Oracle Linux 8

Setting Up Networking

F21491-25
January 2024

Oracle Linux 8 Setting Up Networking,

F21491-25

Contents

Preface

Documentation License vi

Conventions vi

Documentation Accessibility vi

Access to Oracle Support for Accessibility vi

Diversity and Inclusion vii

1 Configuring the System's Network

Network Configuration Tools 1-1

Configuring Network Interfaces 1-2

- About Network Interface Names 1-2
- Using the Network Connection Editor GUI 1-3
- Using the Text Based User Interface 1-4
- Using the Command Line 1-6

Configuring Network Routing 1-8

- Using the Network Connection Editor 1-8
- Using the Command Line 1-9
- Using the Command Line in Interactive Mode 1-11

Using NetworkManager Connection Profiles 1-12

- Creating a keyfile Connection Profile in Offline Mode Using nmcli 1-12
- Creating a keyfile Connection Profile Manually 1-14
- Renaming Process Differences Between Connection Profile Formats 1-16
- Converting Connection Profile Formats From ifcfg to keyfile 1-16

2 Optimizing Network Servers for High Availability

Working With Network Bonding 2-1

- Configuring Network Bonding 2-2
  - Using the Command Line 2-2
  - Using the Network Connections Editor 2-3
- Verifying the Network Bond Status 2-5

Working With Network Interface Teaming 2-5

Configuring Network Interface Teaming 2-6
Adding Ports to and Removing Ports from a Team 2-7
Changing the Configuration of a Port in a Team 2-7
Removing a Team 2-8
Displaying Information About Teams 2-8
Configuring VLANs With Untagged Data Frames 2-9

3 Configuring DHCP Services

Setting Up the Server's Network Interfaces 3-1
Understanding DHCP Declarations 3-3
Activating the DHCP Services 3-8
Recovering From a Corrupted Lease Database 3-8

4 About Network Address Translation

5 Configuring the Name Service

About DNS and BIND 5-1
Types of Name Servers 5-2
Installing and Configuring a Name Server 5-2
Working With DNS Configuration Files 5-3
Configuring the named Daemon 5-4
About Resource Records in Zone Files 5-7
About Resource Records for Reverse-Name Resolution 5-9
Administering the Name Service 5-10
Performing DNS Lookups 5-10

6 Configuring Network Time

About the chrony Suite 6-1
About the chronyd Service Daemon 6-1
Using the chronyc Service Utility 6-1
Configuring the chronyd Service 6-3
Editing the chronyd Configuration File 6-4
Converting From ntp to chrony 6-5
About PTP 6-5
Configuring the PTP Service 6-7
Using PTP as a Time Source for NTP 6-10

7 Configuring Virtual Private Networks

Preface

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

Documentation License

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/.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at https://www.oracle.com/corporate/accessibility/templates/t2-11535.html.

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.

1

Configuring the System's Network

To enable the system to connect to the network, transmit and receive traffic with other systems, you would need to configure the system to have identifiable names, IP addresses, routes, and so on. Depending on the system's available resources, you can further optimize the network configuration to attain high availability and improved performance by implementing added network technologies such as network bonds, teams, and multipathing.

> **Note:**
>
> In Oracle Linux 8, network configuration is managed by `NetworkManager`. If the system still uses network scripts from previous Oracle Linux releases, migrate them to `NetworkManager`.

Network Configuration Tools

Different tools are available to configure the network. All them typically 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. For more information, see https://docs.oracle.com/en/operating-systems/oracle-linux/cockpit/.
- GNOME based tools
  If you selected the default System With GUI installation profile or environment to install Oracle Linux, these tools are automatically included. For more information on installation profiles, see the Oracle Linux release's installation guide.
  - The GNOME settings application enables you to perform various system configurations, including networking. To access this application, click the network icon at the upper right of the desktop and select **Settings**. Alternatively, click Activities on the desktop's menu bar, select Show Applications, then select Settings. From the list on the left panel, select the type of configuration you want to do.
  - The Network Connection Editor is a subset of the GNOME settings application which you can use to directly perform network configurations. To start the editor, type the `nm-connection-editor` command on a terminal window.
- `NetworkManager` command line tools
  Use these tools if you didn't select the Server With GUI installation profile to install Oracle Linux.
  - To start `NetworkManager`'s text based user interface (TUI), type the `nmtui` command on a terminal window. Navigate through the interface by using keyboard keys instead of the mouse device.
  - `NetworkManager`'s command line consists of the `nmcli` command with different subcommands and options. With a combination of subcommands, options, and arguments, you can complete network configurations in a single command syntax. Other commands, such as `ip` and `ethtool`, complement `nmcli` for configuring and

managing network settings. Optionally, to avoid entering long commands, you can use `nmcli` in interactive mode.
For more information, see the `nmcli(1)`, `ip(8)`, and `ethtool(8)` manual pages.

Configuring Network Interfaces

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

To configure NICs by using Cockpit, see Oracle Linux: Using the Cockpit Web Console .

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 don't 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, including add-in adapters. Server platforms with several network adapters could have 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 implements consistent naming on devices is enabled in Oracle Linux 8 by default. Network interface names are based on information that's derived from the system BIOS. Alternatively, they can be based on 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 `f`*`function`* and `d`*`device-id`* that are added to the suffix.
- The bus and slot number `p`*`bus`*`s`*`n`* , and thus. `enp0s8`.

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

- The MAC address `x`*MAC-addr* , and thus, `enx0217b08b`.

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

Using the Network Connection Editor GUI

1. If not already installed, install the `nm-connection-editor` package.

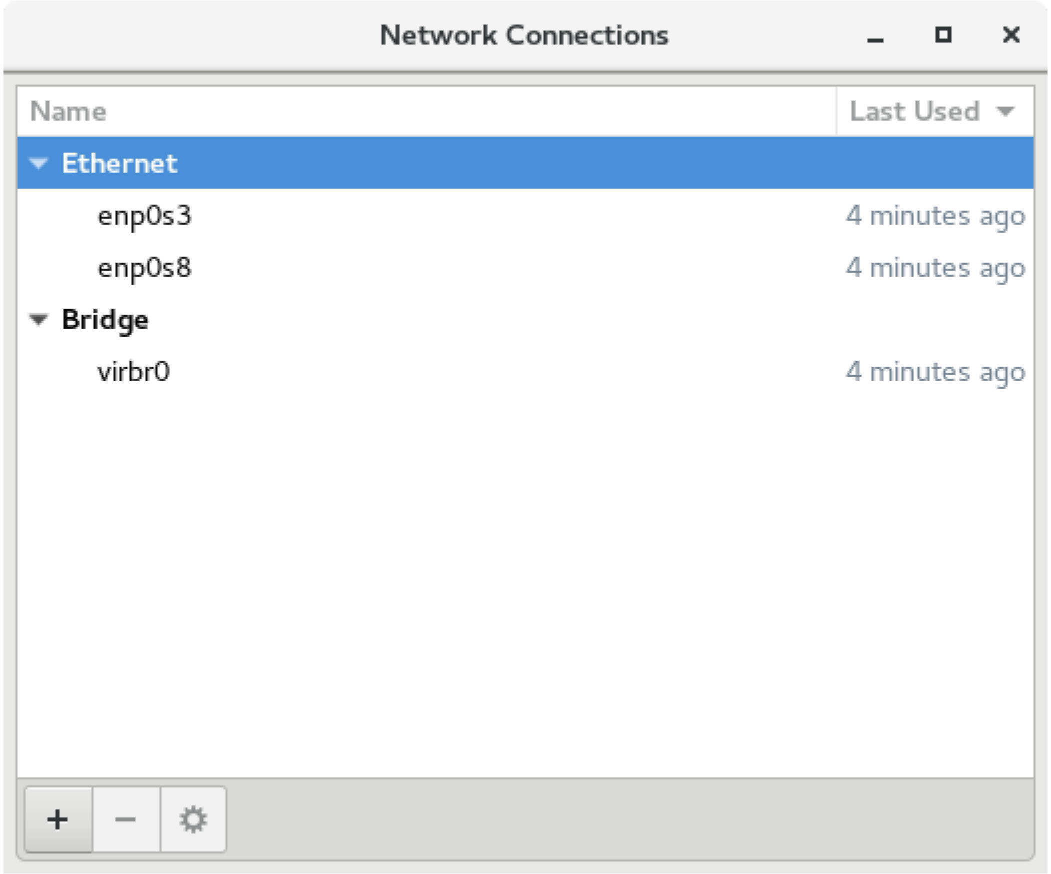
   ```
   sudo dnf install -y nm-connection-editor
   ```

