Configuring and Administering Network Components in Oracle® Solaris 11.2

Part No: E37475-02
September 2014

ORACLE

Contents

Using This Documentation

- **Overview** – Provides information about how to configure and administer various network components in the Oracle Solaris operating system (OS), such as datalinks, IP interfaces and addresses, naming and directory services, reactive profiles, and wireless networks.
- **Audience** – System administrators who are responsible for managing network configuration in corporate datacenters.
- **Required knowledge** – Basic and advanced network administration concepts and practices.

Product Documentation Library

Late-breaking information and known issues for this product are included in the documentation library at http://www.oracle.com/pls/topic/lookup?ctx=E36784.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Feedback

Provide feedback about this documentation at http://www.oracle.com/goto/docfeedback.

♦ ♦ ♦ CHAPTER 1

About Network Administration in Oracle Solaris

This chapter provides an overview of the various components that comprise network configuration of a host client system in Oracle Solaris. The tasks and examples that are described in this book assume that you are performing network configuration after an installation. For instructions on configuring the network during an installation, see "Installing Oracle Solaris 11.2 Systems ".

For information about new networking features, see "What's New in Oracle Solaris 11.2 ".

For planning tasks that are required prior to configuring a client system on the network, see "Planning for Network Deployment in Oracle Solaris 11.2 ".

For a shortcut to commonly used network administration commands, see Chapter 3, "Network Administration Command Cheatsheet," in "Strategies for Network Administration in Oracle Solaris 11.2 ".

For information about administering existing TCP/IP networks, see Chapter 1, "Administering TCP/IP Networks," in "Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 ".

This chapter contains the following topics:

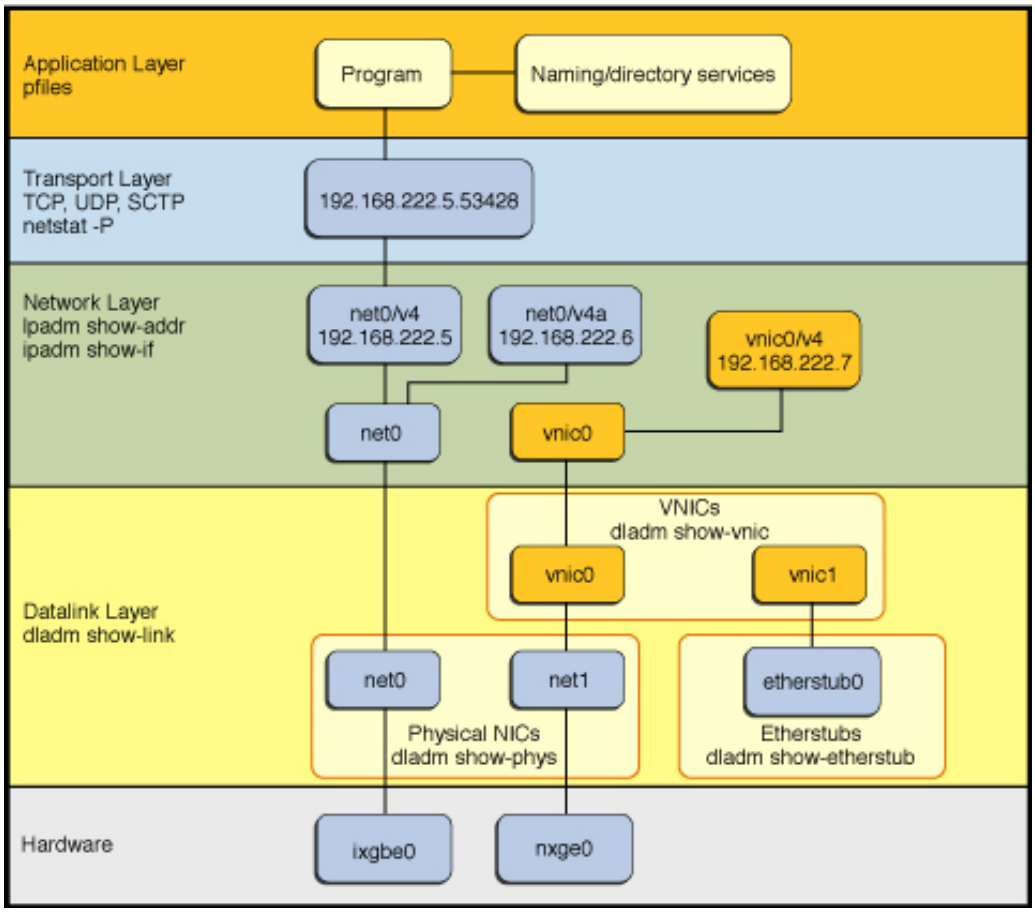
Description of the Oracle Solaris Network Protocol Stack

Network interfaces provide a connection between a system and the network. These interfaces are configured over datalinks, which in turn correspond to instances of hardware devices on the system.

In Oracle Solaris 10, a one-to-one relationship that binds the device, the datalink, and the interface exists, which means network configuration is dependent on the hardware configuration and also the network topology. If changes are implemented in the hardware layer, for example, the replacement of a network interface card (NIC), you must reconfigure the interfaces on the system.

However, in this Oracle Solaris release, the naming of physical datalinks is no longer tied to the underlying hardware associated with the network device. By default, such devices are assigned the generic name `net` and a suffix that reflects the device's physical location in the system, as shown in Figure 1–1. With this separation, the network configuration on the network layer is no longer bound to the chipset or to the network topology in the hardware layer.

FIGURE 1-1 Oracle Solaris 11 Network Protocol Stack



This implementation makes network administration more flexible in the following ways:

- Network configuration is insulated from any changes that might occur in the hardware layer. Link and interface configurations are preserved even if the underlying hardware is removed. These same configurations can then be reapplied to any replacement NIC, provided that the two NICs are of the same type.
- The separation of the network configuration from the network hardware configuration enables the use of customized link names at the datalink layer.
- With the abstraction of the datalink layer, multiple networking abstractions or configurations, such as virtual local area networks (VLANs), virtual network interface cards (VNICs), physical devices, link aggregations, and IP tunnels are unified into a common administrative entity, which is the datalink.

To compare the Oracle Solaris 10 network stack with the Oracle Solaris 11 network stack, see "Comparing the Oracle Solaris 10 Network Protocol Stack to the Oracle Solaris 11 Network Protocol Stack" in "Transitioning From Oracle Solaris 10 to Oracle Solaris 11.2 ".

Hardware Layer

Network hardware devices are also called *network interface cards (NICs)* or *network adapters*. NICs can be built in and already present on the system when the system is purchased. Or, you can purchase separate NICs to add to the system. Certain NICs have only a single interface that resides on the card. Other NIC brands might have multiple interfaces that you can configure to perform various network operations.

Datalink Layer

You perform network configuration at the datalink layer by using the `dladm` command.

The following are some of the types of datalink configuration that can be performed at this layer:

- Basic network configuration of physical links
- Configuration of VNICs (virtual links over physical links)

  For VNIC configuration, each virtual link has it's own MAC address
- Link aggregation configuration

  Aggregations are configured over physical links for reliability and performance.
- Etherstubs for supporting virtual local area networks (VLANs)
- Bridges for supporting VLANs

For other examples, see "Managing Network Datalinks in Oracle Solaris 11.2 ".

Network Layer

You perform network configuration at the network layer by using the `ipadm` command.

The following are some of the types of IP configuration that you can perform at this layer:

- IP interface configuration with names that have a one-to-one correspondence with datalink names
- IP network multipathing (IPMP)
- Virtual Network Interfaces (VNIs)
- Multiple IP addresses for one IP interface
- IPv4 and IPv6 addresses configured on a single IP interface

- IP addresses that are managed by using their address object names

  An *address object name* consists of the interface name followed by a unique string and represents an IP address configured on the system.

For other examples, see the following references:

- "Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 "
- "Managing Network Virtualization and Network Resources in Oracle Solaris 11.2 "
- "Creating and Using Oracle Solaris Zones "

Transport Layer

Explicit configuration at the transport layer is rarely necessary. Applications that work with Oracle Solaris typically select the appropriate transport protocol and the corresponding port numbers automatically. You can view active ports by using the `netstat` command. See `netstat`(1M).

You can tune some transport protocol parameters by using the `ipadm` command. See "Administering Transport Layer Services" in "Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 " and "Oracle Solaris 11.2 Tunable Parameters Reference Manual " for more information.

Application Layer

Application programs access the network through the socket `xti` or `tli` Application Programming Interfaces (APIs). These APIs require that client applications provide the corresponding server's IP address and transport port number when initiating a connection. Normally, the server is known only by a remote host or service name rather than an IP address. Applications use standard library services to translate host and service names to IP addresses before attempting to make network connections.

For more information, see "Oracle Solaris 11.2 Programming Interfaces Guide " for details.

Naming and Directory Services Configuration Within the Oracle Solaris Network Protocol Stack

Client systems must have a network's name and a directory service configured to perform IP address lookups. Depending on your network environment, the configuration process could occur automatically during a system boot or it might need to be performed manually. This configuration takes place at the application layer of the network stack.

For client-side naming and directory service configuration options, see Chapter 4, “Administering Naming and Directory Services on an Oracle Solaris Client”.

