a server can provide. This chapter also assumes that system's NICs are properly configured, as described in Chapter 1, *Configuring the System's Network*. The same tools that are described in that chapter are used to create high availability configurations.

Note

The same tutorial for basic network configuration using Cockpit provides examples for configuring the network for high availability. See https://docs.oracle.com/en/operating-systems/oracle-linux/8/obe-cockpit-network/index.html.

2.1 Working With Network Bonding

To improve network performance, one method you can adapt is to configure a network bond.

2.1.1 About Network Bonding

Multiple physical network interfaces that are present on a system and that are connected to a network switch, can can be grouped together into a single logical interface to provide higher throughput or availability. This grouping, or aggregation, of physical network interfaces is known as a network bond.

A bonded network interface can increase data throughput by load balancing or can provide redundancy by allowing failover from one component device to another. By default, a bonded interface appears like a normal network device to the kernel, but it sends out network packets over the available slave devices by using a simple round-robin scheduler. You can configure bonding module parameters in the bonded interface's configuration file to alter the behavior of load-balancing and device failover.

The network bonding driver within the kernel can be used to configure the network bond in different modes to take advantage of different bonding features, depending on your requirements and your available network infrastructure. For example, the `balance-rr` mode can be used to provide basic round-robin

load-balancing and fault tolerance across a set of physical network interfaces; while the `active-backup` mode provides simple fault tolerance for high availability configurations. Some bonding modes, such as `802.3ad`, or dynamic link aggregation, require particular hardware features and configuration on the switch that the physical interfaces connect to. Basic load-balancing modes (`balance-rr` and `balance-xor`) work with any switch that supports EtherChannel or trunking. Advanced load-balancing modes (`balance-tlb` and `balance-alb`) do not impose requirements on the switching hardware, but do require that the device driver for each component interfaces implement certain specific features such as support for `ethtool` or the ability to modify the hardware address while the device is active.

For more information on the kernel bonding driver, please see the upstream documentation at https://www.kernel.org/doc/Documentation/networking/bonding.txt or included at `/usr/share/doc/iputils-*/README.bonding`.

Note

For network configurations where systems are directly cabled together for high availability, a switch is required to support certain network interface bonding features such as automatic fail-over. Otherwise, the mechanism might not work.

2.1.2 Configuring Network Bonding

You can configure network bonding either by using the command line or the Network Connections Editor.

2.1.2.1 Using the Command Line

1. Add a bond interface using the `nmcli connection add` command.

```
# nmcli connection add type bond con-name "Bond Connection 1" ifname bond0 \
bond.options "mode=active-backup"
```

   Take note to set the bond connection name, the bond interface name, and, importantly, the bond mode option. In this example, we have set the mode to `active-backup`. If you do not set the bond connection name, then the bond interface name is used as the connection name as well.

2. Optionally configure the IP address for the bond interface using the `nmcli connection modify` command. By default the interface is configured to use DHCP, but if you require static IP addressing, you should manually configure the address. For example, to configure IPv4 settings for the bond, type:

```
# nmcli connection modify "Bond Connection 1" ipv4.addresses '192.0.2.2/24'
# nmcli connection modify "Bond Connection 1" ipv4.gateway '192.0.2.1'
# nmcli connection modify "Bond Connection 1" ipv4.dns '192.0.2.254'
# nmcli connection modify "Bond Connection 1" ipv4.method manual
```

3. Add the physical network interfaces to the bond as slave-type interfaces using the `nmcli connection add` command. For example:

```
# nmcli connection add type ethernet slave-type bond con-name bond0-if1 ifname enp1s0 master bond0
# nmcli connection add type ethernet slave-type bond con-name bond0-if2 ifname enp2s0 master bond0
```

   Give each slave a connection name, and select the interface name for each interface that you wish to add. You can get a list of available interfaces by running the `nmcli device` command. Specify the interface name of the bond to which you want to attach the slave network interfaces.

4. Start the bond interface.

```
# nmcli connection up "Bond Connection 1"
```

5. Verify that the network interfaces have been added to the bond correctly. You can check this by looking at the device list again.

```
# nmcli device
...
enp1s0   ethernet  connected  bond0-if1
enp2s0   ethernet  connected  bond0-if2
```

2.1.2.2 Using the Network Connections Editor

1. Launch the editor:

```
# nm-connection-editor
```

   The Network Connections window opens.

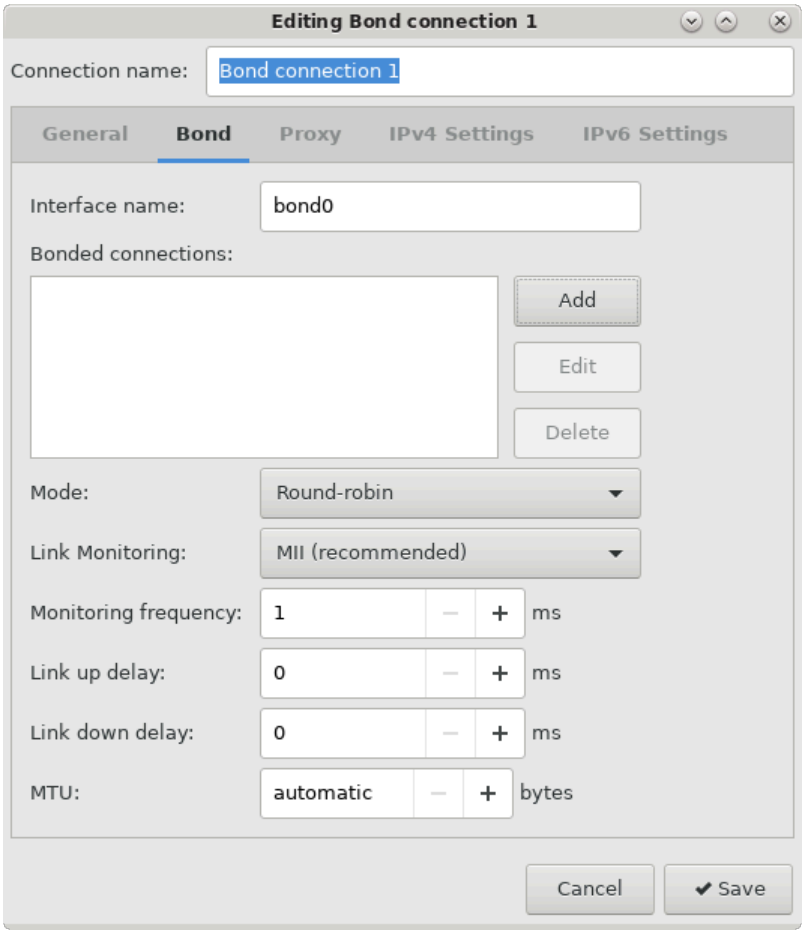
2. To add a connection, use the plus (+) button located at the bottom of the window.

   This step opens another window that prompts you for the type of connection to create.

3. From the window's drop-down list and under the Virtual section, select **Bond**, then click **Create**.

   The Network Bond Editor window opens.

   Figure 2.1 Network Bond Editor



4. Optionally configure a connection name for the bond.

5. Add physical network interfaces to the network bond by clicking on the **Add** button.

a. A new window that allows you to select the type of physical interface to add to the network bond. For example, you can select the **Ethernet** type to add an ethernet interface to the network bond. Click the **Create** button to configure the slave interface.

b. Optionally configure a name for the slave interface.

c. In the **Device** field, select the physical network interface to add as a slave to the bond. Note that if a device is already configured for networking it is not listed as available to configure within the bond.

d. Click **Save** to add the slave device to the network bond.

Repeat these steps for all of the physical network interfaces that make up the bonded interface.