2. Start the editor:

   ```
   sudo nm-connection-editor
   ```

   The editor detects the network devices that are on the system and lists them and their current states:

   Figure 1-1 Network Connections

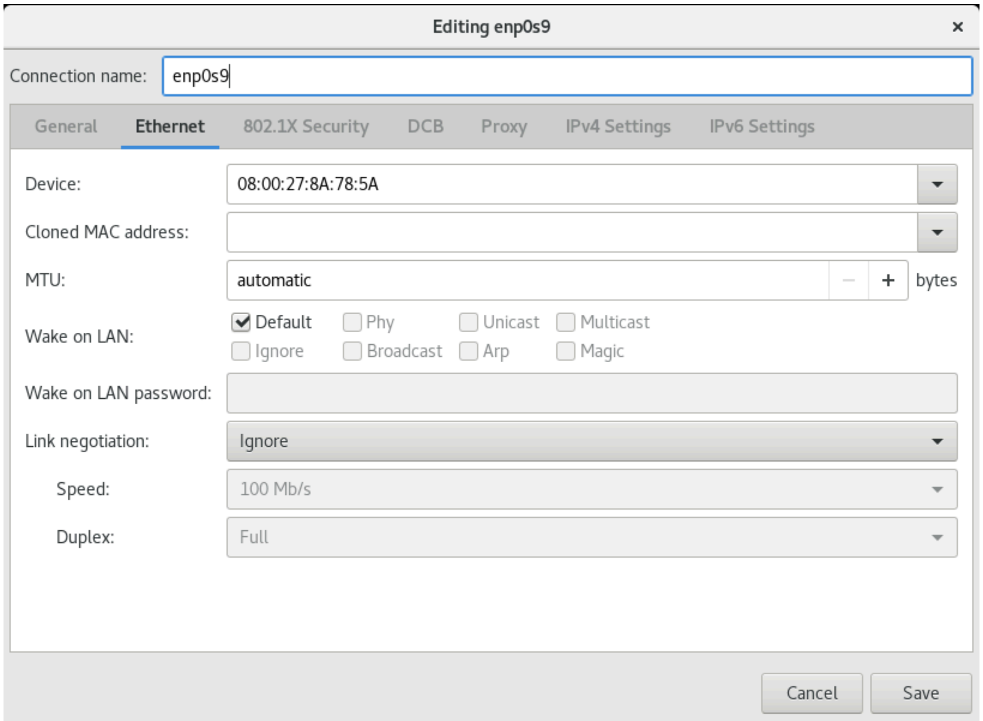
   

3. To add or remove a connection, use the plus (+) or minus (-) buttons at the bottom of the editor window.

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



Connection name: enp0s9

Device: 08:00:27:8A:78:5A

Cloned MAC address:

MTU: automatic − + bytes

Wake on LAN: ☑ Default ☐ Phy ☐ Unicast ☐ Multicast ☐ Ignore ☐ Broadcast ☐ Arp ☐ Magic

Wake on LAN password:

Link negotiation: Ignore

Speed: 100 Mb/s

Duplex: Full

4. Click each tab as needed and enter the required information about the interface.
5. Click **Save** after you have completed the configuration.

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

Using the Text Based User Interface

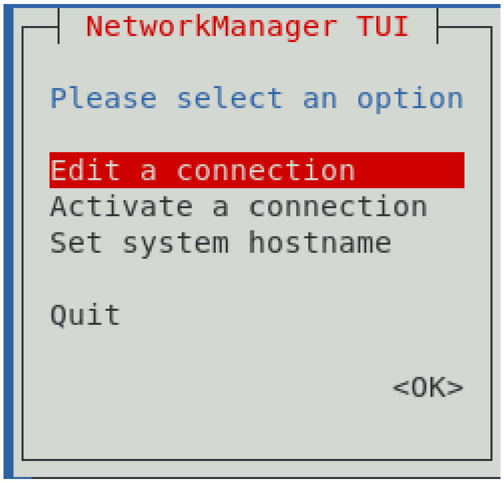
1. If not already installed, install the `NetworkManager-tui` package.

   ```
   sudo dnf install -y NetworkManager-tui
   ```

2. Open `NetworkManager`'s text-based user interface.

   ```
   sudo nmtui
   ```

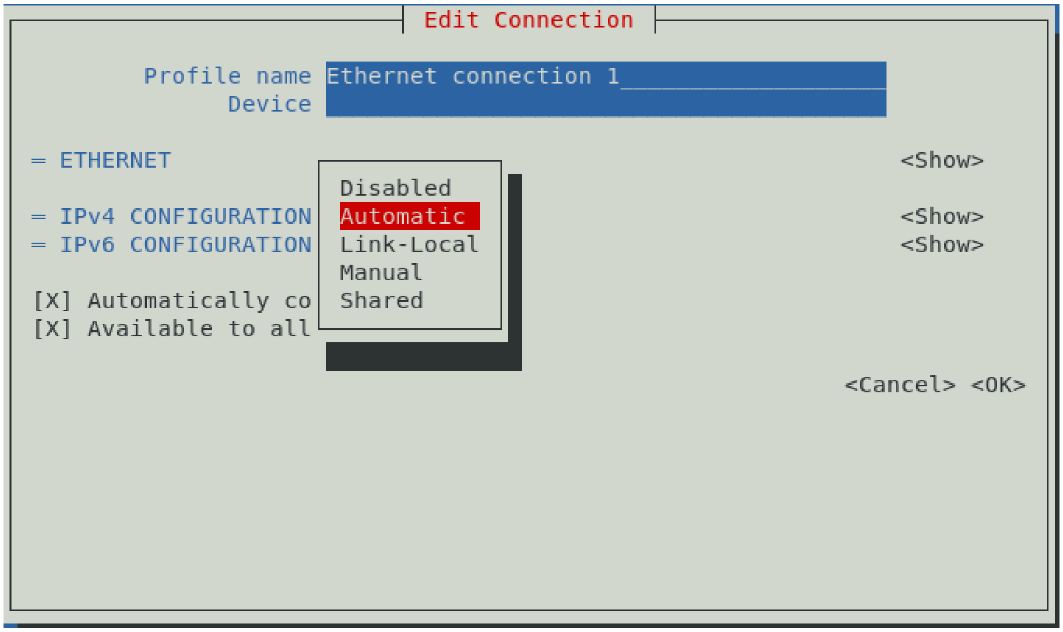
Figure 1-3 TUI Main Menu



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

3. To add a connection, select **Edit a connection**, then click **Add**.
4. After selecting a connection type, the Edit Connection window opens.