Network Devices and Datalink Naming in Oracle Solaris

A datalink represents a link object in the second layer (L2) of the Open Systems Interconnection (OSI) model. The *physical link* is directly associated with a device and possesses a device name. The device name contains the driver name and the device instance number. The instance number can have a value from zero to *n*, depending on how many NICs use that driver on the system.

The device instance name continues to depend on the system's underlying hardware. However, the hardware and software layers are separated, which means that the datalinks configured on top of these devices are no longer similarly bound. Thus, datalinks can be assigned names other than the device names on which they are configured.

By default, datalinks are assigned generic names that use the net# naming convention, where # is the device instance number. The instance number increments for each device on the system, for example, `net0`, `net1`, `net2`, and so on. For a more detailed overview, see “About Datalink Configuration” on page 23.

About Network Configuration Modes

Two network configuration modes are supported in Oracle Solaris: fixed and reactive.

Note - The terms fixed and reactive modes refer to the ability of the system to automatically adjust to changes in the current network environment and not to whether you can configure static or fixed IP addresses when using in these modes.

Fixed Mode

Fixed mode means the instantiated configuration on the system is persistent, regardless of whether any changes in network conditions occur. When such changes occur, such as the addition of interfaces, you have to reconfigure the network for the system to adapt to the new environment. When using a fixed mode, your system is configured by using the same set of network configuration commands every time. Corporate servers most often use this configuration mode due to a relatively stable network environment. When using the fixed mode, you use the `dladm` and `ipadm` commands to manage the various aspects of network configuration. See “Oracle Solaris Network Administration Commands” on page 17.

Reactive Mode

Reactive mode, on the other hand, is when the network is configured automatically in response to current network conditions. This mode is primarily used for laptop computers and notebook personal computers (PCs) and in situations where network conditions might change.

In reactive mode, a network daemon (`nwamd`) monitors the state of the system's network interfaces. Whenever network conditions change, the network daemon adjusts the network configuration dynamically. For example, a notebook PC might be physically attached to the corporate network, or it might not be physically attached. When physically attached, you most likely would disable the notebook's wireless interface. Also, it is most often desirable to have the wireless interface automatically enabled when the Ethernet cable is detached from the notebook. In addition, you might want the system to automatically adjust IP Filter settings when switching to a wireless network. The network daemon can automatically perform these types of dynamic configuration changes for you if you are in the reactive mode. Conversely, these types of changes require manual reconfiguration steps if you are in the fixed mode.

About Profile-Based Network Configuration

Profile-based network configuration enables you to define multiple alternative configurations, each identified by a single profile (referred to as a network configuration profile (NCP)). For example, you could create a profile named `office` for a notebook PC that configures the system with static IP addresses and DNS server locations. An alternate `home` profile might use DHCP to acquire this information. A single command enables you to switch from one profile to another profile in a matter of seconds. The various types of profiles that you can enable support two possible network configuration modes: fixed and reactive. The default mode is determined by whichever profile is currently active on your system.

For information about how profiles are activated on a system during an Oracle Solaris installation, see "How the Network Is Configured During an Installation" in "Transitioning From Oracle Solaris 10 to Oracle Solaris 11.2 ".

If you are unsure of which profile is currently active on your system, use the `netadm list` command to display this information. See "Enabling and Disabling Profiles" on page 109 for more information.

For a complete description of the various types of profiles that are supported in Oracle Solaris, see Chapter 5, "About Administering Profile-Based Network Configuration in Oracle Solaris".

Oracle Solaris Network Administration Commands

The following commands are used to administer network configuration:

- `dladm`
- `ipadm`
- `route`
- `netcfg`
- `netadm`

`dladm` Command

Introduced in Oracle Solaris 10, the `dladm` is used to configure datalinks.

The `dladm` command is used to manage the following types of network configuration:

- **Physical interfaces** – Ethernet, wireless, and InfiniBand
- **Virtual networking features** – Etherstubs, VNICs, and IP tunnels
- **Switch features** – Link aggregations, VLANs, and bridging technologies
- **Device characteristics** – Speed, duplexing, priority, and feature negotiation

The `dladm` command creates persistent network configuration for the profile that is currently active on the system. Thus, `net0` can have different MTU values in different profiles. For example, if a datalink named `net0` is configured with a specific maximum transmission unit (MTU) of 1200, that MTU value is persistent for `net0` just for that profile. If you then activate another profile and set a different MTU value for that profile by using the `dladm` command, the new MTU value would be applied just to that profile. See Chapter 2, “Administering Datalink Configuration in Oracle Solaris”.

The `dladm` command also replaces the `ndd` command that is used to configure protocol properties in Oracle Solaris 10. As a tool for setting Layer 2 driver properties, the `dladm` command provides several advantages over the `ndd` command. See “Comparing the ndd Command to the ipadm Command” in “Transitioning From Oracle Solaris 10 to Oracle Solaris 11.2 ”.

`ipadm` Command

The `ipadm` command replaces the `ifconfig` command for configuring IP interfaces and addresses in this release. The `ipadm` command manages IP interfaces and IP addresses more efficiently because the command is solely used for IP interface administration. Also, unlike the `ifconfig` command, the `ipadm` command implements persistent network configuration. See “Comparing the ifconfig Command to the ipadm Command” in “Transitioning From Oracle Solaris 10 to Oracle Solaris 11.2 ”.

The `ipadm` command also replaces the `ndd` command that was used to configure protocol properties in Oracle Solaris 10. As a tool for setting protocol properties, the `ipadm` command

provides several advantages over the `ndd` command. See “Comparing the ndd Command to the ipadm Command” in “Transitioning From Oracle Solaris 10 to Oracle Solaris 11.2 ”.

`route` Command

Because the `/etc/defaultrouter` file is deprecated in Oracle Solaris 11, you can no longer manage routes (default or otherwise) by using this file. Instead, use the `route` command to manually manipulate the network routing tables. The `route` command manipulates routes for the active profile *only*. The default route, as well as all other routes, could potentially be replaced if the active profile changes. This issue is of no concern if you do not switch profiles on your system.

For more information, see “Creating Persistent (Static) Routes” on page 62.

`netcfg` and `netadm` Commands

The `netcfg` and `netadm` commands are used to manage various types of profiles. Most of the features that are provided by these two commands are targeted at managing reactive profiles. The `netcfg` command is rarely used on corporate servers. These types of servers typically use the fixed mode.

The `netadm` command is used to enable and disable profiles and display information about profiles and their states. See “Enabling and Disabling Profiles” on page 109 and “Administering Profiles” on page 124.

Note - You typically configure the properties of reactive profiles by using the `netcfg` command. However, you can also create persistent configuration for a reactive profile by using the `dladm` and `ipadm` commands, if the profile is currently active. However, you cannot use the `netcfg` command to configure the system's only fixed profile, `DefaultFixed`. For more information, see the `netcfg`(1M) man page.

Commands for Reconfiguring a System's Network

Another option that is available for reconfiguring your system's network is the `sysconfig` utility, also called the System Configuration Interactive (SCI) Tool. The SCI tool supports the configuration of freshly installed or unconfigured systems and is designed to provide system configuration for newly created non-global zones during text installations. You can use the SCI tool interactively or non-interactively.

You can perform three operations by using the `sysconfig` utility: unconfiguration, configuration, and profile creation. The `unconfigure` subcommand is used to unconfigure an entire system. This command leaves the system in an unconfigured state.

The `configure` subcommand is used to reconfigure an entire system or a portion of system, which includes the following six functional groupings:

- `network`
- `location`
- `users`
- `identity`
- `support`
- `kdb_layout`

For more information about the default values for unconfigured groupings, see “Functional Groupings Overview” in “Installing Oracle Solaris 11.2 Systems ”.

For example, you would reconfigure a system's existing naming services as follows:

```
# sysconfig configure -g network,naming_services
```

The `-g` option is used to specify a specific functional grouping that should be configured. In this example, the network component of the system is configured.

For more information, see the `sysconfig`(1M) man page and Chapter 6, “Unconfiguring or Reconfiguring an Oracle Solaris Instance,” in “Installing Oracle Solaris 11.2 Systems ”.

Information That Is Required to Configure Client Systems on the Network

You can configure the network while you are installing Oracle Solaris or after you have installed Oracle Solaris. This book describes tasks for configuring client systems on the network after an installation. For instructions on configuring the network during an installation, see “Installing Oracle Solaris 11.2 Systems ”.

To configure client systems on the network, you must provide the following information:

- Host name
- IP address
- Netmask

  See “Deciding on an IP Addressing Format for Your Network” in “Planning for Network Deployment in Oracle Solaris 11.2 ” and “Obtaining Your Network’s IP Number” in “Planning for Network Deployment in Oracle Solaris 11.2 ” for more information.

If your network is divided into subnets, you must have the subnet numbers and the IP address schema to apply to the systems in each subnet, including their respective netmask. See “Using Subnets on Your Network” in “Planning for Network Deployment in Oracle Solaris 11.2 ”.

- Domain name to which the system belongs

  See “Using Naming Entities on Your Network” in “Planning for Network Deployment in Oracle Solaris 11.2 ”.

- Default router address

  You supply this information if you have a simple network topology with only one router attached to each network. See “Planning for Routers on Your Network” in “Planning for Network Deployment in Oracle Solaris 11.2 ”.