6. Configure the bonding mode that you want to use for the network bond.

   Select the bonding mode from the **Mode** drop-down list. Note that some modes may require additional configuration on your network switch.

7. Configure other bond parameters such as link monitoring as required if you do not want to use the default settings.

   Additionally, if you do not intend to use DHCP for network bond IP configuration, set your IP addressing by clicking on the **IPv4** and **IPv6** tabs.

8. Click the **Save** button to save the configuration and to create the network bond.

2.1.2.3 Verifying the Network Bond Status

1. Run the following command to obtain information about the network bond with device name *bond0*:

   ```
   # cat /proc/net/bonding/bond0
   ```

   The output shows the bond configuration and status, including which bond slaves are active. The output also provides information about the status of each slave interface.

2. Temporarily disconnect the physical cable that is connected to one of the slave interfaces. There is no alternate reliable method to test link failure.

3. Check the status of the bond link as shown in the initial step for this procedure. The status of the slave interface should indicate that it is down and there is a link failure.

2.2 Working With Network Interface Teaming

Network interface teaming is similar to network interface bonding and provides a way of implementing link aggregation that is relatively maintenance-free, as well as being simpler to modify, expand, and debug, compared to bonding.

A lightweight kernel driver implements teaming and the `teamd` daemon implements load-balancing and failover schemes termed *runners*.

The following standard runners are defined:

`activebackup`	Monitors the link for changes and selects the active port that is used to send packets.
`broadcast`	Sends packets on all member ports.

`lacp`	Provides load balancing by implementing the Link Aggregation Control Protocol 802.3ad on the member ports.
`loadbalance`	In passive mode, uses the Berkeley Packet Filter (BPF) hash function to select the port that is used to send packets. In active mode, uses a balancing algorithm to distribute outgoing packets over the available ports.
`random`	Selects a port at random to send each outgoing packet.
`roundrobin`	Transmits packets over the available ports in a round-robin fashion.

For specialized applications, you can create customized runners that `teamd` can interpret. The `teamdctl` command enables you to control the operation of `teamd`.

For more information, see the `teamd.conf(5)` manual page.

2.2.1 Configuring Network Interface Teaming

You can configure a teamed interface by creating JSON-format definitions that specify the properties of the team and each of its component interfaces. The `teamd` daemon then interprets these definitions. You can use the JSON-format definitions to create a team interface by starting the `teamd` daemon manually, by editing interface definition files in `/etc/sysconfig/network-scripts`, by using the `nmcli` command, or by using the Network Configuration editor (`nm-connection-editor`). The following task describes the first of these methods.

To create a teamed interface by starting `teamd` manually:

1. Create a JSON-format definition file for the team and its component ports. For sample configurations, see the files under `/usr/share/doc/teamd/example_configs/`.

   The following example from `activebackup_ethtool_1.conf` defines an active-backup configuration where `eth1` is configured as the primary port and `eth2` as the backup port and these ports are monitored by `ethtool`.

```
{
        "device":       "team0",
        "runner":       {"name": "activebackup"},
        "link_watch":   {"name": "ethtool"},
        "ports":        {
                "eth1": {
                        "prio": -10,
                        "sticky": true
                },
                "eth2": {
                        "prio": 100
                }
        }
}
```

2. Bring down the component ports.

```
# ip link set eth1 down
# ip link set eth2 down
```

Note

Active interfaces cannot be added to a team.

3. Start an instance of the `teamd` daemon and have it create the teamed interface by reading the configuration file.

   In the following example, `/root/team_config/team0.conf` is used.

   ```
   # teamd -g -f /root/team_config/team0.conf -d
   Using team device "team0".
   Using PID file "/var/run/teamd/team0.pid"
   Using config file "/root/team_config/team0.conf"
   ```

   where the `-g` option displays debugging messages and can be omitted.

4. Set the IP address and network mask prefix length of the teamed interface.

   ```
   # ip addr add 10.0.0.5/24 dev team0
   ```

For more information, see the `teamd(8)` manual page.

2.2.2 Adding Ports to and Removing Ports from a Team

To add a port to a team, use the `teamdctl` command:

```
# teamdctl team0 port add eth3
```

To remove a port from a team:

```
# teamdctl team0 port remove eth3
```

For more information, see the `teamdctl(8)` manual page.

2.2.3 Changing the Configuration of a Port in a Team

Use the `teamdctl` command to update the configuration of a constituent port of a team, for example:

```
# teamdctl team0 port config update eth1 '{"prio": -10, "sticky": false}'
```

Enclose the JSON-format definition in single quotes and do not split it over multiple lines.

For more information, see the `teamdctl(8)` manual page.

2.2.4 Removing a Team

Use the following command to kill the `teamd` daemon:

```
# teamd -t team0 -k
```

For more information, see the `teamd(8)` manual page.

2.2.5 Displaying Information About Teams

Display the network state of the teamed interface as follows:

```
# ip addr show dev team0
7: team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 08:00:27:15:7a:f1 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.5/24 scope global team0
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe15:7af1/64 scope link
       valid_lft forever preferred_lft forever
```

Use the `teamnl` command to display information about the component ports of the team:

```
# teamnl team0 ports
 5: eth2: up 1000Mbit FD
 4: eth1: up 1000Mbit FD
```

To display the current state of the team, use the `teamdctl` command:

```
# teamdctl team0 state
setup:
  runner: activebackup
ports:
  eth1
    link watches:
      link summary: down
      instance[link_watch_0]:
        name: ethtool
        link: down
  eth2
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
runner:
  active port: em4
```

You can also use the `teamdctl` command to display the JSON configuration of the team and each of its constituent ports:

```
# teamdctl team0 config dump
{
    "device": "team0",
    "link_watch": {
        "name": "ethtool"
    },
    "mcast_rejoin": {
        "count": 1
    },
    "notify_peers": {
        "count": 1
    },
    "ports": {
        "eth1": {
            "prio": -10,
            "sticky": true
        },
        "eth2": {
            "prio": 100
        }
    },
    "runner": {
        "name": "activebackup"
    }
}
```

For more information, see the `teamdctl(8)` and `teamnl(8)` manual pages.

2.3 Configuring VLANs With Untagged Data Frames

A VLAN is a group of machines that can communicate as though they are attached to the same physical network. A VLAN enables you to group systems, regardless of their actual physical location on a LAN. In a VLAN that uses untagged data frames, you create the broadcast domain by assigning the ports of network switches to the same permanent VLAN ID or PVID (other than 1, which is the default VLAN). All of the

ports that you assign with this PVID are in a single broadcast domain. Broadcasts between devices in the same VLAN are not visible to other ports with a different VLAN, even if they exist on the same switch.

You can use the Network Settings editor or the `nmcli` command to create a VLAN device for an Ethernet interface.

To create a VLAN device from the command line:

```
# nmcli con add type vlan con-name bond0-pvid10 ifname bond0-pvid10 dev bond0 id 10
```

Running the previous command sets up the VLAN device `bond0-pvid10` with a PVID of 10 for the bonded interface `bond0`. In addition to the regular interface, `bond0`, which uses the physical LAN, you now have a VLAN device, `bond0-pvid10`, which can use untagged frames to access the virtual LAN.

Note

You do not need to create virtual interfaces for the component interfaces of a bonded interface. However, you must set the PVID on each switch port to which they connect.

You can also use the command to set up a VLAN device for a non-bonded interface, for example:

```
# nmcli con add type vlan con-name en1-pvid5 ifname en1-pvid5 dev en1 id 5
```

To obtain information about the configured VLAN interfaces, view the files in the `/proc/net/vlan` directory.

2.3.1 Creating VLAN Devices by Using the ip Command

Alternatively, you can use the `ip` command to create VLAN devices. However, note that such devices do not persist across system reboots.