Figure 1-4 Edit Connection



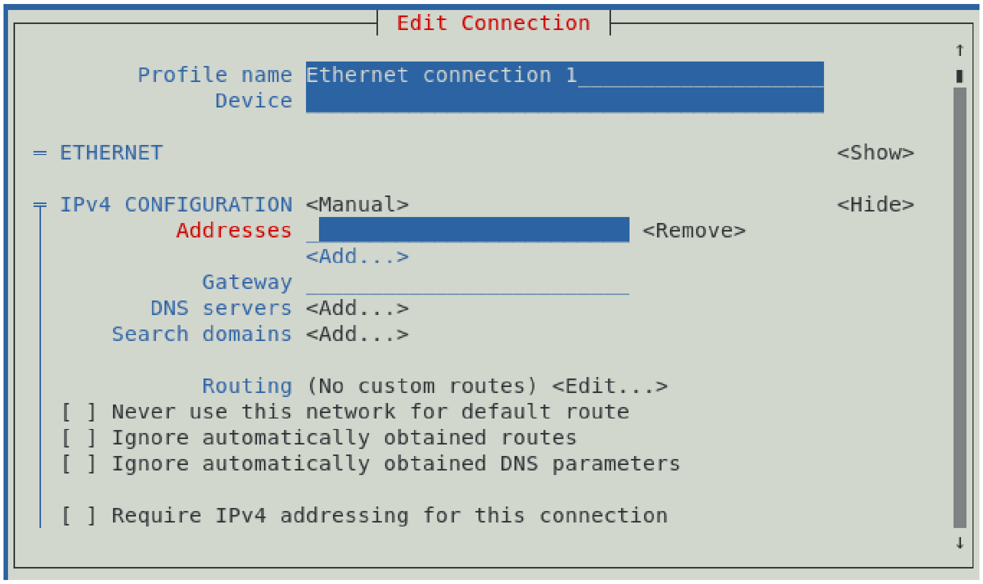
```
Edit Connection

        Profile name Ethernet connection 1
              Device
```

5. As an option, specify a preferred profile name and the name of the device.
6. 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's selected. For example, 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



```
                         Edit Connection

         Profile name Ethernet connection 1________________
               Device

═ ETHERNET                                              <Show>

╤ IPv4 CONFIGURATION <Manual>                           <Hide>
│           Addresses _________________ <Remove>
│                     <Add...>
│             Gateway _________________
│         DNS servers <Add...>
│      Search domains <Add...>
│
│             Routing (No custom routes) <Edit...>
│ [ ] Never use this network for default route
│ [ ] Ignore automatically obtained routes
│ [ ] Ignore automatically obtained DNS parameters
│
│ [ ] Require IPv4 addressing for this connection
```

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

Using 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 need 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 don't 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.

   ```
   sudo 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 it is managed or unmanaged.

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

   ```
   sudo 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.

   ```
   sudo 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:

   ```
   sudo nmcli con add type ethernet ifname enp0s2 con-name "My Work
   Connection" ip4 192.168.5.10/24 gw4 192.168.5.2
   ```

   The output would acknowledge that the connection is successfully completed.

4. Activate the interface.

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

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

   ```
   sudo 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 profile is created. For more information on connection profiles, see Using NetworkManager Connection Profiles.

```
ls -lrt /etc/sysconfig/network-scripts/ifcfg*


...
-rw-r--r--. 1 root root 266 Aug  6 11:03 /etc/sysconfig/network-
scripts/ifcfg-My_Work_Connection
```

Configuring Network Routing

A system uses its routing table to identify which network interface to use when sending packets to remote systems. For a system with only a single interface, configuring the IP address of a gateway system on the local network suffices to route packets to other networks. For example, see the image Figure 1-5, which shows a field where you can enter the IP address of the default gateway.

On systems that have several IP interfaces, you can define static routes so that traffic for a special host or network is forwarded to that network through the default gateway. You use the same tools to configure routing as you do to configure network interfaces.

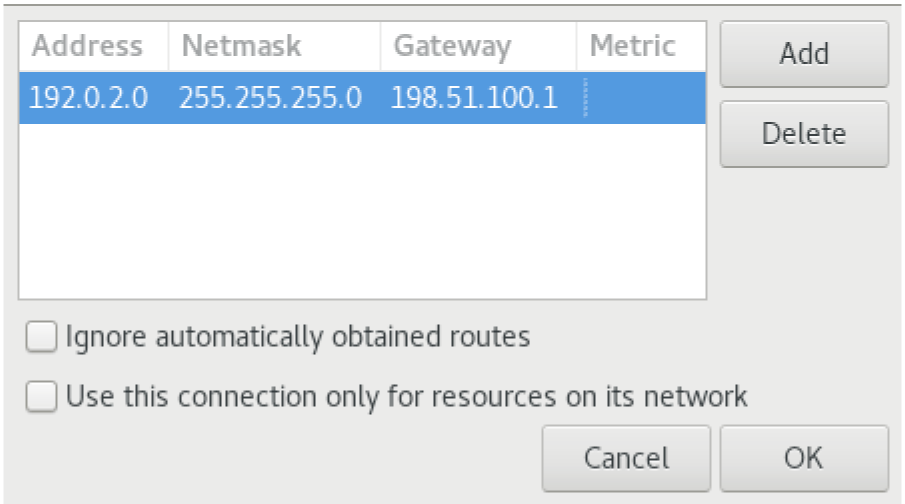
Using the Network Connection Editor

To create a static route to the 192.0.2.0/24 network through the gateway 198.51.100.1, ensure first that the default gateway 198.51.100.1 is reachable on the interface. Then, complete the following steps:.

1. Start the editor.

   ```
   nm-connection-editor
   ```

2. From the list of connections, select the device under the connection name for which you want to create a static route. For example, under `myconnection`, you would select the device `ens3`.
3. Click the settings icon (gear wheel) to edit the connection settings.
4. Click the IPv4 Settings tab.
5. Click **Routes**.
6. Click **Add**.
7. Enter the network's address and netmask for which the route is created, and specify the gateway IP address through which the route is established. You can optionally enter a metric value and select the other available options on display.



Address	Netmask	Gateway	Metric
192.0.2.0	255.255.255.0	198.51.100.1	

□ Ignore automatically obtained routes

□ Use this connection only for resources on its network

8. Clcik **OK** and then save.
9. Back at the terminal window, restart the connection.
   This step causes the connection to temporarily drop.

```
sudo nmcli connection up myconnection
```

10. Optionally, verify that the new route is active.

```
ip route
```

```
...
192.0.2.0/24 via 198.51.100.1 dev myconnection proto static metric 100
```

Using the Command Line

To configure static routes with the `nmcli` command, use the following syntax:

```
nmcli connection modify connection_name +ipv4.routes "ip[/prefix] options(s)
attribute(s)"[next_hop] [metric] [attribute=value] [attribute=value] ..."
```

+ipv4.routes
The plus (+) sign indicates that you're creating an IPv4 route. Without the sign, the command changes an existing IPv4 setting.

connection-name
Connection name or label for which you're creating a static route.

***ip*[*/prefix*]**
IP address of the static route that you're creating. The IP address can also be in CIDR notation.

options
Options include next hop addresses and optional route metrics. These options are separated by spaces. For more information, see the `nm-settings-nmcli(5)` manual pages.

attributes
Attributes are entered as *attribute=value* and are also separated by spaces. Some attributes are `mtu`, `src`, `type`, `cwnd`, and so on. For more information, see the `nm-settings-nmcli(5)` manual pages.

Suppose that you have the following configurations:

- Name of the connection: `myconnection`
- Default gateway address: 198.51.100.1
- Network to which you want to create a statci route: 192.0.2.0/24

To create the route, ensure first that the default gateway for the route is directly reachable on the interface. Then, do the following:

1. Create the static route.

   ```
   sudo nmcli connection modify myconnection +ipv4.routes
   "192.0.2.0/24 198.51.100.1"
   ```

   To create several static routes in a single command, separate the *route gateway* entries with commas, for example:

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

2. Verify the new routing configuration.

   ```
   nmcli connection show myconnection


   --
   ipv4.routes:   { ip = 192.0.2.0/24, nh = 198.51.100.1 }
   --
   ```

3. Restart the network connection.
   This step causes the connection to temporarily drop.

   ```
   sudo nmcli connection up myconnection
   ```

4. Optionally, verify that the new route is active.

```
ip route
```

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

Using the Command Line in Interactive Mode

You can also use the `nmcli` command in interactive mode to configure network settings, including configuring static routes. When in interactive mode, the `nmcli>` prompt appears where you can run commands to configure static routes for a specific connection profile.