  You also supply this information if your routers do not run routing protocols such as the Router Discovery Server Protocol (RDISC) or the Router Information Protocol (RIP). For more information about routers as well as the list of routing protocols that are supported by Oracle Solaris, see “Routing Protocols” in “Configuring an Oracle Solaris 11.2 System as a Router or a Load Balancer ”.

When configuring client systems on the network, refer to the information in “IPv4 Autonomous System Topology” in “Planning for Network Deployment in Oracle Solaris 11.2 ”.

For more detailed information about each of these components and related tasks, see Chapter 1, “Planning For Network Deployment,” in “Planning for Network Deployment in Oracle Solaris 11.2 ”.

Where to Find More Information About Network Administration in Oracle Solaris

Beyond the basic configuration information that is described in this book, see the following references for information about performing other types of network administration:

- To customize network protocols and administer transport layer services, see Chapter 1, “Administering TCP/IP Networks,” in “Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 ”.
- To configure a system as a router or a load balancer, see “Configuring an Oracle Solaris 11.2 System as a Router or a Load Balancer ”.
- To perform advanced datalink and IP configuration, including link aggregations and various types of bridges, see “Managing Network Datalinks in Oracle Solaris 11.2 ”.
- To configure IPMP groups and IP tunnels, see “Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 ”.
- To establish security for your network, see “Securing the Network in Oracle Solaris 11.2 ”.
- To implement network virtualization features, see “Managing Network Virtualization and Network Resources in Oracle Solaris 11.2 ”.

♦ ♦ ♦ CHAPTER 2

Administering Datalink Configuration in Oracle Solaris

This chapter describes network devices and datalinks and includes tasks for managing basic datalink configuration when using the fixed mode.

The following information pertains exclusively configuring physical links or links that represent network devices. For more information about other Layer 2 (L2) entities that you can configure, see "Managing Network Datalinks in Oracle Solaris 11.2 " and "Managing Network Virtualization and Network Resources in Oracle Solaris 11.2 ".

This chapter contains the following topics:

About Datalink Configuration

Administrators create IP interfaces on top of datalinks. Each datalink represents a link object in the second layer of the Open Systems Interconnection (OSI) model. Datalinks can represent many different L2 entities such as physical network devices (termed *physical links*), aggregations of physical datalinks, virtual network interface cards (VNICs), etc.

Link names are either assigned when the associated link object is automatically created, or you can explicitly assign link names when you create the datalinks. Physical links (those that are associated with physical network devices) are created automatically when devices are added or when an Oracle Solaris system first boots after an installation. In this Oracle Solaris release, the naming of physical datalinks is no longer tied to the underlying hardware that is associated with the network device. By default, datalinks are assigned names that are prefixed by `net` and suffixed by a number that reflects the physical location of the datalink in the system. For example, the first onboard network device `e1000g0` would be assigned the name `net0`, while the next `e1000g1` device would be assigned the name `net1`, and so on. You can assign arbitrary names to datalinks that you explicitly create, for example, link aggregations. Also, you can explicitly rename the default-assigned `net`*N* name of a datalink, if desired.

Generic or flexible link names provide the following advantages for network configuration:

- Within a single system, dynamic reconfiguration (DR) becomes easier. The network configuration for a given NIC can be inherited by a different NIC replacement.
- The network setup for zones migration becomes less complicated. The zone in the migrated system preserves its network configuration if the destination system's link shares the same name with the link that is assigned to the zone prior to migration. Thus, no additional network configuration for the zone is required after the migration.
- The generic naming convention makes network configuration that is specified in the System Configuration (SC) manifest used during an installation less complicated. Because the primary network datalink is generically named `net0` for all systems, you can use a generic SC manifest for multiple systems that specify a configuration for `net0`.
- Datalink administration also becomes flexible. You can further customize the names of datalinks, for example, to reflect a specific function that the datalink serves.

  The following table illustrates the new correspondence between the hardware (NIC), the device instance, the link name, and the interface over the link. The names of the datalinks are automatically provided by the OS.

Hardware (NIC)	Device Instance	Link's Assigned Name	IP Interface
`e1000g`	`e1000g0`	`net0`	`net0`
`igb`	`ixgbe`	`net1`	`net1`

  As indicated in this table, while the device instance name remains hardware-based, the datalinks are renamed by the OS after the installation.

To display the mapping between datalinks, their generic names, and the corresponding device instances, use the `dladm show-phys` command as follows:

```
# dladm show-phys
LINK     MEDIA        STATE     SPEED     DUPLEX     DEVICE
net2     Ethernet     up        1000      full       bge2
net0     Ethernet     up        1000      full       e1000g0
net3     Ethernet     up        1000      full       nge3
net1     Ethernet     up        1000      full       e1000g1
```

Assigning Generic Names to Datalinks

- Physical network devices are ordered according to media type, where certain types have priority over others. The media types are ordered in descending priority as follows:
  1. Ethernet
  2. InfiniBand devices

3. Ethernet over IB
4. WiFi

- After devices are grouped and sorted according to media types, these devices are further ordered based on their physical locations, where on-board devices are favored over peripheral devices.
- Devices that have higher a priority based on their media type and location are assigned lower instance numbers.

On SPARC based systems, the net*N* names are assigned to match the net*N* device aliases that are used in the OpenBoot PROM (OBP). On x86 based systems, SMBIOS data (where available) is used to identify onboard Ethernet devices and assign them to `net0`, `net1`, and so on. In addition to (or absent from) these sources of information, devices on a lower motherboard or IO board, host bridge, PCIe root complex, bus, device, and function are ranked ahead of other devices and assigned lower `net` instances than those on higher motherboards, host bridges, etc.

To display the correspondences between link names, devices, and locations, use the `dladm show-phys` command with the `-L` option as follows:

```
# dladm show-phys -L
LINK         DEVICE       LOCATION
net0         e1000g0      MB
net1         e1000g1      MB
net2         e1000g2      MB
net3         e1000g3      MB
net4         ibp0         MB/RISER0/PCIE0/PORT1
net5         ibp1         MB/RISER0/PCIE0/PORT2
net6         eoib2        MB/RISER0/PCIE0/PORT1/cloud-nm2gw-2/1A-ETH-2
net7         eoib4        MB/RISER0/PCIE0/PORT2/cloud-nm2gw-2/1A-ETH-2
```

Customizing How Generic Link Names Are Assigned By the Operating System

Caution - You must customize how generic link names are automatically assigned *before* you install Oracle Solaris. After installation, you cannot customize the default link names without removing existing configurations.

Oracle Solaris uses the prefix `net` when assigning link names. However, you can use any custom prefix that you prefer, such as `eth`. You can also disable the automatic assignment of generic link names.

To disable automatic link naming, or to customize the prefix of link names, set the following property in the System Configuration (SC) manifests. The SC manifests are used by the Automated Installer (AI) feature of Oracle Solaris.

```
<service name="network/datalink-management"
version="1" type="service">
<instance name="default enabled="true">
<property_group name='linkname-policy'
type='application'>
<propval name='phys-prefix' type='astring'
value='net'/>
</property_group>
</instance>
</service
```

By default, the value for the `phys-prefix` property is set to `net`, as shown in bold in the previous output.

- To disable automatic naming, set the value for the `phys-prefix` property to an empty string, for example:

  ```
  <propval name='phys-prefix' type='astring' value=''/>
  ```

  If you disable automatic naming, then datalink names are based on their associated hardware drivers, such as `bge0`, `e1000g0`, and so on.
- To use a prefix other than `net`, specify a new prefix as the value of `phys-prefix`, such as `eth`.

If the value that is provided for the `phys-prefix` property is invalid, then that value is ignored. The datalinks are then named according to their associated hardware drivers, such as `bge0`, `e1000g0`, and so on. For rules about valid link names, see "Rules for Valid Link Names" on page 28.

Link Names in Upgraded Systems

On freshly installed systems, datalinks are automatically named `net0` through `net`*N-1*, where *N* represents the total number of network devices.

On the contrary, if you upgrade from another Oracle Solaris 11 release, the datalinks retain their names that were established prior to the upgrade. These names are either the default hardware-based names or customized names that the administrator assigned to the datalinks before the upgrade. Further, on these upgraded systems, new network devices that are subsequently added also retain the default hardware-based names rather than receive generic names. This behavior for upgraded systems ensures that no generic names assigned by the OS become mixed with other hardware-based names or customized names assigned by the administrator before the upgrade.

You can replace both hardware-based names as well as OS-supplied link names with other names that you prefer to use. Typically, the default link names that OS assigns suffice for creating the system's network configuration. However, consider the following information before making changes to link names.

Replacing Hardware-Based Link Names

If your system's links have hardware-based names, rename these links with at least generic names. If you retain the hardware-based names, confusion might arise later when these physical devices are removed or replaced.

For example, you retain the link name `bge0` that is associated with the device `bge0`. All link configurations are performed by referring to the link name. Later, you might replace the NIC `bge` with the NIC `e1000g`. To reapply the former device's link configuration to the new NIC `e1000g0`, you would need to reassign the link name `bge0` to `e1000g0`. The combination of a hardware-based link name `bge0` with a different associated NIC `e1000g0` can cause confusion. By using names that are not hardware-based, you can better distinguish the links from the associated devices.