For example, you would create a VLAN interface `en1.5` for `en1` with a PVID of `5` as follows:

```
# ip link add link eth1 name eth1.5 type vlan id 5
```

For more information, see the `ip(8)` manual page.

Chapter 3 Configuring Network Addressing

Table of Contents

This chapter describes how to configure a DHCP server and DHCP client, and Network Address Translation (NAT) processes.

3.1 About the Dynamic Host Configuration Protocol

The Dynamic Host Configuration Protocol (DHCP) enables client systems to obtain network configuration information from a DHCP server each time they connect to the network. The DHCP server is configured with a range of IP addresses and other network configuration parameters that clients need.

When you configure an Oracle Linux system as a DHCP client, the client daemon, `dhclient`, contacts the DHCP server to obtain the networking parameters. As DHCP is broadcast-based, the client must be on the same subnet as either a server or a relay agent. If a client cannot be on the same subnet as the server, a DHCP relay agent can be used to pass DHCP messages between subnets.

The server provides a lease for the IP address that it assigns to a client. The client can request specific terms for the lease, such as the duration. You can configure a DHCP server to limit the terms that it can grant for a lease. Provided that a client remains connected to the network, `dhclient` automatically renews the lease before it expires. You can configure the DHCP server to provide the same IP address to a client, based on the MAC address of its network interface.

The advantages of using DHCP include the following:

- Centralized management of IP addresses
- Ease of adding new clients to a network
- Reuse of IP addresses reducing the total number of IP addresses that are required
- simple reconfiguration of the IP address space on the DHCP server without needing to reconfigure each client

For more information about DHCP, see RFC 2131.

3.2 Configuring a DHCP Server

To configure an Oracle Linux system as a DHCP server:

1. If necessary, install the `dhcp-server` package:

```
# dnf install dhcp-server
```

2. Edit the `/etc/dhcp/dhcpd.conf` file to store the settings that the DHCP server can provide to the clients.

Note

This file is empty. To facilitate adding configuration information, use the template in `/usr/share/doc/dhcp-server/dhcpd.conf.example`, as shown in the following example:

```
option domain-name "mydom.org";
option domain-name-servers 192.168.2.1, 10.0.1.4;
option broadcast-address 192.168.2.255;
option routers 192.168.2.1;

subnet 192.168.2.0 netmask 255.255.255.0 {
  range 192.168.2.101 192.168.2.254;
  default-lease-time 10800;
  max-lease-time 43200;
}

host svr01 {
  hardware ethernet 80:56:3e:00:10:00;
  fixed-address 192.168.2.100;
  max-lease-time 86400;
}
```

The previous example configures the following parameters:

- Domain name (`mydom.org`)
- Subnet (`192.168.2.0/24`).
- Range of IP addresses that can be allotted (`192.168.2.101` through `192.168.2.254`).
- IP addresses of the DNS servers and the router.
- Default and maximum lease times (in seconds).
- Static IP address for the server `svr01`, which is defined by its MAC address.

The DHCP server sends the information from the `option` lines to each client when it requests a lease on an IP address. An option applies only to a subnet if you define it inside a `subnet` definition. In the example, the options are global and apply to both the `subnet` and `host` definitions. The `subnet` and `host` definitions have different settings for the maximum lease time.

See the `dhcpd(8)` and `dhcp-options(5)` manual pages for more details.

3. Check that `/var/lib/dhcpd/dhcpd.leases` exists.
4. Start the DHCP service and ensure that it restarts after a system reboot.

```
# systemctl start dhcpd
# systemctl enable dhcpd
```

3.3 Configuring a DHCP Client

The `dhcp-client` package is included in the Oracle Linux installation. To configure the system as a DHCP client:

1. Run the following command:

```
# nmcli con modify iface ipv4.method auto
```

where *iface* is the connection name.

This command changes the BOOTPROTO definition in the ifcfg-*iface* file to dhcp.

2. Edit the /etc/sysconfig/network file and add the following setting, if necessary:

```
NETWORKING=yes
```

3. To specify other options for the client, you can use different networking commands with nmcli, such as those in the following example, where the nmcli command is used to make lease renewal requests persistent, instead of the default 45 seconds:

```
# nmcli con modify iface ipv4.dhcp-timeout infinity
```

This setting directs NetworkManager to persist when requesting a lease renewal until it succeeds.

3.4 About Network Address Translation

Network Address Translation (NAT) is a process that assigns a public address to a computer or a group of computers inside a private network by using a different address scheme. The public IP address masquerades all of the requests as though they are going to one server, rather than several servers. NAT is useful for limiting the number of public IP addresses that an organization must finance. NAT also provides extra security by hiding the details of internal networks.

The netfilter kernel subsystem provides the nat table to implement NAT, in addition to its tables for packet filtering. The kernel consults the nat table whenever it handles a packet that creates a new incoming or outgoing connection.

By default, IP forwarding is enabled and your system can route packets among configured network interfaces. To turn on IP forwarding, use the following command:

```
# sysctl -a | grep ip_forward
net.ipv4.ip_forward = 1
```

You can change the status of IP forwarding on your system by using the following command:

```
# sysctl -w net.ipv4.ip_forward=0|1
```

The new status is displayed when you execute the command. To make the change persist across system reboots, add the corresponding line to the /etc/sysctl.conf file. If you switched IP forwarding off, add the following line to the file so that IP forwarding remains disabled:

```
net.ipv4.ip_forward = 0
```

You can also use the Firewall Configuration GUI (firewall-config) to configure masquerading and port forwarding. See *Oracle® Linux 8: Configuring the Firewall* for more information.

Chapter 4 Configuring the Name Service

Table of Contents

This chapter describes how to use Berkeley Internet Name Domain (BIND) to set up a Domain Name System (DNS) name server.

4.1 About DNS and BIND

DNS is a network-based service that resolves domain names to IP addresses. For a small, isolated network you can use entries in the `/etc/hosts` file to provide the name-to-address mapping. However, most networks that are connected to the Internet use DNS.

DNS is a hierarchical and distributed database . Consider the following fully qualified domain name (FQDN):

```
wiki.us.mydom.com
```

In this example, the top-level domain is `com`, `mydom` is a sub-domain of `com`, `us` is a sub-domain of `mydom`, and `wiki` is the host name.

Each of these domains are grouped into zones for administrative purposes. A DNS server, or *name server*, stores the information that is needed to resolve the component domains inside a zone. In addition, a zone's DNS server stores pointers to the other DNS servers that are responsible for resolving each sub-domain.

If an external client requests its local name server to resolve a FQDN, such as `wiki.us.mydom.com` to an IP address for which that server is not authoritative, the server queries a `root` name server for the address of a name server that is authoritative for the `.com` domain. This server then provides the IP address of another name server authoritative for the `mydom.com` domain, which in turn provides the IP address of the authoritative name server for `us.mydom.com`, and so on.

The querying process ends with the IP address for the FQDN being provided to the external client that made the request. This process is known as a recursive query, where the local name server handles each referral from an external name server to another name server on behalf of the resolver.

Iterative queries rely on the resolver being able to handle the referral from each external name server to trace the name server that is authoritative for the FQDN. Most resolvers use recursive queries and so cannot use name servers that support only iterative queries.

Most Oracle Linux releases provide the BIND implementation of DNS. The `bind` package includes the DNS server daemon (`named`), tools for working with DNS, such as `rndc`, and a number of configuration files, including the following:

`/etc/named.conf`	Contains settings for `named` and lists the location and characteristics of the zone files for your domain. Zone files are usually stored in `/var/named`.
`/etc/named.rfc1912.zones`	Contains several zone sections for resolving local loopback names and addresses.
`/var/named/named.ca`	Contains a list of the root authoritative DNS servers.