The procedure in this section assumes the following network settings for creating the static route:

- Name of the connection: `myconnection`
- Default gateway address: 198.51.100.1
- Network to which you want to create a statci route: 192.0.2.0/24

To create the route, ensure first that the default gateway for the route is directly reachable on the interface. Then, do the following:

1. Start the command's interactive mode.

```
sudo nmcli connection modify myconnection
```

```
nmcli>
```

2. Create the static route.

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

3. Optionally, display the new configuration.

```
nmcli> print
```

```
...
ipv4.routes:        { ip = 192.0.2.1/24, nh = 198.51.100.1 }
...
```

4. Save the configuration.

```
nmcli> save persistent
```

5. Restart the network connection.
This step causes the connection to temporarily drop.

```
nmcli> activate myconnection
```

6. Exit the interactive mode.

   ```
   nmcli> quit
   ```

7. Optionally, verify that the new route is active.

   ```
   ip route
   ```

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

Using NetworkManager Connection Profiles

Each network connection configuration that you create becomes a `NetworkManager` connection profile on the system. In Oracle Linux 8, profiles recognize both the key file format and the `ifcfg` format so that network scripts from previous Oracle Linux releases continue to work. However, network scripts are deprecated in Oracle Linux 8. If you manually create connection profiles, use the key file format instead.

Depending on its purpose, a `NetworkManager` connection profile can be stored in one of the following locations:

- `/etc/NetworkManager/system-connections/`: Default location of persistent profiles that are created by the user. Profiles in this directory can also be edited.
- `/run/NetworkManager/system-connections/`: Location of temporary profiles that are automatically removed when you reboot the system.
- `/usr/lib/NetworkManager/system-connections/`: Location of predeployed and permanent connection profiles. If you edit one of these profiles by using the `NetworkManager` API, then the profile is copied either to the persistent or the temporary directory.

For more information about configuring `NetworkManager` connection profiles, see:

- Creating a `keyfile` Connection Profile in Offline Mode Using `nmcli`
- Creating a `keyfile` Connection Profile Manually
- Renaming Process Differences Between Connection Profile Formats
- Converting Connection Profile Formats From `ifcfg` to `keyfile`

Creating a `keyfile` Connection Profile in Offline Mode Using `nmcli`

When creating or updating `NetworkManager` profile connections, we recommend using its CLI tool in offline mode (`nmcli --offline`). In offline mode, `nmcli` operates without the `NetworkManager` service, which offers user enhanced editing control and the ability to create various connection profiles in `keyfile` format. For example, you can create the following type of connection profiles in `keyfile` format:

- static Ethernet connection
- dynamic Ethernet connection
- network bond

- network bridge
- VLAN or any kind of enabled connections

To create a `keyfile` connection profile using `nmcli` in offline mode, follow these steps:

1. Use the required `NetworkManager` configuration properties to create a profile connection in offline mode.

   For example, the following syntax creates a keyfile connection profile in offline mode for an Ethernet device with a manually assigned IPv4 address and DNS address.

   ```
   nmcli --offline connection add type ethernet con-name Example-Connection
   ipv4.addresses ###.#.#.#/# ipv4.dns ###.#.#.### ipv4.method manual > /etc/
   NetworkManager/system-connections/outputmconnection
   ```

   where:

   - `nmcli --offline` = ncmi mode property, which instructs `nmcli` to operate in offline mode.
   - `connection add type ethernet` = add connection and type properties, which is used to create a connecton profile and specify a connection type value (in this example: Ethernet).
   - `con-name` = connection name property, which is saved into the `id` variable for the generated connection profile.
     When you manage this connection later, using `nmcli`, note the following `id` variable usages:
     - In cases where the `id` variable is provided, use the connection name. For example: `Example-Connection`.
     - In cases where the `id` variable is omitted, use the file name without the `.nmconnection` suffix, for example `output`.

   > **Note:**
   >
   > For more information about connection profile properties and their settings, see the `nm-settings(5)` manual page.

2. Set permissions to the configuration file so that only the `root` user can read and update it. For example:

   ```
   chmod 600 /etc/NetworkManager/system-connections/outputmconnection
   chown root:root /etc/NetworkManager/system-connections/outputmconnection
   ```

3. Start the `NetworkManager` service:

   ```
   systemctl start NetworkManager.service
   ```

4. If you set the `autoconnect` variable in the profile to `false`, activate the connection:

   ```
   nmcli connection up Example-Connection
   ```

5. (Optional) To verify the profile configuration, perform these steps:

a. Verify that the `NetworkManager` service is running, for example:

```
systemctl status NetworkManager
● NetworkManager.service - Network Manager
   Loaded: loaded (/usr/lib/systemd/system/
NetworkManager.service enabled vendor preset: enabled)
   Active: active (running) because Wed -03 13:08:32 CEST   ago
```

b. Verify that `NetworkManager` can read the profile from the configuration file, for example:

```
nmcli -f TYPE,FILENAME,NAME connection
TYPE
FILENAME                                                    NAME
ethernet /etc/NetworkManager/system-connections/
outputmconnection Example-Connection
ethernet  /etc/sysconfig/network-scripts/ifcfg-
enp0                 enp0
```

If the output doesn't display the newly created connection, verify that the `keyfile` permissions and the syntax used are correct.

c. To display the connection profile use the `nmcli connection show` command, for example:

```
nmcli connection show Example-Connection
connection.id:                          Example-Connection
connection.uuid:                        ce8d4422-9603-4d6f-
b602-4f71992c49c2
connection.stable-id:                   --
connection.type:                        802-3-ethernet
connection.interface-name:              --
connection.autoconnect:                 yes
```

Creating a `keyfile` Connection Profile Manually

To manually create a `NetworkManager` connection profile in a keyfile format, follow these steps:

> **Note:**
>
> Manually creating or updating the configuration files can result in an unexpected network configuration. Another option would be to use `nmcli` in offline mode. See

1. If you're creating a profile for a hardware interface, such as Ethernet, display the hardware's MAC address.

   ```
   ip address show ens3
   ```

   ```
   2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc pfifo_fast
   state UP group default qlen 1000
       link/ether 02:00:17:03:b9:ae brd ff:ff:ff:ff:ff:ff
       ...
   ```

2. Use any text editor to create a connection profile that contains the network settings that you want to define for the connection.
   For example, if the connection uses DHCP, the profile would contain settings similar to the following example:

   ```
   [connection]
   id=myconnection
   type=ethernet
   autoconnect=true

   [ipv4]
   method=auto

   [ipv6]
   method=auto

   [ethernet]
   mac-address=02:00:17:03:b9:ae
   ```

3. Save the profile to `/etc/NetworkManager/system-connections/`*`filname`*`.nmconnection`.
   In this current procedure, the profile would be `/etc/NetworkManager/system-connections/myconnection.nmconnection`.

   > **Note:**
   >
   > The defined ID variable, such as `myconnection`, doesn't need to be identical with the profile's file name, for example `myethernet.nmconnection`. When you change the profile by using the `nmcli` command, you can identify the profile by the defined ID (`myconnection`) or by the file name, but excluding the file extension name (`myethernet`).