Caution About Changing Link Names

Although replacing hardware-based link names is a best practice, you must still plan carefully before you rename links. Changing a device's link name does not automatically propagate the new name to all existing associated configurations. The following examples illustrate the risks when you change link names:

- Some rules in an IP Filter configuration apply to specific links. When you change a link's name, the filter rules continue to refer to the link's original name. Consequently, these rules no longer behave as expected after you rename the link. You need to adjust the filter rules to apply to the link by using the new link name.
- Consider the possibility of exporting network configuration information. As previously explained, by using the default `net#` names provided by the OS, you can migrate zones and easily export network configuration to another system. If the target system's network devices are named with generic names such as `net0`, `net1`, and so on, then the zone simply inherits the network configuration of the datalink whose name matches the datalink assigned to the zone.

Thus, as a general rule, do not rename datalinks randomly. When renaming datalinks, ensure that all of the link's associated configurations continue to apply after the link name is changed.

Some of the configurations that might be affected by renaming links are as follows:

- IP Filter rules
- IP configurations that are specified by using the `ipadm` command
- Oracle Solaris 11 zones
- `autopush` configuration

Note - No changes are required in the `autopush` configuration when you rename links. However, you must be aware of how the configuration works with the per-link `autopush` property after the link has been renamed. For more information, see Setting the STREAMS Module on Datalinks.

Rules for Valid Link Names

When you assign link names, observe the following rules:

- Link names must consist of a string and a *physical point of attachment* (PPA) number.
- The link name must abide by the following constraints:
  - Names ideally consist of between 3 to 8 characters. However, names can have a maximum of 16 characters.
  - Valid characters for names are alphanumeric (a–z, 0–9) and the underscore (_).

Caution - Do not use upper case letters on link names.

- Each datalink must have only one link name at one time.
- Each datalink must have a unique link name within the system.

Note - As an added restriction, you cannot use `lo0` as a flexible link name. This name is reserved to identify the IP loopback interface.

The function of the link within your network setup can be a useful reference when you assign link names. For example, `netmgt0` can be a link that is dedicated to network management. `Upstream2` can be the link that connects to the ISP. As a general rule to avoid confusion, do *not* assign names of known devices to your links.

Administering Datalink Properties

Using the `dladm` command to customize common datalink properties provides the following benefits:

- The `dladm` command is the only command interface that is required for configuring network driver properties. This command replaces the former practice of using a

combination of the `ndd` command and `driver.conf` file modifications to set driver properties.

- The following uniform syntax is used, regardless of which properties are set:

  `dladm` *subcommand properties datalink*

- Use of the `dladm` command applies to both public and private properties of the driver.
- Using the `dladm` command on a specific driver does not disrupt network connections of other NICs of similar types. Thus, you can configure datalink properties dynamically.
- Datalink configuration values are stored in a `dladm` repository and persist over system reboots.

Displaying General Information About Datalinks

When used without any options, the `dladm` command displays general information about the system's datalinks, including the class, state, and the underlying physical links.

```
# dladm
LINK        CLASS     MTU     STATE     OVER
net0        phys      1500    unknown   --
net1        phys      1500    up        --
net2        phys      1500    unknown   --
net3        phys      1500    unknown   --
net4        phys      1500    up        --
aggr0       aggr      1500    up        net1,net4
```

Datalinks can be of different classes, other than physical links, for example, link aggregations, virtual LANs (VLANs), and virtual NICs (VNICs). These other datalinks are also included in the default information that is displayed by the `dladm` command. For example, in the previous output a link aggregation (`aggr0`) is configured over the physical datalinks `net1` and `net4`.

For information about link aggregations and VLANs, see "Managing Network Datalinks in Oracle Solaris 11.2 ". For information about VNICs, see "Managing Network Virtualization and Network Resources in Oracle Solaris 11.2 ".

Displaying a System's Datalinks

Use the `dladm show-link` command to display both the physical and virtual datalinks on a system. A system has as many datalinks as there are installed NICs. You can use various options with this command to customize the information that is displayed.

When used with no additional options or arguments, the `dladm show-link` command displays the following information:

```
# dladm show-link
LINK                CLASS     MTU    STATE    OVER
net1                phys      1500   down     --
net3                phys      1500   unknown  --
net0                phys      1500   up       --
net2                phys      1500   unknown  --
net11               phys      1500   up       --
net5                phys      1500   up       --
net6                phys      1500   up       --
```

In the previous output, the `STATE` column shows the current state of the *virtual datalink*. The state can be `up`, `down`, or `unknown`. For virtual datalinks, when a NIC is split up into multiple VNICs, a virtual switch is implicitly created internally. This creation of a virtual switch enables the VNICs and the primary datalink to communicate with each other, as long as they are on the same VLAN, even if the physical datalink has no connection to the external network. This relationship forms the *virtual state* of the datalink.

Use the `-P` option to display persistent configuration information about the datalinks. Based on the information that is provided by this command, you can proceed with further network configuration. For example, you can determine the number of NICs on the system, and you can select which datalink to use, over which you can configure IP interfaces. When you type the command, the information that is displayed is similar to the following example:

```
# dladm show-link -P
LINK        CLASS     OVER
net0        phys      --
net1        phys      --
net2        phys      --
```

The previous example shows that the system has three datalinks that are directly associated with their corresponding physical NICs. No special datalinks exist, such as aggregations or virtual NICs, which are configured over the datalinks under the `phys` class.

Displaying the Physical Attributes of Datalinks

Use the `dladm show-phys` command to obtain information about the system's datalinks in relation to the physical NICs with which they are associated. Used without any options, the command displays information that is similar to the following example:

```
# dladm show-phys
LINK        MEDIA          STATE     SPEED     DUPLEX     DEVICE
net0        Ethernet       up        100Mb     full       e1000g0
net1        Ethernet       down      0Mb       --         nge0
net2        Ethernet       up        100Mb     full       bge0
net3        InfiniBand     --        0Mb       --         ibd0
```

The previous output shows, among other details, the physical NICs with which the datalinks that have generic link names are associated. For example, `net0` is the datalink name of the NIC

`e1000g0`. To display information about flags that have been set for the datalinks, use the `-P` option. For example, a datalink that is flagged with `r` means that its underlying NIC has been removed.

In the previous output, the `STATE` column shows the current state of the *physical datalink*. The state can be `up`, `down`, or `unknown`. The physical link state identifies whether the physical device has connectivity with the external network (which it does, if the cable is plugged in and the state of the port on the other end of the cable is `up`.

The `-L` option is another useful option that you can use. This option displays the physical location for each datalink. The location determines the instance number of the datalink, such as `net0`, `net1`, and so on.

```
# dladm show-phys -L
LINK     DEVICE     LOCATION
net0     bge0       MB
net2     ibp0       MB/RISER0/PCIE0/PORT1
net3     ibp1       MB/RISER0/PCIE0/PORT2
net4     eoib2      MB/RISER0/PCIE0/PORT1/cloud-nm2gw-2/1A-ETH-2
```

Use the `-m` option to display the MAC addresses of the physical links in a system:

```
# dladm show-phys -m
LINK                SLOT     ADDRESS            INUSE CLIENT
net0                primary  0:11:22:a9:ee:66   yes   net0
```

This command is similar to using the `ifconfig` command.

Display the MAC addresses of all of the links in a system (physical and non-physical) as follows:

```
# dladm show-linkprop -p mac-address
LINK     PROPERTY        PERM VALUE        EFFECTIVE    DEFAULT   POSSIBLE
net0     mac-address     rw   0:11:22:a9:ee:66 0:11:22:a9:ee:66 0:11:22:a9:ee:66
--
```

Deleting a Datalink

Use the `dladm delete-phys` command to remove a datalink from the system.

Removing a datalink is only loosely connected to the removal of a physical NIC. For example, if a physical NIC is removed from the system, the datalink configuration that is associated with that NIC remains because the software layer is no longer bound to the hardware layer, as described in “Comparing the Oracle Solaris 10 Network Protocol Stack to the Oracle Solaris 11 Network Protocol Stack” in “Transitioning From Oracle Solaris 10 to Oracle Solaris 11.2 ”. Thus you can still use the datalink configuration on a different underlying physical NIC by assigning that datalink's name to the other NIC's associated link.

If you detach a NIC without replacing it, and you no longer need its datalink configuration, then you can delete the datalink as follows:

```
# dladm delete-phys datalink
```

Tip - To confirm whether a datalink's NIC had been removed, use the `dladm show-phys -P` command. The output provides a FLAGS column in which the `r` flag indicates whether the physical device that is associated with a physical link has been removed.

Renaming a Datalink

Use the `dladm rename-link` command to rename a datalink. On an Oracle Solaris system, the OS automatically provides generic names to all datalinks. For more information about generic datalink names, see “About Datalink Configuration” on page 23.

By default, these generic names use the naming format `net`*n*, such as `net0`, `net1`, `net2`, and so on. Because the OS manages these names, you would not rename datalinks as a regular part of your administrative tasks. For a procedure that requires changing link names, see “How to Move IP Configuration From One Network Device to Another Device” on page 38.

Obtaining Runtime Statistics for Datalinks

You use the `dlstat` command to obtain runtime datalink statistics for all types of datalinks. When used by itself with no other options, the `dlstat` displays statistical information about all of the datalinks that are on the system, as shown in the following output:

```
% dlstat
          LINK    IPKTS   RBYTES    OPKTS   OBYTES
          net0   58.00K    9.52M    5.61K    1.91M
```

For more information about using the `dlstat` command, see Chapter 8, “Monitoring Network Traffic and Resource Usage,” in “Managing Network Virtualization and Network Resources in Oracle Solaris 11.2 ”. See also the `dlstat`(1M) man page.