4.2 Types of Name Servers

You can configure several types of name servers by using BIND, including the following:

Master name server	Authoritative for one or more domains, a primary (master) name server maintains its zone data in several database files, and can transfer this information periodically to any backup name servers that are also configured in the zone An organization might maintain the following two primary name servers for a zone: one primary server outside the firewall to provide restricted information about the zone for publicly accessible hosts and services, and a hidden or *stealth* primary server inside the firewall that contains details of internal hosts and services.
Slave name server	Acting as a backup to a primary name server, a backup name server maintains a copy of the zone data, which it periodically refreshes from the primary server's copy.
Stub name server	A primary name server for a zone might also be configured as a stub name server that maintains information about the primary and backup name servers of child zones.
Caching-only name server	Performs queries on behalf of a client and stores the responses in a cache after returning the results to the client. It is not authoritative for any domains and the information that it records is limited to the results of queries that it has cached.
Forwarding name server	Forwards all queries to another name server and caches the results, which reduces local processing, external access, and network traffic.

In practice, a name server can be a combination of several of these types in complex configurations.

4.3 Installing and Configuring a Name Server

By default, the BIND installation enables you to configure a caching-only name server using the configuration settings that are provided in the `/etc/named.conf` file and files that it includes. The following procedure assumes that you will either use the default settings or configure new `named` configuration and zone files.

To configure a name server:

1. Install the bind package.

```
# dnf install bind
```

2. If `NetworkManager` is enabled on the system, edit the `/etc/sysconfig/network-scripts/ifcfg-`*`interface`* file, and add the following entry:

```
DNS1=127.0.0.1
```

This line causes `NetworkManager` to add the following entry to `/etc/resolv.conf` when the network service starts:

```
nameserver 127.0.0.1
```

This entry points the resolver at the local name server.

3. If you have disabled `NetworkManager`, edit the `/etc/resolv.conf` file to include the `nameserver 127.0.0.1` entry.

4. If required, modify the `named` configuration and zone files.

   See Section 4.4.1, “Configuring the named Daemon” more details.

5. Configure the system firewall to allow incoming TCP connections to port 53 and incoming UDP datagrams on port 53:

```
# firewall-cmd --zone=zone --add-port=53/tcp --add-port=53/udp
# firewall-cmd --permanent --zone=zone --add-port=53/tcp --add-port=53/udp
```

   For more information about securing the firewall, see *Oracle® Linux 8: Configuring the Firewall*.

6. Restart the `NetworkManager` service and the `named` services, and then configure the `named` service to start following system reboots:

```
# systemctl restart NetworkManager
# systemctl start named
# systemctl enable named
```

4.4 Working With DNS Configuration Files

Domains are grouped into zones that are configured through the use of zone files. Zone files store information about domains in the DNS database. Each zone file contains directives and resource records. Optional directives apply settings to a zone or instruct a name server to perform certain tasks. Resource records specify zone parameters and define information about the systems or hosts in a zone.

Examples of BIND configuration files can be found in the `/usr/share/doc/bind/sample/etc` file.

4.4.1 Configuring the named Daemon

The main configuration file for the `named` service is `/etc/named.conf`. The following example comes from the default `/etc/named.conf` file that is installed with the `bind` package and which configures a caching-only name server:

```
options {
    listen-on port 53 { 127.0.0.1; };
    listen-on-v6 port 53 { ::1; };
    directory       "/var/named";
    dump-file       "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    secroots-file   "/var/named/data/named.secroots";
    recursing-file  "/var/named/data/named.recursing";
    allow-query { localnets; };
    recursion yes;

    dnssec-enable yes;
    dnssec-validation yes;
```

```
    /* Path to ISC DLV key */
    bindkeys-file "/etc/named.iscdlv.key";

    managed-keys-directory "/var/named/dynamic";

   pid-file "/run/named/named.pid";
    session-keyfile "/run/named/session.key";

    /* https://fedoraproject.org/wiki/Changes/CryptoPolicy */
    include "/etc/crypto-policies/back-ends/bind.config";

};

logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

zone "." IN {
    type hint;
    file "named.ca";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

The `options` statement defines the global server configuration options and sets defaults for other statements.

`listen-on`	Is the port on which `named` listens for queries.
`directory`	Specifies the default directory for zone files if a relative pathname is specified.
`dump-file`	Specifies where `named` dumps its cache if it crashes.
`statistics-file`	Specifies the output file for the `rndc stats` command.
`memstatistics-file`	Specifies the output file for `named` memory-usage statistics.
`allow-query`	Specifies which IP addresses may query the server. `localnets` specifies all locally attached networks.
`recursion`	Specifies whether the name server performs recursive queries.
`dnssec-enable`	Specifies whether to use secure DNS (DNSSEC).
`dnssec-validation`	Specifies whether the name server should validate replies from DNSSEC-enabled zones.
`dnssec-lookaside`	Specifies whether to enable DNSSEC Lookaside Validation (DLV) using the key in `/etc/named.iscdlv.key` defined by `bindkeys-file`.

The `logging` section enables the logging of messages to `/var/named/data/named.run`. The `severity` parameter controls the logging level, and the `dynamic` value means that this level can be controlled by using the `rndc trace` command.

The `zone` section specifies the initial set of root servers using a hint zone. This zone specifies that `named` should consult `/var/named/named.ca` for the IP addresses of authoritative servers for the root domain (`.`).

You can add definitions to the configuration file that are appropriate to your network environment. The following example defines settings for the service as well as the top-level definitions for zones:

```
include "/etc/rndc.key";

controls {
    inet 127.0.0.1 allow { localhost; } keys { "rndc-key"; }
};

zone "us.mydom.com" {
    type master;
    file "master-data";
    allow-update { key "rndc-key"; };
    notify yes;
};

zone "mydom.com" IN {
    type slave;
    file "sec/slave-data";
    allow-update { key "rndc-key"; };
    masters {10.1.32.1;};
};

zone "2.168.192.in-addr.arpa" IN {
    type master;
    file "reverse-192.168.2";
    allow-update { key “rndc-key”; };
    notify yes;
};
```

The `include` statement allows external files to be referenced so that potentially sensitive data such as key hashes can be placed in a separate file with restricted permissions.

The `controls` statement defines access information and the security requirements that are necessary to use the `rndc` command with the `named` server:

`inet` — Specifies which hosts can run `rndc` to control named. In this example, `rndc` must be run on the local host (`127.0.0.1`).

`keys` — Specifies the names of the keys that can be used. The example specifies using the key named `rndc-key`, which is defined in `/etc/rndc.key`. Keys authenticate various actions by `named` and are the primary method of controlling remote access and administration.

The `zone` statements define the role of the server in different zones.

The following zone options are used:

`type` — Specifies that this system is the primary name server for the zone `us.mydom.com` and a backup server for `mydom.com`. `2.168.192.in-addr.arpa` is a reverse zone for resolving IP addresses to host names. See Section 4.4.3, “About Resource Records for Reverse-Name Resolution”.

`file` — Specifies the path to the zone file relative to `/var/named`. The zone file for `us.mydom.com` is stored in `/var/named/master-data` and the transferred zone data for `mydom.com` is cached in `/var/named/sec/slave-data`.

`allow-update` — Specifies that a shared key must exist on both the primary and backup name servers for a zone transfer to take place from the primary to the

backup. The following is an example record for a key in the `/etc/rndc.key` file:

```
key "rndc-key" {
    algorithm hmac-md5;
    secret "XQX8NmM41+RfbbSdcqOejg==";
};
```

You can use the `rndc-confgen -a` command to generate a key file.

`notify` — Specifies whether to notify the backup name servers when the zone information is updated.

`masters` — Specifies the primary name server for a backup name server.

For more information, see the `named.conf(5)` manual page and the BIND documentation in `/usr/share/doc/bind-version/arm`.

4.4.2 About Resource Records in Zone Files

A resource record in a zone file contains the following fields, some of which are optional, depending on the record type:

Name — Domain name or IP address.

TTL (time to live) — The maximum time that a name server caches a record before it checks whether a newer one is available.

Class — Always `IN` for the Internet.

Type — Type of record, for example:

`A` (address)	IPv4 address corresponding to a host.
`AAAA` (address)	IPv6 address corresponding to a host.
`CNAME` (canonical name)	Alias name corresponding to a host name.
`MX` (mail exchange)	Destination for email addressed to the domain.
`NS` (name server)	Fully qualified domain name of an authoritative name server for a domain.
`PTR` (pointer)	Host name that corresponds to an IP address for address-to-name lookups (reverse-name resolution).
`SOA` (start of authority)	Authoritative information about a zone, such as the primary name server, the email address of the domain's administrator, and the domain's serial number. All records following a `SOA` record relate to the

zone that it defines up to the next SOA record.

Data — Information that the record stores, such as an IP address in an A record, or a host name in a CNAME or PTR record.

The following example shows the contents of a typical zone file such as /var/named/master-data:

```
$TTL 86400        ; 1 day
@ IN SOA dns.us.mydom.com. root.us.mydom.com. (
            57 ; serial
            28800 ; refresh (8 hours)
            7200 ; retry (2 hours)
            2419200 ; expire (4 weeks)
            86400 ; minimum (1 day)
            )
              IN  NS      dns.us.mydom.com.

dns           IN  A       192.168.2.1
us.mydom.com  IN  A       192.168.2.1
svr01         IN  A       192.168.2.2
www           IN  CNAME   svr01
host01        IN  A       192.168.2.101
host02        IN  A       192.168.2.102
host03        IN  A       192.168.2.103
...
```

A comment on a line is preceded by a semicolon (;).

The $TTL directive defines the default time-to-live value for all resource records in the zone. Each resource record can define its own time-to-live value, which overrides the global setting.

The SOA record is mandatory and includes the following information:

us.mydom.com — The name of the domain.

dns.us.mydom.com. — The fully qualified domain name of the name server, including a trailing period (.) for the root domain.

root.us.mydom.com. — The email address of the domain administrator.

serial — A counter that, if incremented, tells named to reload the zone file.

refresh — The time after which a primary name server notifies backup name servers that they should refresh their database.

retry — If a refresh fails, the time that a backup name server should wait before attempting another refresh.

expire — The maximum elapsed time that a backup name server has to complete a refresh before its zone records are no longer considered authoritative and it will stop answering queries.

minimum — The minimum time for which other servers should cache information obtained from this zone.

An NS record declares an authoritative name server for the domain.

Each A record specifies the IP address that corresponds to a host name in the domain.

The CNAME record creates the alias www for svr01.

For more information, see the BIND documentation in `/usr/share/doc/bind-version/arm`.

4.4.3 About Resource Records for Reverse-Name Resolution

Forward resolution returns an IP address for a specified domain name. Reverse-name resolution returns a domain name for a specified IP address. DNS implements reverse-name resolution by using the special `in-addr.arpa` and `ip6.arpa` domains for IPv4 and IPv6.

The characteristics for a zone's `in-addr.arpa` or `ip6.arpa` domains are usually defined in `/etc/named.conf`, for example:

```
zone "2.168.192.in-addr.arpa" IN {
    type master;
    file "reverse-192.168.2";
    allow-update { key “rndc-key”; };
    notify yes;
};
```

The zone's name consists of `in-addr.arpa`, preceded by the network portion of the IP address for the domain, with its dotted quads written in reverse order.

If your network does not have a prefix length that is a multiple of 8, see RFC 2317 for the format that you should use instead.

The `PTR` records in `in-addr.arpa` or `ip6.arpa` domains define host names that correspond to the host portion of the IP address. The following example is take from the `/var/named/reverse-192.168.2` zone file:

```
$TTL 86400        ;
@ IN SOA dns.us.mydom.com. root.us.mydom.com. (
            57 ;
            28800 ;
            7200 ;
            2419200 ;
            86400 ;
            )
              IN  NS      dns.us.mydom.com.

1             IN  PTR     dns.us.mydom.com.
1             IN  PTR     us.mydom.com.
2             IN  PTR     svr01.us.mydom.com.
101           IN  PTR     host01.us.mydom.com.
102           IN  PTR     host02.us.mydom.com.
103           IN  PTR     host03.us.mydom.com.
...
```

For more information, see the BIND documentation in `/usr/share/doc/bind-version/arm`.

4.5 Administering the Name Service

The `rndc` command enables you to administer the `named` service, either locally or from a remote machine (if permitted in the `controls` section of the `/etc/named.conf` file). To prevent unauthorized access to the service, `rndc` must be configured to listen on the selected port (by default, port 953), and both named and `rndc` must have access to the same key. To generate a suitable key, use the `rndc-confgen` command:

```
# rndc-confgen -a
wrote key file "/etc/rndc.key"
```

Check the status of the `named` service as follows:

```
# rndc status
number of zones: 3
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
recursive clients: 0/1000
tcp clients: 0/100
server is up and running
```

If you modify the `named` configuration file or zone files, the `rndc reload` command instructs `named` to reload the files:

```
# rndc reload
server reload successful
```

For more information, see the `named(8)`, `rndc(8)` and `rndc-confgen(8)` manual pages.

4.6 Performing DNS Lookups

The `host` utility is recommended for performing DNS lookups. Without any arguments, the command displays a summary of its command-line arguments and options.

For example, look up the IP address for `host01`:

```
$ host host01
```

Perform a reverse lookup for the domain name that corresponds to an IP address:

```
$ host 192.168.2.101
```

Query DNS for the IP address that corresponds to a domain:

```
$ host dns.us.mydoc.com
```

Use the `-v` and `-t` options to display verbose information about records of a certain type:

```
$ host -v -t MX  www.mydom.com
Trying "www.mydom.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49643
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;www.mydom.com.   IN MX

;; ANSWER SECTION:
www.mydom.com.  135 IN CNAME www.mydom.com.acme.net.
www.mydom.com.acme.net. 1240 IN CNAME d4077.c.miscacme.net.

;; AUTHORITY SECTION:
c.miscacme.net. 2000 IN SOA m0e.miscacme.net. hostmaster.misc.com. ...

Received 163 bytes from 10.0.0.1#53 in 40 ms
```

The `-a` option, which is equivalent to the `-v`, `-t`, and `ANY` options displays all of the available records for a zone, for example:

```
$ host -a www.us.mydom.com
Trying "www.us.mydom.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40030
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
```

```
;; QUESTION SECTION:
;www.us.mydom.com.   IN ANY

;; ANSWER SECTION:
www.us.mydom.com.  263 IN CNAME www.us.mydom.acme.net.