4. Restrict the permissions of the profile.

   ```
   sudo chown root:root /etc/NetworkManager/system-connections/
   myconnection.nmconnection
   sudo chown 600 /etc/NetworkManager/system-connections/
   myconnection.nmconnection
   ```

5. Reload the connection profiles.

   ```
   sudo nmcli connection reload
   ```

6. Verify that `NetworkManager` can read the profile.

   ```
   sudo nmcli -f NAME,UUID,FILENAME connection
   ```

   ```
   NAME           UUID       FILENAME
   myconnection   uuid        /etc/NetworkManager/system-connections/
   myconnection.nmconnection
   ```

7. If you specified `false` for the profile's `autoconnect` parameter, then activate the connection.

   ```
   sudo nmcli connection up myconnection
   ```

Renaming Process Differences Between Connection Profile Formats

In cases when you need to assign a custom name to an interface, the `udev` service renaming process works differently depending on the format of the connnection profile. For example,

- `ifcfg` format interface renaming process involves these steps:
  1. The `/usr/lib/udev/rules.d/60-net.rulesudev` rule calls the `/lib/udev/rename_device` helper utility.
  2. The helper utility searches for the `HWADDR` parameter in `/etc/sysconfig/network-scripts/ifcfg-*` files.
  3. If the value set in the variable matches the MAC address of an interface, the helper utility renames the interface to the name set in the `DEVICE` parameter of the file.
- `keyfile` format interface renaming process involves these steps:
  1. To rename an interface, create a systemd link file or a udev rule.
  2. In the `NetworkManager` connection profile, specify the custom name in the `interface-name` property.

Converting Connection Profile Formats From `ifcfg` to `keyfile`

To convert the `NetworkManager` legacy `ifcfg` profile formats to the preferred `NetworkManager keyfile` format, follow these steps:

> **Note:**
>
> For more information about the `keyfile` profile format, see the nm-settings-keyfile(5) manual page.

1. Ensure that the following prerequisites are met:
   - Existing connection profiles in `ifcfg` format that are stored in the `/etc/sysconfig/network-scripts/` directory.

- If the connection profiles contain a DEVICE variable that is set to a custom device name, such as provider or lan, you created a systemd link file or a udev rule for each of the custom device names.

2. Using the nmcli, migrate the ifcfg connection profiles to the preferred keyfile formats.

```
nmcli connection migrate
```

3. (Optional) Verify that all the legacy ifcfg connection profiles migrated successfully:

```
nmcli -f TYPE,FILENAME,NAME connection
```

2 Optimizing Network Servers for High Availability

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. With virtual local area networks (VLANs), you can also organize the network such that systems with similar functions are grouped together as though they belong to their own virtual networks. This feature improves network management and administration.

For a system to avail of these advanced features, it must have several NICs. The more NICs, the better assurances of network availability that a server can provide.

Working With Network Bonding

A system's physical network interfaces that are connected to a network switch can be grouped together into a single logical interface to provide better 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 activating failover from one component device to another. By default, a bonded interface appears similar to a normal network device to the kernel, but it sends out network packets over the available secondary devices by using a 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 the requirements and the 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 basic 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`) don't 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 change the hardware address while the device is active.

For more information on the kernel bonding driver, 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 failover. Otherwise, the mechanism might not work.

Configuring Network Bonding

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

Using the Command Line

For a tutorial, which includes a hands-on lab environment, on configuring network bonds, see Create Network Bonds using Network Manager CLI .

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

   ```
   sudo 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, the mode is set to `active-backup`. If you don't set the bond connection name, then the bond interface name is also used as the connection name.

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, manually configure the address. For example, to configure IPv4 settings for the bond, type:

   ```
   sudo nmcli connection modify "Bond Connection 1" ipv4.addresses
   '192.0.2.2/24'
   ```

   ```
   sudo nmcli connection modify "Bond Connection 1" ipv4.gateway
   '192.0.2.1'
   ```

   ```
   sudo nmcli connection modify "Bond Connection 1" ipv4.dns
   '192.0.2.254'
   ```

   ```
   sudo nmcli connection modify "Bond Connection 1" ipv4.method manual
   ```

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

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

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

   Give each secondary a connection name, and select the interface name for each interface that you want 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 secondary network interfaces.

4. Start the bond interface.

   ```
   sudo 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.

   ```
   sudo nmcli device
   ```

   ```
   ...
   enp1s0   ethernet  connected  bond0-if1
   enp2s0   ethernet  connected  bond0-if2
   ```

Using the Network Connections Editor

1. Start the editor:

   ```
   sudo nm-connection-editor
   ```

   The Network Connections window opens.

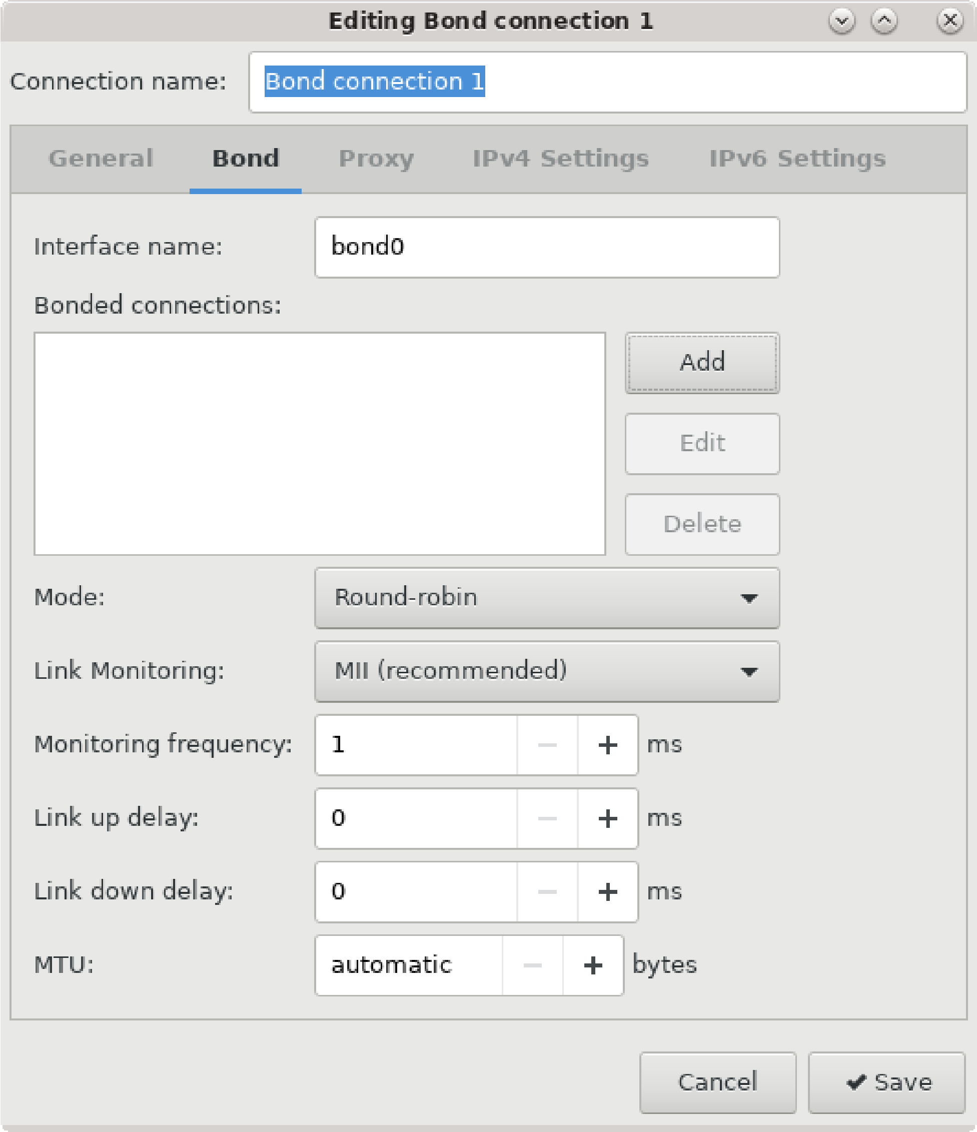
2. To add a connection, use the plus (+) button at the bottom of the window.