Customizing Datalink Properties

In addition to performing basic datalink configuration, you can also use the `dladm` command to set datalink properties and customize them according to the requirements of your network.

The following three `dladm` subcommands are used to administer datalink properties:

`dladm show-linkprop -p` *propertydatalink*	Displays the properties of a datalink and its current values. If you do not use the `-p` *property* option, then all of the properties of the datalink are displayed. If you do not specify a datalink, then all of the properties of all of the datalinks are displayed.
`dladm set-linkprop -p` *property=value datalink*	Assigns a value to a datalink's property.
`dladm reset-linkprop -p` *property datalink*	Resets a specific property of a datalink to its default value.

Datalink properties that you can customize depend on the properties a specific NIC driver supports.

Datalink properties that are configurable by using the `dladm` command fall into one of two categories:

- Public properties – These properties can be applied to any driver of the given media type such as link speed, auto-negotiation for Ethernet, or the maximum transmission unit (MTU) size that can be applied to all datalink drivers.
- Private properties – These properties are particular to a certain subset of NIC drivers for a given media type. These properties can be specific to that subset because they are closely related either to the hardware that is associated with the driver or to the details of the driver implementation itself, such as debugging-related tunables.

Link properties typically have default values. However, certain networking scenarios might require that you to change specific property values. For example, a NIC might be communicating with an old switch that does not properly perform auto-negotiation. Or, a switch might have been configured to support Jumbo frames. Or, driver specific properties that regulate packet transmission or packet reception might need to be modified for the specific driver.

Enabling Support for Jumbo Frames

MTU defines the size of the largest packet that a protocol can transmit from the system. By default, most NIC drivers define the MTU size to 1500. However, if Jumbo frames are traversing the network, the default value is insufficient. Support for Jumbo frames requires the MTU size to be at least 9000.

Note - The MTU property is common to both datalinks and IP interfaces, which means you can have one MTU value for a datalink and another MTU value for the IP interface that is configured over that link. The value of the datalink MTU impacts the possible values that you can set for an IP interface's MTU. For more information about the implications of this behavior when configuring the MTU property for datalinks and IP interfaces, see "Setting the MTU Property" on page 73.

Change the default value of the MTU size as follows:

```
# dladm set-linkprop -p mtu=new-size  datalink
```

After changing the MTU size, you can reconfigure an IP interface over the datalink.

The following example shows how to enable support for Jumbo frames. This example assumes that you have already removed any existing IP interface configuration over the datalink.

```
# dladm show-linkprop -p mtu net1
LINK     PROPERTY        PERM VALUE        EFFECTIVE    DEFAULT   POSSIBLE
net1     mtu             rw   1500         1500         1500      1500

# dladm set-linkprop -p mtu=9000 net1
# dladm show-link net1
LINK     CLASS     MTU      STATE     BRIDGE     OVER
web1     phys      9000     up        --         --
```

Modifying Link Speed Parameters

Most network setups consist of a combination of systems that have varying speed capabilities. Each system advertises speed capabilities to other systems on the network that informs how each system transmits and receives network traffic.

The following paired datalink properties regulate the speed capabilities that are advertised by a system:

- `adv_10gfdx_cap/en_10gfdx_cap`
- `adv_1000fdx_cap/en_1000fdx_cap`
- `adv_1000hdx_cap/en_1000hdx_cap`
- `adv_100fdx_cap/en_100fdx_cap`
- `adv_100hdx_cap/en_100hdx_cap`
- `adv_10fdx_cap/en_10fdx_cap`
- `adv_10hdx_cap/en_10hdx_cap`

Each link speed capability is referred to by a pair of properties: the advertised speed (`adv_*_cap`) and the enabled advertised speed (`en_*_cap`). Further, datalink speed information is also provided for both full-duplex and half-duplex capabilities, as designated by `*fdx*`

and *hdx* in the property names. The advertised speed property is a read-only property that indicates whether the specific datalink speed is advertised. You determine whether a specific datalink speed is advertised by setting the corresponding en_*_cap property.

By default, all of the speed and duplex capabilities of a datalink are advertised. However, cases might exist where a new system is communicating with an older system and auto-negotiation is disabled or unsupported. To enable communication between these two systems, the advertised speed between an older system and a newer system might need to be changed to a lower value. The Gigabit capabilities of the system might need to be switched off and only the slower speed capabilities are advertised. In this case, you would type the following command to switch off the advertisement of the Gigabit capabilities for both the full-duplex capability and the half-duplex capability:

```
# dladm set-linkprop -p en_1000fdx_cap=0 datalink
# dladm set-linkprop -p en_1000hdx_cap=0 datalink
```

To display the new values of these properties, use the dladm show-linkprop command as follows:

```
# dladm show-linkprop -p adv_10gfdx_cap datalink
# dladm show-linkprop -p adv_1000hdx_cap datalink
```

Normally, the values of a given enabled speed property and the corresponding advertised property are identical. However, if a NIC supports some advanced features such as Power Management, those features might set limits on the bits that are actually advertised between the host and its link partner. For example, with Power Management, the settings of the adv_*_cap properties might only be a subset of the settings of the en_*_cap properties.

Setting the STREAMS Module on Datalinks

You can set up to eight STREAMS modules to be pushed onto the stream when the datalink is opened. These modules are typically used by third-party networking software such as virtual private networks (VPNs) and firewalls. Documentation about such networking software is provided by the software vendor.

The list of modules to push on a specific datalink is controlled by the autopush property. In turn, the value of the autopush property is set by using the dladm set-linkprop command.

There is a separate autopush command that you can use to push modules onto the datalink's stream on a per-driver basis. This command uses a configuration file that is set up for each driver and which informs the command the modules to push. However, the driver is always bound to the NIC. If the datalink's underlying NIC is removed, then the link's autopush property information is lost as well.

Therefore, using the dladm command for this purpose is more preferable than using the autopush command. If both per-driver and per-link types of autopush configuration exist for a

specific datalink, the per-link information that is set with the `dladm set-linkprop` command is used, and the per-driver information is ignored.

To push modules to the STREAMS when the datalink is opened, use the same `dladm set-linkprop` command to specify modules for the `autopush` property. For example, you would push the `vpnmod` and `bufmod` modules on top of the link `net0` as follows:

```
# dladm set-linkprop -p autopush=vpnmod.bufmod net0
```

Obtaining Status Information for Datalink Properties

To obtain information about datalink properties, you can use either of the following commands:

- `dladm show-linkprop -p` *property datalink*
- `dladm show-ether` *datalink*

Displaying Datalink Properties

To display a complete list of datalink properties, type the command without specifying a property, as shown in the following example:

```
# dladm show-linkprop net1
LINK     PROPERTY        PERM VALUE        EFFECTIVE    DEFAULT   POSSIBLE
net1     speed           r-   0            0            0         --
net1     autopush        rw   --           --           --        --
net1     zone            rw   --           --           --        --
net1     duplex          r-   unknown      unknown      unknown   half,full
net1     state           r-   up           up           up        up,down
net1     adv_autoneg_cap --   --           --           0         1,0
net1     mtu             rw   1500         1500         1500      1500
net1     flowctrl        --   --           --           no        no,tx,rx,bi,
                                                                  pfc,auto
net1     adv_10gfdx_cap  r-   --           --           0         1,0
net1     en_10gfdx_cap   --   --           --           0         1,0
net1     adv_1000fdx_cap r-   --           --           0         1,0
net1     en_1000fdx_cap  --   --           --           0         1,0
net1     adv_1000hdx_cap r-   --           --           0         1,0
net1     en_1000hdx_cap  --   --           --           0         1,0
net1     adv_100fdx_cap  r-   --           --           0         1,0
net1     en_100fdx_cap   --   --           --           0         1,0
net1     adv_100hdx_cap  r-   --           --           0         1,0
net1     en_100hdx_cap   --   --           --           0         1,0
net1     adv_10fdx_cap   r-   --           --           0         1,0
net1     en_10fdx_cap    --   --           --           0         1,0
net1     adv_10hdx_cap   r-   --           --           0         1,0
net1     en_10hdx_cap    --   --           --           0         1,0
net1     maxbw           rw   --           --           --        --
net1     cpus            rw   --           --           --        --
```

Displaying Ethernet Property Values

If you do not specify any options with the `dladm show-ether` command, only the current Ethernet property values for the datalink are displayed. To obtain information beyond what is provided by default, use the -x option, as shown in the following example:

```
# dladm show-ether -x net1
LINK     PTYPE       STATE    AUTO  SPEED-DUPLEX             PAUSE
net1     current     up       yes   1G-f                     both
--       capable     --       yes   1G-fh,100M-fh,10M-fh     both
--       adv         --       yes   100M-fh,10M-fh           both
--       peeradv     --       yes   100M-f,10M-f             both
```

With the -x option, the command also displays the built-in capabilities of the specified link, as well as the capabilities that are currently advertised between the host and the link partner.