Received 72 bytes from 10.0.0.1#53 in 32 ms
```

For more information, see the `host(1)` manual page.

Chapter 5 Configuring Network Time

Table of Contents

This chapter describes how to configure a system to use `chrony` as an implementation of the Network Time Protocol (NTP) feature, as a replacement for `ntp`. The chapter also describes the Precision Time Protocol (PTP) daemons that are used to set the system time.

5.1 About the chrony Suite

`chrony` is a feature that implements NTP to maintain accurate timekeeping on the network. In Oracle Linux 8, the `chrony` daemon service replaces `ntpd` for the management of NTP.

`chrony` has two components, which are provided in the `chrony` package:

- `chronyd` service daemon
- `chornyc` service utility

5.1.1 About the chronyd Service Daemon

The `chronyd` service daemon enables mobile systems and virtual machines to update their system clock after a period of suspension or disconnection from a network. The service can also be used to implement a simple NTP client or NTP server. As an NTP server, `chronyd` can synchronize with higher stratum NTP servers or act as a stratum 1 server using time signals that are received from the Global Positioning System (GPS) or radio broadcasts such as DCF77, MSF, or WWVB.

In an Oracle Linux 8 system, this service daemon is enabled by default

Note

`chronyd` uses NTP version 3 (RFC 1305), with features that are compatible with NTP version 4 (RFC 5905). However, `chronyd` does not support several important features of NTP version 4, nor does it support the use of PTP.

For more information, see the `chrony(1)` manual page and files in the `/usr/share/doc/chrony/` directory.

5.1.2 Using the chronyc Service Utility

The `chronyc` utility is a command that enables you to manage the `chronyd` service, display information about the service's operation, or change the service's configuration.

The command operates in two modes:

- Non-interactive mode: In this mode, you use the following syntax:

```
# chronyc subcommand
```

- Interactive mode: Typing the command by itself activates the interactive mode and displays the `chronyc>` prompt. From this prompt you can issue additional `chronyc` subcommands, for example:

```
# chronyc
chrony version version
...
200 OK
chronyc> sources
210 Number of sources = 4
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^+ service1-eth3.debrecen.hp     2   6    37    21  -2117us[-2302us] +/-   50ms
^* ns2.telecom.lt                2   6    37    21   -811us[ -997us] +/-   40ms
^+ strato-ssd.vpn0.de            2   6    37    21   +408us[ +223us] +/-   78ms
^+ kvm1.websters-computers.c     2   6    37    22  +2139us[+1956us] +/-   54ms
chronyc> sourcestats
210 Number of sources = 4
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
==============================================================================
service1-eth3.debrecen.hp   5   4   259     -0.394     41.803  -2706us   502us
ns2.telecom.lt              5   4   260     -3.948     61.422   +822us   813us
strato-ssd.vpn0.de          5   3   259      1.609     68.932   -581us   801us
kvm1.websters-computers.c   5   5   258     -0.263      9.586  +2008us   118us
chronyc> tracking
Reference ID    : 212.59.0.2 (ns2.telecom.lt)
Stratum         : 3
Ref time (UTC)  : Tue Sep 30 12:33:16 2014
System time     : 0.000354079 seconds slow of NTP time
Last offset     : -0.000186183 seconds
RMS offset      : 0.000186183 seconds
Frequency       : 28.734 ppm slow
Residual freq   : -0.489 ppm
Skew            : 11.013 ppm
Root delay      : 0.065965 seconds
Root dispersion : 0.007010 seconds
Update interval : 64.4 seconds
Leap status     : Normal
chronyc> exit
```

Note

Any changes you implement with the `chronyc` command are effective only until the next restart of the `chronyd` daemon. To make the changes permanent, you must enter these in the `/etc/chrony.conf` file. See Section 5.1.4, “Editing the chronyd Configuration File”.

For more information, see the `chronyc(1)` manual page and files in the `/usr/share/doc/chrony/` directory.

5.1.3 Configuring the chronyd Service

To configure the `chronyd` service on a system:

1. Install the `chrony` package.

```
# dnf install chrony
```

2. If remote access to the local NTP service is required, configure the system firewall to allow access to the NTP service in the appropriate zones, for example:

```
# firewall-cmd --zone=zone --add-service=ntp
success
# firewall-cmd --zone=zone --permanent --add-service=ntp
success
```

3. If necessary, start the `chronyd` service and configure it to start following a system reboot.

   Note that by default, `chrony` is enabled after installation.

```
# systemctl start chronyd
# systemctl enable chronyd
```

5.1.4 Editing the chronyd Configuration File

In the `/etc/chrony.conf` file, the default configuration assumes that the system has network access to public NTP servers with which it can synchronise.

The following example configuration for a system enables it to access three NTP servers:

```
pool NTP_server_1
pool NTP_server_2
pool NTP_server_3
driftfile /var/lib/chrony/drift
keyfile /etc/chrony.keys
...
```

To configure `chronyd` to act as an NTP server for a specified client or subnet, use the `allow` directive, as shown in bold in the following example:

```
pool NTP_server_1
pool NTP_server_2
pool NTP_server_3
allow 192.168.2/24
driftfile /var/lib/chrony/drift
keyfile /etc/chrony.keys
...
```

To create keys for an authentication mechanism based on public key cryptography, use the `chronyc keygen` command.

Note

`Autokey` in `ntp` is no longer supported in `chrony`.

If a system has only intermittent access to NTP servers, the following configuration might be appropriate:

```
pool NTP_server_1 offline
pool NTP_server_2 offline
pool NTP_server_3 offline
driftfile /var/lib/chrony/drift
keyfile /etc/chrony.keys
...
```

If you specify the `offline` keyword, `chronyd` does not poll the NTP servers until it receives communication that network access is available. You can use the `chronyc online` and `chronyc offline` commands to inform `chronyd` of the state of network access.

For a more information about the configuration file and its directives, see the `chrony.conf(5)` manual page.

5.1.5 Converting From ntp to chrony

The following table shows file, command, and terminology equivalents between `ntp` and `chrony`.

ntp	chrony
`/etc/ntp.conf`	`/etc/chrony.conf`
`/etc/ntp/keys`	`/etc/chrony.keys`
`ntpd`	`chronyd`
`ntpq` command	`chronyc` command
`ntpd.service`	`chronyd.service`
`ntp-wait.service`	`chrony-wait.service`
`ntpdate` and `sntp` utilities	`chronyd -q` and `chronyd -t` commands

The `ntpstat` utility which is available in the `ntpstat` package, now supports `chronyd`. Thus, you can still use the utility in Oracle Linux 8. The command generates output that is similar to when it is used with `ntp`.

The `/usr/share/doc/chrony/ntp2chrony.py` script is available to help convert existing `ntp` configuration to `chrony`, for example:

```
# python3 /usr/share/doc/chrony/ntp2chrony.py -b -v
```

The script supports the conversion of the most common directives in `/etc/ntp.conf` to `chrony`. In the example, the `-b` and `-v` options specify creating backup configuration files before converting and displaying verbose messages during the migration process, respectively.

To list the different options that you can use with the script, type the following command:

```
# python3 /usr/share/doc/chrony/ntp2chrony.py --help
```

5.2 About PTP

PTP enables you to synchronise system clocks on a LAN to a higher accuracy than NTP. Provided that network drivers support either hardware or software time stamping, a PTP clock can use the time stamps in PTP messages to compensate for propagation delays across a network. Software time stamping allows PTP to synchronise systems to within a few tens of microseconds. With hardware time stamping, PTP can synchronise systems to within a few tenths of a microsecond. If you require high-precision time synchronization of systems, use hardware time stamping.

A typical PTP configuration on an enterprise local area network consists of:

- One or more *grandmaster clock* systems.

  A grandmaster clock is typically implemented as specialized hardware that can use high-accuracy GPS signals or lower-accuracy code division multiple access (CDMA) signals, radio clock signals, or NTP as a time reference source. If several grandmaster clocks are available, the best master clock (BMC) algorithm selects the grandmaster clock based on the settings of their `priority1`, `clockClass`, `clockAccuracy`, `offsetScaledLogVariance`, and `priority2` parameters and their unique identifier, in that order.

- Several *boundary clock* systems.

Each boundary clock is backed up to a grandmaster clock on one subnetwork and relays PTP messages to one or more additional subnetworks. A boundary clock is usually implemented as a function of a network switch.

- Multiple *slave clock* systems.

  Each slave clock on a subnetwork is backed up to a boundary clock, which acts as the *master clock* for that slave clock.

For a simpler configuration, set up a single grandmaster clock and multiple slave clocks on the same network segment and thus eliminates any need for an intermediate layer of boundary clocks.

Grandmaster and slave clock systems that use only one network interface for PTP are termed *ordinary clocks*.

Boundary clocks require at least two network interfaces for PTP: one interface acts a slave to a grandmaster clock or a higher-level boundary clock; the other interfaces act as masters to slave clocks or lower-level boundary clocks.

Synchronization of boundary and slave clock systems is achieved by sending time stamps in PTP messages. By default, PTP messages are sent in UDPv4 datagrams. It is also possible to configure PTP to use UDPv6 datagrams or Ethernet frames as its transport mechanism.

To use PTP on a system, the driver for at least one of the system's network interfaces must support either software or hardware time stamping. To find out whether the driver for a network interface supports time stamping, use the `ethtool` command:

```
# ethtool -T en1
Time stamping parameters for en1:
Capabilities:
 hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
 software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
 hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
 software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
 software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
 hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
...
```

The output in the example shows that the `en1` interface supports both hardware and software time stamping capabilities.

With software time stamping, `ptp4l` synchronises the system clock to an external grandmaster clock.

If hardware time stamping is available, `ptp4l` can synchronise the PTP hardware clock to an external grandmaster clock. In this case, you use the `phc2sys` daemon to synchronise the system clock with the PTP hardware clock.

5.2.1 Configuring the PTP Service

To configure the PTP service on a system:

1. Install the `linuxptp` package.

   ```
   # dnf install linuxptp
   ```

2. Edit `/etc/sysconfig/ptp4l` and define the start-up options for the `ptp4l` daemon.

   Grandmaster clocks and slave clocks require that you define only one interface.

For example, to use hardware time stamping with interface `en1` on a slave clock:

```
OPTIONS="-f /etc/ptp4l.conf -i en1 -s"
```

To use software time stamping instead of hardware time stamping, specify the `-S` option:

```
OPTIONS="-f /etc/ptp4l.conf -i en1 -S -s"
```

Note

The `-s` option specifies that the clock operates only as a slave (`slaveOnly` mode). Do not specify this option for a grandmaster clock or a boundary clock.

For a grandmaster clock, omit the `-s` option, for example:

```
OPTIONS="-f /etc/ptp4l.conf -i en1"
```

A boundary clock requires that you define at least two interfaces, for example:

```
OPTIONS="-f /etc/ptp4l.conf -i en1 -i en2"
```

You might need to edit the `/etc/ptp4l.conf` file to make further adjustments to the configuration of `ptp4l`, for example:

- For a grandmaster clock, set the value of the `priority1` parameter to a value between 0 and 127, where lower values have higher priority when the BMC algorithm selects the grandmaster clock. For a configuration that has a single grandmaster clock, a value of 127 is suggested.
- If you set the value of `summary_interval` to an integer value *N* instead of 0, `ptp4l` writes summary clock statistics to `/var/log/messages` every 2^N seconds instead of every second (2^0 = 1). For example, a value of 10 would correspond to an interval of 2^{10} or 1024 seconds.
- The `logging_level` parameter controls the amount of logging information that `ptp4l` records. The default value of `logging_level` is `6`, which corresponds to `LOG_INFO`. To turn off logging completely, set the value of `logging_level` to `0`. Alternatively, specify the `-q` option to `ptp4l`.

See the `ptp4l(8)` manual page.

3. Configure the system firewall to allow access by PTP event and general messages to UDP ports 319 and 320 in the appropriate zone, for example:

```
# firewall-cmd --zone=zone --add-port=319/udp --add-port=320/udp
success
# firewall-cmd --permanent --zone=zone --add-port=319/udp --add-port=320/udp
success
```

4. Start the `ptp4l` service and configure it to start following a system reboot.

```
# systemctl start ptp4l
# systemctl enable ptp4l
```

5. To configure `phc2sys` on a clock system that uses hardware time stamping:

   a. Edit the `/etc/sysconfig/phc2sys` file and define the start-up options for the `phc2sys` daemon.

   On a boundary clock or slave clock, synchronise the system clock with the PTP hardware clock that is associated with the slave network interface, for example:

```
OPTIONS="-c CLOCK_REALTIME -s en1 -w"
```

Note

The slave network interface on a boundary clock is the one that it uses to communicate with the grandmaster clock.

The `-w` option specifies that `phc2sys` waits until `ptp4l` has synchronised the PTP hardware clock before attempting to synchronise the system clock.

On a grandmaster clock, which derives its system time from a reference time source such as GPS, CDMA, NTP, or a radio time signal, synchronise the network interface's PTP hardware clock from the system clock, for example:

```
OPTIONS="-c en1 -s CLOCK_REALTIME -w"
```

See the `phc2sys(8)` manual page.

b. Start the `phc2sys` service and configure it to start following a system reboot.

```
# systemctl start phc2sys
# systemctl enable phc2sys
```

You can use the `pmc` command to query the status of `ptp4l` operation. The following example shows the results of running `pmc` on a slave clock system that is directly connected to the grandmaster clock system without any intermediate boundary clocks:

```
# pmc -u -b 0 'GET TIME_STATUS_NP'
sending: GET TIME_STATUS_NP
 080027.fffe.7f327b-0 seq 0 RESPONSE MANAGEMENT TIME_STATUS_NP
  master_offset              -98434
  ingress_time               1412169090025854874
  cumulativeScaledRateOffset +1.000000000
  scaledLastGmPhaseChange    0
  gmTimeBaseIndicator        0
  lastGmPhaseChange          0x0000'0000000000000000.0000
  gmPresent                  true
  gmIdentity                 080027.fffe.d9e453
# pmc -u -b 0 'GET CURRENT_DATA_SET'
sending: GET CURRENT_DATA_SET
 080027.fffe.7f327b-0 seq 0 RESPONSE MANAGEMENT CURRENT_DATA_SET
  stepsRemoved     1
  offsetFromMaster  42787.0
  meanPathDelay    289207.0
```

This output includes the following useful information:

`gmIdentity`	The unique identifier of the grandmaster clock, which is based on the MAC address of its network interface.
`gmPresent`	Whether an external grandmaster clock is available. This value is displayed as `false` on the grandmaster clock itself.
`meanPathDelay`	An estimate of how many nanoseconds by which synchronization messages are delayed.
`offsetFromMaster`	The most recent measurement of the time difference in nanoseconds relative to the grandmaster clock.
`stepsRemoved`	The number of network steps between this system and the grandmaster clock.

For more information, see the `phc2sys(8)`, `pmc(8)`, and `ptp4l(8)` manual pages, and IEEE 1588.

5.2.2 Using PTP as a Time Source for NTP

To make the PTP-adjusted system time on an NTP server available to NTP clients, include the following entries in the `/etc/chrony.conf` file on the NTP server:

```
server    127.127.1.0
fudge     127.127.1.0 stratum 0
```

These entries define the local system clock as the time reference.

Note

Do not configure any additional `server` lines in the file.

Chapter 6 Configuring Virtual Private Networks

Table of Contents

This chapter describes how to configure a virtual private networks (VPN) by using IPsec.

VPNs have long been used to enable remote access between endpoints and provide site-to-site connections that simulate a larger network beyond the limitations of a deployed physical network. Implementations of VPNs have varied over time by using different technologies and applications, such as OpenVPN and so on. This document is specific to implementing an IPsec VPN by using Libreswan.

For more information about Libreswan, go to https://libreswan.org.

6.1 Installing Libreswan

To configure an IPsec VPN with Libreswan, download the package as follows:

1. Ensure that the AppStream repository is enabled.
2. Install Libreswan.