   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.

   Network Bond Editor



Editing Bond connection 1

Connection name: Bond connection 1

Interface name: bond0

Bonded connections:

Mode: Round-robin

Link Monitoring: MII (recommended)

Monitoring frequency: 1 ms

Link up delay: 0 ms

Link down delay: 0 ms

MTU: automatic bytes

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 where you can 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 secondary interface.
    b. Optionally configure a name for the secondary interface.
    c. In the **Device** field, select the physical network interface to add as a secondary to the bond. Note that if a device is already configured for networking it's not listed as available to configure within the bond.
    d. Click **Save** to add the secondary device to the network bond.

    Repeat these steps for all 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 might require more configuration on the network switch.

7. Configure other bond parameters such as link monitoring as required if you don't want to use the default settings.

   If you don't intend to use DHCP for network bond IP configuration, set the 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.

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 secondaries are active. The output also provides information about the status of each secondary interface.

2. Temporarily disconnect the physical cable that's connected to one of the secondary interfaces. No other reliable method is available 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 secondary interface would indicate that the interface is down and a link failure has occurred.

Working With Network Interface Teaming

Network interface teaming is similar to network interface bonding and provides a way of implementing link aggregation that's relatively maintenance-free, easier to reconfigure, 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's 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's 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
Sends packets over the available ports in a round-robin fashion.

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

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

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.

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.

   ```
   sudo ip link set eth1 down
   ```

   ```
   sudo ip link set eth2 down
   ```

> **Note:**
>
> Active interfaces can't 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.

   ```
   sudo 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.

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

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

Adding Ports to and Removing Ports from a Team

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

```
sudo teamdctl team0 port add eth3
```

To remove a port from a team:

```
sudo teamdctl team0 port remove eth3
```

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

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:

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

Enclose the JSON-format definition in single quotes and don't split it over several lines.

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

Removing a Team

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

```
sudo teamd -t team0 -k
```

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

Displaying Information About Teams

Display the network state of the teamed interface as follows:

```
sudo 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:

```
sudo 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:

```
sudo 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:

```
sudo 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.

Configuring VLANs With Untagged Data Frames

A VLAN is a group of machines that can communicate as though they're attached to the same physical network. With a VLAN, you can 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 the ports that you assign with this PVID are in a single broadcast domain. Broadcasts between devices in the same VLAN aren't 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:

```
sudo 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 don't 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:

```
sudo 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.

You can also use the `ip` command to create VLAN devices. However, such devices don't persist across system reboots.

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

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

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

3

Configuring DHCP Services

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 can't 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. If 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
- Reconfiguration of the IP address space on the DHCP server without needing to reconfigure each client

For more information about DHCP, see RFC 2131. Likewise, see the following manual pages:

- `dhcpd(8)`
- `dhcp-options(5)`

Setting Up the Server's Network Interfaces

By default, the `dhcpd` service processes requests on those network interfaces that connect them to subnets that are defined in the DHCP configuration file.

Suppose that a DHCP server has mutliple interfaces. Through its interface `enp0s1`, the server is connected to the same subnet as the clients that the server is configured to serve. In this case, `enp0s1` must be set in the DHCP service to enable the server to monitor and process incoming requests on that interface.

Before proceeding to either of the following procedures, ensure that you meet the following requirements:

- You have the proper administrative privileges to configure DHCP.
- You have installed the `dhcp-server` package.

If not, install the package with the following command:

```
sudo dnf install dhcp-server
```

Configure the network interfaces as follows:

- For IPv4 networks:
  1. Copy the `/usr/lib/systemd/system/dhcpd.service` file to the `/etc/systemd/system/` directory.

     ```
     sudo cp /usr/lib/systemd/system/dhcpd.service /etc/systemd/
     system/
     ```

  2. Edit the `/etc/systemd/system/dhcpd.service` by locating the line that defines the `ExecStart` parameter.
  3. Append the interface names on which the `dhcpd` service should listen.

     See the sample entries in bold.

     ```
     ExecStart=/usr/sbin/dhcpd -f -cf /etc/dhcp/dhcpd.conf -user
     dhcpd -group dhcpd --no-pid $DHCPDARGS int1-name int2-name
     ```

  4. Reload the `systemd` manager configuration.

     ```
     sudo systemctl daemon-reload
     ```

  5. Restart the `dhcpd` service configuration.

     ```
     sudo systemctl restart dhcpd.service
     ```

     Alternatively, you can also type:

     ```
     sudo systemctl restart dhcpd
     ```

- For IPv6 networks:
  1. Copy the `/usr/lib/systemd/system/dhcpd6.service` file to the `/etc/systemd/system/` directory.

     ```
     sudo cp /usr/lib/systemd/system/dhcpd6.service /etc/systemd/
     system/
     ```

  2. Edit the `/etc/systemd/system/dhcpd6.service` file by locating the line that defines the `ExecStart` parameter.
  3. Append the names of the interfaces on which the `dhcpd6` service should listen.

     See the sample entries in bold.

     ```
     ExecStart=/usr/sbin/dhcpd -f -cf /etc/dhcp/dhcpd6.conf -user
     dhcpd -group dhcpd --no-pid $DHCPDARGS int1-name int2-name
     ```

4. Reload the `systemd` manager configuration.

   ```
   sudo systemctl daemon-reload
   ```

5. Restart the `dhcpd` service configuration.

   ```
   sudo systemctl restart dhcpd6.service
   ```

   Alternatively, you can also type:

   ```
   sudo systemctl restart dhcpd6
   ```

Understanding DHCP Declarations

The way the DHCP provides services to its clients is defined through parameters and declarations in the `/etc/dhcp/dhcpd.conf` file for IPv4 networks and `/etc/dhcp/dhcpd6.conf` file for IPv6 networks. The file would contain details such as client networks, address leases, IP address pools, and so on.

> **Note:**
>
> In a newly installed Oracle Linux system, both the `dhcpd.conf` and `dhcpd6.conf` files are empty. If the server is being configured for DHCP for the first time, then you can use the templates so you can be guided in configuring the files. Type one of the following commands:
>
> - For IPv4
>
>   ```
>   cp /usr/share/doc/dhcp-server/dhcpd.conf.example /etc/dhcp/
>   dhcpd.conf
>   ```
>
> - For IPv6
>
>   ```
>   cp /usr/share/doc/dhcp-server/dhcpd6.conf.example /etc/dhcp/
>   dhcpd6.conf
>   ```
>
> Then when you open either file, examples, and explanations are available for reference.

The information in the configuration file consists of a combination of the following declarations:

- Global Settings
- Subnet Declarations
- Shared-network Declarations
- Host Declarations
- Group Declarations

Global Settings

Global parameters define settings that apply to all networks that are supported or serviced by the DHCP server.

Consider the following settings that would globally apply through out the entire network:

- Domain name of the company network: `example.com`.d
- Network's DNS servers: `dn1.example.com` and `dns2.example.com`
- Lease time assigned to all clients: 12 hours (43200 seconds)
- Maximum lease time that can be assigned: 24 hours (86400 seconds)

In this case, you would configure the global settings in the configuration file as follows:

```
option domain-name "example.com";
default-lease-time 43200;
max-lease-time 86400;

authoritative;
```

The `authoritative` parameter identifies the server as an official or primary server for DHCP services. The parameter is typically used in a setup that has multiple DHCP servers. Servers with the `authoritative` parameter have priority to process requests over servers without the parameter.

Subnet Declarations

A `subnet` declaration provides details about a subnet to which the DHCP server is directly connected and where the systems in that subnet are also being served as clients.

Consider the following configuration of a DHCP server:

- The server's `enp0s1` interface is directly connected to the 192.0.2.0/24 network.
- The systems in the 192.0.2.0/24 network are DHCP clients.
- The topology of this client subnet is as follows:
  - Subnet's DNS server: 192.0.2.1.
  - Subnet gateway: 192.0.2.1.
  - Broadcast address: 192.0.2.255.
  - Address range for clients: 192.0.2.10 through 192.0.2.100.
  - Maximum lease time for each client: 86,400 seconds (1 day).

In this case, you would enter the following declaration in `dhcp.conf`:

```
subnet 192.0.2.0 netmask 255.255.255.0 {
  range 192.0.2.10 192.0.2.100;
  option domain-name-servers 192.0.2.1;
  option routers 192.0.2.1;
  option broadcast-address 192.0.2.255;
  max-lease-time 86400;
}
```

On an IPv6 network environment, a `subnet` declaration in the `dhcpd6.conf` file would resemble the following example:

```
subnet6 2001:db8:0:1::/64 {
  range6 2001:db8:0:1::20 2001:db8:0:1::100;
  option dhcp6.name-servers 2001:db8:0:1::1;
  max-lease-time 172800;
}
```

Shared-network Declarations

You define a `shared-network` declaration if the DHCP server needs to provide servers to clients in other subnets that aren't directly connected to the server.

Consider the following example, which expands but slightly differs from the scenario in the preceding section:

- The DHCP server belongs to the 192.0.2.0/24 network but doesn't provide services to the systems in this network.
- The server processes requests from clients in the following remote subnets:
  - 192.168.5.0/24.
  - 198.51.100.0/24.
- The remote subnets share the same DNS server, but each subnet has its own router and IP address range.

In this case, you would enter the following declarations in `dhcp.conf`:

```
shared-network example {

  option domain-name-servers 192.168.2.1;
  ...

  subnet 192.168.5.0 netmask 255.255.255.0 {
    range 192.168.5.10 192.168.5.100;
    option routers 192.168.5.1;
  }

  subnet 198.51.100.0 netmask 255.255.255.0 {
    range 198.51.100.10 198.51.100.100;
    option routers 198.51.100.1;
  }
  ...
}

subnet 192.0.2.0 netmask 255.255.255.0 {
}
```

In the preceding example, the final `subnet` declaration refers to the server's own network and is outside the `shared-network` scope. The declaration is called an empty declaration because it defines the server's subnet. Because the server doesn't provide services to this subnet, no added entries are added, such as lease, address range, DNS information, and so on. Though empty, the declaration is required, otherwise, the `dhcpd` service doesn't start.

On an IPv6 network environment, a `shared-network` declaration in the `dhcpd6.conf` file would resemble the following example:

```
shared-network example {
  option domain-name-servers 2001:db8:0:1::1:1
  ...

  subnet6 2001:db8:0:1::1:0/120 {
    range6 2001:db8:0:1::1:20 2001:db8:0:1::1:100
  }

  subnet6 2001:db8:0:1::2:0/120 {
    range6 2001:db8:0:1::2:20 2001:db8:0:1::2:100
  }
  ...
}

subnet6 2001:db8:0:1::50:0/120 {
}
```

Host Declarations

You define a `host` declaration if a client needs to have a static IP address.

Consider the following example of a client printer in the server's 192.0.2.0/24 network. This time, the server provides DHCP services to the subnet.

- Printer's MAC address: 52:54:00:72:2f:6e.
- Printer's IP address: 192.0.2.130

> **Important:**
>
> A client's fixed IP address must be outside the pool of dynamic IP addresses distributed to other clients. Otherwise, address conflicts might occur.

In this case, you would enter the following declaration in `dhcp.conf`:

```
host printer.example.com {
    hardware ethernet 52:54:00:72:2f:6e;
    fixed-address 192.0.2.130;
}
```

Systems are identified by the hardware ethernet address, and not the name in the `host` declaration. Thus, the host name might change, but the client continues to receive services through the ethernet address.

On an IPv6 network environment, a `host` declaration in the `dhcpd6.conf` file would resemble the following example:

```
host server.example.com {
    hardware ethernet 52:54:00:72:2f:6e;
    fixed-address6 2001:db8:0:1::200;
}
```

Group Declarations

You define a `group` declaration to apply the same parameters to multiple shared networks, subnets, and hosts all at the same time.

Consider this example:

- The DHCP server belongs to and serves the subnet 192.0.2.0/24.
- One client requires a fixed address, while the rest of the clients use dynamic IP addresses from the server.
- All the clients use the same DNS server.

In this case, you would enter the following declaration in `dhcp.conf`:

```
group {
  option domain-name-servers 192.0.2.1;

  host server1.example.com {
    hardware ethernet 52:54:00:72:2f:6e;
    fixed-address 192.0.2.130;
  }

  subnet 192.0.2.0 netmask 255.255.255.0 {
  range 192.0.2.10 192.0.2.100;
  option routers 192.0.2.1;
  option broadcast-address 192.0.2.255;
  max-lease-time 86400;
  }
}
```

On an IPv6 network environment, a `group` declaration in the `dhcpd6.conf` file would resemble the following example:

```
group {
  option dhcp6.domain-search "example.com";

  host server1.example.com {
    hardware ethernet 52:54:00:72:2f:6e;
    fixed-address 2001:db8:0:1::200;
  }

  host server2.example.com {
    hardware ethernet 52:54:00:1b:f3:cf;
    fixed-address 2001:db8:0:1::ba3;
  }
}

subnet6 2001:db8:0:1::/64 {
  range6 2001:db8:0:1::20 2001:db8:0:1::100;
  option dhcp6.name-servers 2001:db8:0:1::1;
  max-lease-time 172800;
}
```

Activating the DHCP Services

All the DHCP services are defined in the server's `/etc/dhcp/dhcpd.conf` or `/etc/dhcp/dhcpd6.conf` file. To configure and then activate the configured services, follow these steps:

- For IPv4 networks:
  1. Open the `/etc/dhcp/dhcpd.conf` file.
  2. Add parameters and declarations to the file.
     For guidance, see Understanding DHCP Declarations or to the comments and notes in the `/usr/share/doc/dhcp-server/dhcpd.conf.example` template.
  3. Optionally, set the `dhcpd` service to start automatically in a server reboot.

     ```
     sudo systemctl enable dhcpd
     ```

  4. Start or restart the `dhcpd` service.

     ```
     sudo systemctl start dhcpd
     ```

- For IPv6 networks:
  1. Open the `/etc/dhcp/dhcpd6.conf` file.
  2. Add parameters and declarations to the file.
     For guidance, see Understanding DHCP Declarations or to the comments and notes in the `/usr/share/doc/dhcp-server/dhcpd6.conf.example` template.
  3. Optionally, set the `dhcpd6` service to start automatically in case of a server reboot.

     ```
     sudo systemctl enable dhcpd6
     ```

  4. Start or restart the `dhcpd` service.

     ```
     sudo systemctl start dhcpd6
     ```

Recovering From a Corrupted Lease Database

The `dhcpd` service maintains lease information, such as IP addresses, MAC addresses, and lease expiry times, in the following flat-file databases:

- For DHCPv4: `/var/lib/dhcpd/dhcpd.leases`.

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

To prevent the lease database files from becoming too large with stale data, the `dhcpd` service periodically regenerates the files through the following mechanism:

1. The service renames the existing lease files:
   - `/var/lib/dhcpd/dhcpd.leases` is renamed to `/var/lib/dhcpd/dhcpd.leases~`
   - `/var/lib/dhcpd/dhcpd6.leases` is renamed to `/var/lib/dhcpd/dhcpd6.leases~`
2. The service re-creates brand new `dhcpd.leases` and `dhcpd6.leases` files.

If a lease database file is corrupted, you need to restore the lease database from the last known backup of the database.

Typically, the most recent backup of a lease database is the *filename*.leases~ file.

> **Note:**
>
> A backup instance is a snapshot taken at a particular point in time, and therefore might not reflect the latest state of the system.

Ensure that you have the required administrative privileges and complete the following steps:

- For DHCPv4
  1. Stop the `dhcpd` service:

     ```
     sudo systemctl stop dhcpd
     ```

  2. Rename the corrupt lease database:

     ```
     sudo mv /var/lib/dhcpd/dhcpd.leases /var/lib/dhcpd/
     dhcpd.leases.corrupt
     ```

  3. Restore the lease database from its corresponding *filename*.leases~ backup file.

     ```
     sudo cp -p /var/lib/dhcpd/dhcpd.leases~ /var/lib/dhcpd/dhcpd.leases
     ```

  4. Start the `dhcpd` service:

     ```
     sudo systemctl start dhcpd
     ```

- For DHCPv6
  1. Stop the `dhcpd` service:

     ```
     sudo systemctl stop dhcpd6
     ```

2. Rename the corrupt lease database:

   ```
   sudo mv /var/lib/dhcpd/dhcpd6.leases /var/lib/dhcpd/
   dhcpd6.leases.corrupt
   ```

3. Restore the lease database from its corresponding `filename.leases~` backup file.

   ```
   sudo cp -p /var/lib/dhcpd/dhcpd6.leases~ /var/lib/dhcpd/
   dhcpd6.leases
   ```

4. Start the `dhcpd6` service:

   ```
   sudo systemctl start dhcpd6
   ```

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 the requests as though they're 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 the system can route packets among configured network interfaces. To check whether IP forwarding is enabled, use the following command:

```
sudo sysctl -a | grep ip_forward
```

In the ensuing output, a value of `1` for `net.ipv4.ip_forward` indicates that IP forwarding is enabled.

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

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

The new status is displayed when you run the command. To make the change persist across system reboots, copy the command output line and add it to the `/etc/sysctl.conf` file.

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.

5

Configuring the Name Service

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

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 fully qualified domain name (FQDN) `wiki.us.mydom.com`. In this example, the top-level domain is `com`, `mydom` is a subdomain of `com`, `us` is a subdomain 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's 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 subdomain.

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 isn't authoritative, the server queries a `root` name server for the address of a name server that's 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's authoritative for the FQDN. Most resolvers use recursive queries and so can't 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 some configuration files, including the following:

`/etc/named.conf`
Contains settings for `named` and lists the location and characteristics of the zone files for the domain. Zone files are typically 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.

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.

Secondary or backup 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. This server isn't 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.

Installing and Configuring a Name Server

By default, you can use the BIND installation 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 either use the default settings or configure new `named` configuration and zone files.

To configure a name server:

1. Install the bind package.

   ```
   sudo 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, change the `named` configuration and zone files.

   See Configuring the named Daemon more details.
5. Configure the system firewall to accept incoming TCP connections to port 53 and incoming UDP datagrams on port 53:

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

   ```
   sudo 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:

   ```
   sudo systemctl restart NetworkManager
   ```

   ```
   sudo systemctl start named
   ```

   ```
   sudo systemctl enable named
   ```

Working With DNS Configuration Files

Domains are grouped into zones that are configured through 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.

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's 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 might 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 would 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 activates 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 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 the network environment. The following example defines settings for the service and 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 enables external files to be referenced so that 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 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`.

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.

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 the network doesn't have a prefix length that's a multiple of 8, see RFC 2317 for the format that you need to use instead.

The PTR records in in-addr.arpa or ip6.arpa domains define host names that correspond to the host part 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.

Administering the Name Service

The rndc command enables you to administer the named service. The service is administered locally. If the service is configured in the controls section of the /etc/named.conf file, then you can also use the command line to manage named remotely. 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:

```
sudo rndc-confgen -a
```

The command creates the /etc/rndc.key file.

Check the status of the named service as follows:

```
sudo 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 change the named configuration file or zone files, the rndc reload command instructs named to reload the files:

```
sudo rndc reload
```

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

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:

```
sudo host 192.168.2.101
```

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

```
sudo host dns.us.mydoc.com
```

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

```
sudo 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:

```
sudo 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.

6 Configuring Network Time

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.

About the chrony Suite

`chrony` is a feature that implements NTP to maintain timekeeping accurately 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

For practical exercises in using `chrony`, see Configure Chrony on Oracle Linux .

About the chronyd Service Daemon

The `chronyd` service daemon updates the system clock of mobile systems and virtual machines after a period of suspension or disconnection from a network. The service can also be used to implement a basic NTP client or NTP server. As an NTP server, `chronyd` can synchronize with upper level 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.

Using the chronyc Service Utility

The `chronyc` utility is a tool for managing 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:

  ```
  sudo chronyc subcommand
  ```

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

  ```
  sudo chronyc
  ```

  ```
  chronyc>
  ```

  From the prompt, you can issue the different `chronyc` subcommands as needed. The following examples show the information that's generated by the `sources` and `sourcestats` subcommands:

  ```
  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
```

To quit using the interactive mode, type `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 Editing the chronyd Configuration File.

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

Configuring the chronyd Service

To configure the `chronyd` service on a system:

1. Install the `chrony` package.

   ```
   sudo 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:

   ```
   sudo firewall-cmd --zone=zone --add-service=ntp
   ```

   ```
   sudo firewall-cmd --zone=zone --permanent --add-service=ntp
   ```

3. Start the `chronyd` service and configure it to start following a system reboot.

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

   ```
   sudo systemctl start chronyd
   ```

   ```
   sudo systemctl enable chronyd
   ```

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 configures a system 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` no longer works 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` doesn't 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.

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's similar to when it's used with ntp.

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

```
sudo 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 -boption specifies to create backup configuration files before converting, while the -v option specifies to display verbose messages during the migration process.

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

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

About PTP

Use PTP to synchronise system clocks on a LAN more accurately than than NTP. If network drivers support either hardware or software time stamping, a PTP clock can use the time stamps in PTP messages to resolve propagation delays across a network. With software time stamping, PTP synchronises 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 several 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 added subnetworks. A boundary clock is usually implemented as a function of a network switch.

- Several *secondary clock* systems.

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

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

Grandmaster and secondary 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 secondary to a grandmaster clock or a higher-level boundary clock; the other interfaces act as masters to secondary clocks or lower-level boundary clocks.

Synchronization of boundary and secondary clock systems is achieved by sending time stamps in PTP messages. By default, PTP messages are sent in UDPv4 datagrams. You can also 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:

```
sudo 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.

Configuring the PTP Service

To configure the PTP service on a system:

1. Install the `linuxptp` package.

   ```
   sudo dnf install linuxptp
   ```

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

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

   For example, to use hardware time stamping with interface `en1` on a secondary 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 secondary (`slaveOnly` mode). Don't 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 customize `ptp4l` further, for example:

   - For a grandmaster clock, set the value of the `priority1` parameter to a value between 0 and 127, where lesser values have greater 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, 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 accept access by PTP event and general messages to UDP ports 319 and 320 in the appropriate zone, for example:

   ```
   sudo firewall-cmd --zone=zone --add-port=319/udp --add-port=320/udp
   ```

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

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

   ```
   sudo systemctl start ptp4l
   ```

   ```
   sudo 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 secondary clock, synchronise the system clock with the PTP hardware clock that's associated with the secondary network interface, for example:

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

      > **Note:**
      >
      > The secondary 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 synchronising 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.

      ```
      sudo systemctl start phc2sys
      ```

      ```
      sudo 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's directly connected to the grandmaster clock system without any intermediate boundary clocks:

```
sudo 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


sudo 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 examples include 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.

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:**
>
> Don't configure any added `server` lines in the file.

7 Configuring Virtual Private Networks

Information about virtual private networks (VPNs) is now contained in a separate documentation. Go to Oracle® Linux: Configuring Virtual Private Networks.