The following information is displayed in this example:

- For the Ethernet device's current state, the link is up and functioning at 1 Gigabits per second at full duplex. Its auto-negotiation capability is enabled and has bidirectional flow control, in which both the host and link partner can send and receive pause frames. This information is displayed in the first row of the output.
- Subsequent rows of the example's output display information about datalink speed capabilities, actual datalink speeds that are advertised, as well as information from the peer system as follows:
  - The capabilities of the Ethernet device are listed. The negotiation type can be set to automatic. In addition, the device can support speeds of 1 Gigabits per second, 100 megabits per second, and 10 megabits per second, at both full and half duplex. Likewise, pause frames can be received or sent in both directions between host and link partner.
  - The capabilities of `net1` are advertised as follows: autonegotiation, speed-duplex, and flow control of pause frames.
  - Similarly, `net1`'s link or peer partner advertises the following capabilities: autonegotiation, speed-duplex, and flow control of pause frames.

Additional `dladm` Configuration Tasks

This section describes additional configuration procedures that are simplified by using the `dladm` command, such as switching primary interfaces and performing dynamic reconfiguration (DR).

How to Move IP Configuration From One Network Device to Another Device

Use the following procedure if you need to preserve the IP configuration that is associated with one network device and then move that configuration to another network device. You might perform this procedure as a prelude to removing a network card from the system or when changing a network cable connection.

For example purposes only, this procedure describes how to preserve the IP configuration that is associated with the `net0` (`e1000g0`) device, then apply the configuration to the `nge0` device.

1. **Become an administrator.**

2. **Display the physical link to device mappings on the system.**

   The following example assumes that the IP configuration for `e1000g0` is down for whatever reason and therefore the configuration needs to be moved to `nge0`:

   ```
   # dladm show-phys
   LINK    MEDIA    STATE    SPEED    DUPLEX    DEVICE
   net0    Ethernet down     0        unknown   e1000g0
   net1    Ethernet down     0        unknown   e1000g1
   net2    Ethernet up       1000     full      nge0
   net3    Ethernet down     0        unknown   nge1
   ```

3. **Disable the IP configuration on the datalink temporarily, leaving its persistent settings intact.**

   ```
   # ipadm disable-if interface
   ```

   For example, you would disable the IP configuration on `net0` as follows:

   ```
   # ipadm disable-if net0
   ```

   This step enables you to rename the datalink without having to recreate its IP configuration.

4. **Rename the datalink.**

   For example, you might rename the `net0` datalink as follows:

   ```
   # dladm rename-link net0 oldnet0
   ```

5. **Assign the primary link name to the datalink that is designated to become the primary device.**

   ```
   # dladm rename-link new-link primary-link
   ```

   For example, you would reassign the `net0` link name to the `net2` datalink as follows:

   ```
   # dladm rename-link net2 net0
   ```

6. **Re-enable the IP configuration on the newly name datalink. For example:**

```
# ipadm enable-if -t net0
```

Example 2-1 Removing a Datalink Interface

When you perform a fresh installation, all of the datalinks on the system are automatically assigned generic names that use the naming convention, `net0`, `net1`, and `net`*N*, and so on, depending on the total number of network devices on a system. After the installation, you can assign different names to the datalinks. The following example shows how to change an IP address that was initially provided for an interface, which involves first removing the existing interface:

```
# ipadm delete-ip net0
# ipadm create-ip net0
# ipadm create-addr -T addrconf net0/new-add
```

For more information, see Chapter 2, Administering Datalink Configuration in Oracle Solaris.

▼ How to Replace a Network Interface Card With Dynamic Reconfiguration

The following procedure applies only to systems that support dynamic reconfiguration (DR). It specifically refers to configuration steps after DR is completed. You no longer need to reconfigure network links after you complete the DR process. Instead, you just transfer the link configurations of the removed NIC to the replacement NIC.

The procedure does not describe the steps to perform DR itself. Consult your system documentation for that information.

For an introduction to DR, see Chapter 2, “Dynamically Configuring Devices,” in “Managing Devices in Oracle Solaris 11.2 ”.

Before You Begin Make sure that you complete the following steps first:

- Ensure that your system supports DR.
- Consult the appropriate manual that describes DR on your system.

  To locate current documentation about DR on Sun servers from Oracle, search for "dynamic reconfiguration" on http://www.oracle.com/technetwork/indexes/documentation/index.html.

  For information about performing DR in the Oracle Solaris Cluster environment, see “Oracle Solaris Cluster System Administration Guide ”.

1. **Become an administrator.**

2. **(Optional) Display information about the physical attributes of datalinks and their respective locations on the system.**

   ```
   # dladm show-phys -L
   ```

   For more information about the type of information that is displayed by the `dladm show-phys -L` command, refer to the `dladm`(1M) man page.

3. **Perform the DR process, as described in your system's documentation.**

4. **After you have installed the replacement NIC, proceed as follows, depending on the circumstance that applies:**

   - **If you inserted the replacement NIC into the same slot as the old NIC, proceed to Step 5.**

     With the new NIC using the location that was previously occupied by the old NIC, the new NIC inherits the link name and the configuration of the old NIC.

   - **If you inserted the replacement NIC into a different slot, and the new NIC needs to inherit the datalink configuration of the removed NIC, rename the link as follows:**

     ```
     # dladm rename-link new-datalink old-datalink
     ```

new-datalink	Refers to the datalink of the replacement NIC that is in a different slot from the location from which the old NIC was removed.
old-datalink	Refers to the datalink name that is associated with the old NIC that was removed.

     Note - In this scenario, the slot from which the old NIC was removed must remain empty.

     For example, the NIC in slot 1 is removed, and then the new NIC is inserted in slot 2. No NIC is inserted in slot 1. Assume that the datalink on slot 1 is `net0`, and the datalink on slot 2 is `net1`. You would specify that the datalink of the new NIC inherit the datalink configuration of the old NIC as follows:

     ```
     # dladm rename-link net1 net0
     ```

5. **Complete the DR process by enabling the new NIC's resources so that they are available for use.**

   For example, you can use the `cfgadm` command to configure the NIC. For more information see the `cfgadm`(1M) man page.

6. **(Optional) Display link information.**

   You can use either the `dladm show-phys` command or the `dladm show-link` command to display information about the datalinks.

Example 2-2 Performing Dynamic Reconfiguration by Installing a New Network Card

The following example shows how a `bge` card with link name `net0` is replaced by an `e1000g` card. The link configurations of `net0` are transferred from `bge` to `e1000g` after `e1000g` is connected to the system.

```
# dladm show-phys -L
LINK     DEVICE     LOCATION
net0     bge0       MB
net1     ibp0       MB/RISER0/PCIE0/PORT1
net2     ibp1       MB/RISER0/PCIE0/PORT2
net3     eoib2      MB/RISER0/PCIE0/PORT1/cloud-nm2gw-2/1A-ETH-2
```

You would perform the DR-specific steps such as using the `cfgadm` command to remove the `bge` card and then install the `e1000g` card in its place. After the card is installed, the datalink of `e1000g0` automatically assumes the name `net0` and inherits the link's configuration.

```
# dladm show-phys -L
LINK     DEVICE     LOCATION
net0     e1000g0    MB
net1     ibp0       MB/RISER0/PCIE0/PORT1
net2     ibp1       MB/RISER0/PCIE0/PORT2
net3     eoib2      MB/RISER0/PCIE0/PORT1/cloud-nm2gw-2/1A-ETH-2

# dladm show-link
LINK     CLASS     MTU    STATE    OVER
net0     phys      9600   up       ---
net1     phys      1500   down     ---
net2     phys      1500   down     --
net3     phys      1500   down     ---
```

▼ SPARC: How to Ensure That the MAC Address of Each Interface Is Unique

Every SPARC based system has a system-wide MAC address, which by default is used by all interfaces. However, some applications require every interface on a host to have a unique MAC address. Certain types of interface configuration such as link aggregations and IP network multipathing (IPMP) similarly require that interfaces must have their own MAC addresses.

The EEPROM parameter `local-mac-address?` determines whether all interfaces on a SPARC based system use the system-wide MAC address or a unique MAC address. The following procedure describes how to use the `eeprom` command to check the current value of the `local-mac-address?` parameter and change it, if necessary.

1. **Become an administrator.**

2. **Determine whether all of the interfaces on the system currently use the system-wide MAC address.**

```
# eeprom local-mac-address?
local-mac-address?=false
```

In the previous output, the `local-mac-address?=false` setting indicates that all of the interfaces use the system-wide MAC address. You must change the value of the `local-mac-address?=false` setting to `local-mac-address?=true` before any interfaces can become members of an IPMP group, for example.

Note - You should also make this change when configuring link aggregations.

---

3. **Change the value of the `local-mac-address?` setting as follows:**

```
# eeprom local-mac-address?=true
```

When you reboot the system, the interfaces that have factory-installed MAC addresses will use these factory settings rather than the system-wide MAC address. Interfaces without factory-installed MAC addresses will continue to use the system-wide MAC address.