```
$ sudo dnf install -y libreswan
```

3. Start `ipsec` as a persistent service.

```
$ sudo systemctl enable ipsec --now
```

4. Add the `ipsec` service to the firewall service.

```
$ sudo firewall-cmd --add-service="ipsec"
$ sudo firewall-cmd --runtime-to-permanent
```

6.2 Configuring VPN

VPN configurations range from simple setups such as one between hosts to complex ones that involve entire sites.

6.2.1 Creating a Host to Host Connection

Regardless of the types of VPN connections that you want to configure, a common but important step involves obtaining RSA keys that would enable connections between endpoints.

On a host-to-host connection, for example, do the following:

1. Generate an RSA key pair by running the following command.

```
$ sudo ipsec newhostkey --output /etc/ipsec.d/hostkey.secrets
Generated RSA key pair with CKAID 6e6e724aa180b071128632dc09c7d2b25a852d7e was stored in the NSS databa
```

The command generates an RSA key pair with a specific `ckaid` value.

You must run the command on **both** hosts.

2. On the first host, display the `leftrsasigkey` key.

Note

In `libreswan`, the *left* and *right* designations typically refer to the local host and the remote host, respectively. However, because both hosts are peers, the designations can be used interchangeably.

```
$ sudo ipsec showhostkey --list
< 1> RSA keyid: AwEAAaxdf ckaid: 6e6e724aa180b071128632dc09c7d2b25a852d7e

$ sudo ipsec showhostkey --left --ckaid 6e6e724aa180b071128632dc09c7d2b25a852d7e
  # rsakey AwEAAaxdf
  leftrsasigkey=0sAwEAAaxdfaCPrZ72pAm1kjvhAQHHLn3Wg3gAu1Z0U+3FWeh7FN+bHtfy
...
9f8=
```

3. On the second host, display the `rightrsasigkey` key.

```
$ sudo ipsec showhostkey --list
< 1> RSA keyid: AwEAAdSSY ckaid: 5dddc2334515702c3a605bc00daed1e44e18767d

$ ipsec showhostkey --right --ckaid 5dddc2334515702c3a605bc00daed1e44e18767d
  # rsakey AwEAAb1nC
  rightrsasigkey=0sAwEAAdSSYrNO2QOY8RXgLlJZilBokPb9cFzCbU+VYY7eFcoZMmVWPVI
...
zu+/7BE5kjXHAAI1fvYha+CFbuh6KYAlpoHvX81ALusfQs+6wwTsde5jlfcrXNlqX
```

4. On each host, create and edit a configuration file in `/etc/ipsec.d`, for example, `host2host.conf`, with the following entries:

```
conn tunnel-name
    leftid=@host1-tunnel-id
    left=host1-IPaddress
    leftrsasigkey=host1-leftrsasigkey
    rightid=@host2-tunnel-id
    right=host2-IPaddress
    rightrsasigkey=host2-rightrsasigkey
    authby=rsasig
```

For more information about the configuration file and other parameters you can set, see the `ipsec.conf(5)` man page.

5. Restart the IPsec service.

```
$ sudo systemctl restart ipsec
```

6. Start `libreswan`.

```
$ sudo ipsec setup start
```

7. Load the VPN tunnel connection.

```
$ sudo ipsec auto --add tunnel-name
```

8. Establish the tunnel connection.

```
$ sudo ipsec auto --up tunnel-name
```

9. Start the tunnel automatically when the `ipsec` service is started by adding the following line to the configuration file:

```
auto=start
```

6.2.2 Creating a Site to Site Connection

A VPN connection between sites means that a connection is established between two networks. When you configure a pair of hosts for this type of connection, the hosts effectively become gateways through which traffic can enter or exit to access other hosts in the network.

To configure a site to site VPN, a configured host to host VPN must already be existing and operational as described in Section 6.2.1, “Creating a Host to Host Connection”. To proceed with configuring a connection between sites, follow these steps:

1. Create a copy of the host to host configuration file to serve as the configuration file for the site to site connection, for example:

```
$ sudo cp /etc/ipsec.d/host2host.conf /etc/ipsec.d/site2site.conf
```

   Copies must exist in both hosts.

2. Edit the new configuration file by adding subnet information, for example:

```
conn subnet-name
  also=tunnel-name
  leftsubnet=subnet1-IP
  rightsubnet=subnet2-IP
  auto=start

host connection information...
```

Note

The subnets can be in CIDR notation.

6.3 Verifying the Status of VPN Services

To check if the `ipsec` service is running, type this command:

```
$ sudo systemctl status ipsec
ipsec.service - Internet Key Exchange (IKE) Protocol Daemon for IPsec
   Loaded: loaded (/usr/lib/systemd/system/ipsec.service; enabled; vendor prese>
   Active: active (running) since Mon 2021-04-26 02:27:39 PDT; 7h ago
     Docs: man:ipsec(8)
           man:pluto(8)
           man:ipsec.conf(5)
...
```

To check the correctness of the `ipsec` configuration, type this command:

```
$ sudo ipsec verify
Verifying installed system and configuration files

Version check and ipsec on-path                         [OK]
Libreswan 3.32 (netkey) on 5.4.17-2036.104.5.el8uek.x86_64
Checking for IPsec support in kernel                    [OK]
 NETKEY: Testing XFRM related proc values
         ICMP default/send_redirects                    [NOT DISABLED]

  Disable /proc/sys/net/ipv4/conf/*/send_redirects or XFRM/NETKEY will act on or
```

```
 cause sending of bogus ICMP redirects!

        ICMP default/accept_redirects                  [NOT DISABLED]

  Disable /proc/sys/net/ipv4/conf/*/accept_redirects or XFRM/NETKEY will act on
or cause sending of bogus ICMP redirects!

        XFRM larval drop                               [OK]
Pluto ipsec.conf syntax                                [OK]
Checking rp_filter                                     [ENABLED]
 /proc/sys/net/ipv4/conf/all/rp_filter                 [ENABLED]
  rp_filter is not fully aware of IPsec and should be disabled
Checking that pluto is running                         [OK]
 Pluto listening for IKE on udp 500                    [OK]
 Pluto listening for IKE/NAT-T on udp 4500             [OK]
 Pluto ipsec.secret syntax                             [OK]
Checking 'ip' command                                  [OK]
Checking 'iptables' command                            [OK]
Checking 'prelink' command does not interfere with FIPS [OK]
Checking for obsolete ipsec.conf options               [OK]
```

To test the tunnel connections, install the `tcpdump` utility to monitor network traffic.

Run the following command on one of the peers to monitor traffic explicitly on the *interface*. The utility tracks Encapsulated Security Payload (ESP) packets as well as traffic traversing the UDP ports 500 and 4500 that are used by the ipsec service:

```
$ tcpdump -n -i interface esp or udp port 500 or udp port 4500
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on interface, link-type EN10MB (Ethernet), capture size 262144 bytes
10:05:53.578884 IP 10.147.25.195 > 10.147.25.196: ESP(spi=0xcba1dd78,seq=0x2325), length 96
10:05:53.579353 IP 10.147.25.196 > 10.147.25.195: ESP(spi=0x979dcdbe,seq=0x2325), length 124
10:05:56.585128 IP 10.147.25.195 > 10.147.25.196: ESP(spi=0xcba1dd78,seq=0x2326), length 96
10:05:56.585527 IP 10.147.25.196 > 10.147.25.195: ESP(spi=0x979dcdbe,seq=0x2326), length 124
...
```

The utility first reports traffic that is due to the peers exchanging keys regularly.

While the `tcpdump` is running, go to the other peer and perform a network operation, such as a network `ping`, to the first host. The host that is monitoring the traffic should report network activity over the VPN from the second peer.

Press Ctrl+c to end the operations on both peers.