4. **Check the MAC addresses of all of the interfaces on the system.**

Look for cases where multiple interfaces have the same MAC address. In the following example, two interfaces use the system-wide MAC address `8:0:20:0:0:1`.

```
# dladm show-linkprop -p mac-address
LINK     PROPERTY        PERM VALUE        EFFECTIVE    DEFAULT   POSSIBLE
net0     mac-address     rw   0:14:4f:f9:b1:a9 0:14:4f:f9:b1:a9 0:14:4f:f9:b1:a9 --
net3     mac-address     rw   0:14:4f:fb:9a:d4 0:14:4f:fb:9a:d4 0:14:4f:fb:9a:d4 --
net2     mac-address     rw   0:14:4f:f9:c:d 0:14:4f:f9:c:d 0:14:4f:f9:c:d --
net1     mac-address     rw   0:14:4f:fa:ea:42 0:14:4f:fa:ea:42 0:14:4f:fa:ea:42 --
```

5. **(Optional) If necessary, manually configure the remaining interfaces so that all interfaces have unique MAC addresses.**

```
# dladm set-linkprop -p mac-address=mac-address  interface
```

Note - This step is only required if two or more network interfaces have the same MAC address.

In the previous example, you would need to configure `net0` and `net1` with locally administered MAC addresses. For example, to reconfigure `net0` with the locally administered MAC address `06:05:04:03:02`, you would type the following command:

```
# dladm set-linkprop -p mac-address=06:05:04:03:02 net0
```

6. **Reboot the system.**

♦ ♦ ♦ CHAPTER 3

Configuring and Administering IP Interfaces and Addresses in Oracle Solaris

This chapter describes how to configure a network that implements IPv4 and IPv6 addressing. Many of the tasks in this chapter apply to both IPv4 and IPv6-enabled networks. Procedures that pertain to IPv4 or IPv6 networks only are indicated as such.

Prior to configuring your network, review the IP-related planning tasks that are described in "Planning for Network Deployment in Oracle Solaris 11.2 ".

For information about administering other TCP/IP properties such global packet forwarding and transport layer services, see "Administering Transport Layer Services" in "Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 ".

This chapter contains the following topics:

Administering Network Configuration by Using the `ipadm` Command

The `ipadm` command replaces the `ifconfig` command as the primary means for configuring IP interfaces.

The `ipadm` command also replaces the `ndd` command for configuring properties of the following TCP/IP protocols:

- IP
- Address Resolution Protocol (ARP)
- Stream Control Transmission Protocol (SCTP)
- Internet Control Messaging Protocol (ICMP)
- Upper layer protocols such as TCP and the User Datagram Protocol (UDP)

As a tool for configuring interfaces, the `ipadm` command offers the following advantages over the `ifconfig` command:

- Provides an object-oriented subcommand structure that is superior to the structure that is provided by the `ifconfig` command. This change should ultimately make network configuration procedures more easily understood.
- Is capable of making network configuration changes persistent, unlike the `ifconfig` command.
- Supports a parsable output option that can be useful for scripting.

As a tool for setting protocol properties, the `ipadm` command provides the following advantages over the `ndd` command:

- Provides more extensive property information than the `ndd` command, for example, a property's current and default value, as well as the range of possible values.
- Sets property values persistently (or temporarily). The `ndd` command only sets property values temporarily.
- Supports a parsable output option that can be useful for scripting.

To compare the `ipadm` command with the `ifconfig` and `ndd` commands, see "Network Administration Command Changes" in "Transitioning From Oracle Solaris 10 to Oracle Solaris 11.2 ".

For more information about the `ipadm` command, see the `ipadm`(1M) man page.

Configuring IPv4 Interfaces

The following procedure and examples describe how to configure a network that uses IPv4 addresses.

▼ How to Configure an IPv4 Interface

Before You Begin Check which NCP is active on the system to make sure that you are applying the configuration to the correct profile. See Example 6-6.

1. **Become an administrator.**

2. Create the interface.

```
# ipadm create-interface-class  interface
```

interface-class Refers to one of three classes of interfaces that you can create:

- IP interface

  This interface class is the most common class that you create when performing network configuration. To create this interface class, use the `create-ip` subcommand.

- STREAMS virtual network interface (VNI interface)

  To create this interface class, use the `create-vni` subcommand.

  Starting with Oracle Solaris 11.2, you can name VNI interfaces more arbitrarily. Previously, a VNI interface name must have included "vni" in its prefix, for example `vni0`. This requirement no longer applies. For more information about VNI devices and interfaces, see the `vni`(7d) and `ipadm`(1M) man pages.

- IPMP interface

  This interface class is used when you configure IPMP groups. To create this interface class, use the `create-ipmp` subcommand. For more information about IPMP groups, see Chapter 2, “About IPMP Administration,” in “Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 ”.

interface Refers to the name of the interface. The name is identical to the name of the datalink over which the interface is being created. To display the datalinks that are on a system, use the `dladm show-link` command.

3. Configure the IP interface with a valid IP address by using one of the following commands:

- Configure a static IP address:

  ```
  # ipadm create-addr -a address [interface | addrobj]
  ```

  -a *address* Specifies the IP address to configure on the interface.

Note - Tunnel configuration typically requires two addresses for the tunnel interface: a local address and a remote address. For information about local and remote addresses and tunnel configuration, see Chapter 5, “Administering IP Tunnels,” in “Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 ”.

For numeric IP addresses, use Classless Inter-Domain Routing (CIDR) notation. If you do not use CIDR notation, the netmask

is determined by using the `svc:/system/name-service/switch:default` netmask database search order or by using *classful address semantics*.

Optionally, you can specify a host name instead of a numeric IP address. Using a host name is valid if a corresponding numeric IP address is defined for that host name in the `/etc/hosts` file. If no numeric IP address is defined in the file, then the numeric value is uniquely obtained by using the resolver order that is specified for `host` in the `name-service/switch` service. If multiple entries exist for a given host name, an error is generated.

Note - During the boot process, IP addresses are configured prior to naming services being brought online. Therefore, you must ensure that any host name that is used in the network configuration is defined in the `/etc/hosts` file.

[*interface* | *addrobj*] In Oracle Solaris, each address is identified by a corresponding address object and represented in the command by *addrobj*. For any subsequent configuration on the address, you would refer to the address object instead of the actual IP address. For example, you would type `ipadm show-addr` *addrobj* or `ipadm delete-addr` *addrobj*. To have the address object name generated automatically, specify only the interface name for *interface*. To manually name the address object, provide the address object name directly.

- If you specify the interface name, then an address object is automatically named with the format *interface*/*address-family*. *Address family* is either `v4` for an IPv4 address or `v6` for an IPv6 address. If multiple addresses are configured on an interface by using automatically generated address object names, then alphabetic letters are appended to the address object names so that they are unique. For example, `net0/v4`, `net0/v4a`, `net0/v4b`, `net0/v6`, `net0/v6a`, and so on.
- If you manually name the address object for *addrobj*, you must use the format *interface*/*user-specified-string*. *User-specified-string* refers to a string of alphanumeric characters that begins with an alphabetic letter and has a maximum length of 32 characters. For example, you can name address objects `net0/static`, `net0/static1`, `net1/private`, and so on.

- Configure a non-static address.

  ```
  # ipadm create-addr -T address-type [interface | addrobj]
  ```

  where *address-type* is either `dhcp` or `addrconf`. The `addrconf` argument refers to automatically generated IPv6 addresses.

For a more detailed explanation of the *interface* and *addrobj* options, refer to the previous description for creating static addresses.

4. **(Optional) Display information about the newly configured IP interface.**

   You can use the following commands, depending on the information that you want to check:

   `# ipadm` *interface*

   If you do not specify a subcommand, information for all of the interfaces on the system is displayed.

   `# ipadm show-if` *interface*

   If you do not specify an *interface*, information for all of the interfaces on the system is displayed.

   `# ipadm show-addr` *interface*`|`*addrobj*

   If you do not specify an *interface* or *addrobj*, information for all of the address objects is displayed.

   For more information about the output of the `ipadm show-*` subcommand, see "Monitoring IP Interfaces and Addresses" on page 78.

5. **If you are configuring a static IP address that uses a host name, add the entries for the IP address to the `/etc/hosts` file.**

   The entries in this file consist of IP addresses and their corresponding host names.

   Note - If you are configuring a DHCP address, you do not need to update the `/etc/hosts` file.

   ---

6. **Define the default route.**

   `# route -p add default` *address*

   You can verify the contents of the routing table with the `netstat -r` command.

   For more information about managing routes, see the `route`(1M) and "Creating Persistent (Static) Routes" on page 62.

Example 3-1 Configuring an IPv4 Interface With a Static IP Address

The following example shows how to configure an interface with a static IP address. The example begins with enabling the `DefaultFixed` NCP on the system to ensure that the `dladm` and `ipadm` commands do not modify a reactive NCP, which could negate any manual network configuration that you perform, depending on your environment.

```
# netadm enable -p ncp DefaultFixed
```

```
# dladm show-phys
LINK     MEDIA        STATE     SPEED     DUPLEX     DEVICE
net3     Ethernet     up        100Mb     full       bge3

# dladm show-link
LINK     CLASS     MTU     STATE     OVER
net3     phys      1500    up        --       --

# ipadm create-ip net3
# ipadm create-addr -a 192.168.84.3/24 net3
net3/v4

# ipadm
NAME      CLASS/TYPE     STATE     UNDER     ADDR
lo0       loopback       ok        --        --
lo0/v4     static        ok        --     127.0.0.1/8
lo0/v6     static        ok        --     ::1/128
net3      ip             ok        --        --
net3/v4   static         ok        --     192.168.84.3/24

# vi /etc/hosts
# Internet host table
# 127.0.0.1     localhost
10.0.0.14       myhost
192.168.84.3    sales1

# route -p add default 192.168.84.1
# netstat -r
Routing Table: IPv4
Destination          Gateway              Flags Ref   Use        Interface
-------------------- -------------------- ----- ----- ---------- ---------
default              192.168.84.1         UG    2     10466
192.168.84.0         192.168.84.3         U     3     1810       net0
localhost            localhost            UH    2     12         lo0

Routing Table: IPv6
Destination/Mask            Gateway                 Flags Ref   Use    If
--------------------------- ----------------------- ----- --- ------- -----
solaris                       solaris                 UH      2     156  lo0
```

If `sales1` is already defined in the `/etc/hosts` file, you can use that host name when assigning the following address:

```
# ipadm create-addr -a sales1 net3
net3/v4
```

Example 3-2 Configuring a Network Interface To Receive an IP Address From a DHCP Server

In the following example, the IP interface is configured to receive its address from a DHCP server. DHCP typically also installs a default route. Therefore, this example does include a step for manually adding a default route.

```
# dladm show-phys
LINK     MEDIA        STATE     SPEED     DUPLEX     DEVICE
```

```
net3     Ethernet     up        100Mb     full       bge3

# dladm show-link
LINK     CLASS     MTU     STATE     OVER
net3     phys      1500    up        --       --

# ipadm create-ip net3
# ipadm create-addr -T dhcp net3
net3v4

# ipadm
NAME      CLASS/TYPE     STATE     UNDER     ADDR
lo0       loopback       ok        --        --
l0/v4     static         ok        --        127.0.0.1/8
net3      ip             ok        --        --
net3/v4   dhcp           ok        --        10.0.1.13/24
```

Configuring IPv6 Interfaces

As an initial step to use IPv6 addressing on a network, you must configure IPv6 on the system's IP interface. During the installation process, you can enable IPv6 on one or more of a system's interfaces.

If you enable IPv6 support during installation, then after the installation has completed, the following IPv6-related files and tables are in place:

- The `name-service/switch` SMF service is modified to accommodate lookups using IPv6 addresses.
- The IPv6 address selection policy table is created. This table prioritizes the IP address format to use for transmissions over an IPv6-enabled interface.

▼ How to Configure a System For IPv6

The following procedure explains how to enable IPv6 for an interface that was added after an Oracle Solaris installation. You begin the IPv6 configuration process by enabling IPv6 on the interfaces of all of the systems that will become IPv6 nodes. Typical IPv6 deployments use autoconfiguration to configure IP interfaces. An `autoconf` IP address assigns a link-local address and discovers prefixes and routers that are in use on the subnet. You then can tailor the node's configuration based on its function in the IPv6 network, either as a host, server, or a router. Interfaces that are set up for `autoconf` will also automatically request DHCPv6 address information. To enable only static IPv6 addresses, without autoconfiguration or DHCPv6, use the `ipadm` command with the appropriate options to create a link-local address on the interface without adding any other dynamically assigned addresses. See “Migrating From an IPv4 Network to an IPv6 Network” on page 59 for an example.

Note - If the interface is on the same link as a router that currently advertises an IPv6 prefix, the interface obtains that site prefix as part of its autoconfigured addresses. For more information, refer to "How to Configure an IPv6-Enabled Router" in "Configuring an Oracle Solaris 11.2 System as a Router or a Load Balancer ".

1. **(Optional) Configure the IP interface by using the `ipadm create-ip` command with the appropriate options.**

   ```
   # ipadm create-ip interface
   ```

   For example, you would configure an IP interface for `net0` as follows:

   ```
   # ipadm create-ip net0
   ```

   If the interface has already been configured for use with IPv4, this step is not required. Refer to How to Configure an IPv4 Interface for general instructions on configuring an IP interface.

2. **Assign the IP address or addresses.**

   Note - When you assign the IP address, make sure to use the correct option for assigning an IPv6 address:

   ```
   # ipadm create-addr -T addrconf interface
   ```

   To add more addresses, use the following syntax:

   ```
   # ipadm create-addr -a ipv6-address interface
   ```

3. **(Optional) Create a static IPv6 default route.**

   ```
   # /usr/sbin/route -p add -inet6 default ipv6-address
   ```

   Note - As part of autoconfiguration, `in.ndpd` adds default routes as they are discovered, which might result in multiple default routes being made available, including any manually configured default routes. The system automatically makes a default route selection based on all of the available routes, which means that a manually configured default route might not be used every time.

4. **(Optional) Create an `/etc/inet/ndpd.conf` file that defines the parameters for the interface variables that are on the node.**

If you need to create temporary addresses for the host's interface, refer to "Using Temporary Addresses for an IPv6 Interface" on page 54. For details about `/etc/inet/ndpd.conf`, refer to the `ndpd.conf`(4) man page.

5. **(Optional) Display the status of the IP interfaces with their IPv6 configurations as follows:**

```
# ipadm show-addr
```

Example 3-3 Enabling an IPv6 Interface After Installation

The following example shows how to enable IPv6 on the `net0` interface. Before you begin, check the status of all of the interfaces that are configured on the system.

```
# ipadm show-addr
ADDROBJ   TYPE     STATE   ADDR
lo0/v4    static   ok      127.0.0.1/8
net0/v4   static   ok      172.16.27.74/24
```

As shown in the previous output, only the `net0` interface is currently configured for this system. If the `net0` interface has not been configured yet, use the `ipadm create-ip net0` command to bring the interface up.

IPv6 is then enabled on this interface as follows:

```
# ipadm create-addr -T addrconf net0
# ipadm create-addr -a 2001:db8:3c4d:15::203/64 net0

# ipadm show-addr
ADDROBJ      TYPE       STATE   ADDR
lo0/v4       static     ok      127.0.0.1/8
net0/v4      static     ok      172.16.27.74/24
net0/v6      addrconf   ok      fe80::203:baff:fe13:14e1/10
lo0/v6       static     ok      ::1/128
net0/v6a     static     ok      2001:db8:3c4d:15::203/64

# route -p add -inet6 default fe80::203:baff:fe13:14e1
```

Next Steps

- For information about how to configure the IPv6 node as a router, see "Configuring an IPv6 Router" in "Configuring an Oracle Solaris 11.2 System as a Router or a Load Balancer ".
- For information about how to tailor the node as a server, see "Configuring IPv6-Enabled Interfaces on Servers" on page 58.

Using Temporary Addresses for an IPv6 Interface

An *IPv6 temporary address* includes a randomly generated 64-bit number as the interface ID instead of an interface's MAC address. You can use temporary addresses for any interface on an IPv6 node that you want to keep anonymous. For example, you might want to use temporary addresses for the interfaces of a host that needs to access public web servers. Temporary addresses implement IPv6 privacy enhancements. These enhancements are described in RFC 3041, which is available at "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" (http://www.rfc-editor.org/rfc/rfc3041.txt).

You enable a temporary address in the `/etc/inet/ndpd.conf` file for one or more interfaces, if needed. However, unlike standard, autoconfigured IPv6 addresses, a temporary address consists of the 64-bit subnet prefix and a randomly generated 64-bit number. This random number becomes the interface ID segment of the IPv6 address. A link-local address is not generated with the temporary address as the interface ID.

Be aware that temporary addresses have a default *preferred lifetime* of one day. When you enable temporary address generation, you can also configure the following variables in the `/etc/inet/ndpd.conf` file:

valid lifetime TmpValidLifetime	Time span in which the temporary address exists, after which the address is deleted from the host.
preferred lifetime TmpPreferredLifetime	Elapsed time before the temporary address is deprecated. This time span should be shorter than the valid lifetime.
address regeneration	Duration of time before the expiration of the preferred lifetime, during which the host should generate a new temporary address.

You express the duration of time for temporary addresses as follows:

n	*n* number of seconds, which is the default
n `h`	*n* number of hours (`h`)
n `d`	*n* number of days (`d`)

▼ How to Configure a Temporary IPv6 Address

1. If necessary, enable IPv6 on the host's interfaces

Refer to "How to Configure a System For IPv6" on page 51.

2. **Edit the `/etc/inet/ndpd.conf` file to turn on temporary address generation.**

   - To configure temporary addresses on all of the interfaces of a host, add the following line to the `/etc/inet/ndpd.conf` file:

     `ifdefault TmpAddrsEnabled true`

   - To configure a temporary address for a specific interface, add the following line to the `/etc/inet/ndpd.conf` file:

     `if` *interface* **`TmpAddrsEnabled true`**

3. **(Optional) Specify the valid lifetime for the temporary address.**

   `ifdefault TmpValidLifetime` *duration*

   This syntax specifies the valid lifetime for all of the interfaces on a host. The value for *duration* should be in seconds, hours, or days. The default valid lifetime is 7 days. You can also use `TmpValidLifetime` with the `if` *interface* keywords to specify the valid lifetime for a temporary address of a particular interface.

4. **(Optional) Specify a preferred lifetime for the temporary address, after which the address is deprecated.**

   `if` *interface* **`TmpPreferredLifetime`** *duration*

   This syntax specifies the preferred lifetime for the temporary address of a particular interface. The default preferred lifetime is one day. You can also use `TmpPreferredLifetime` with the `ifdefault` keyword to specify the preferred lifetime for the temporary addresses on all of the interfaces of a host.

   Note - Default address selection gives a lower priority to IPv6 addresses that have been deprecated. If an IPv6 temporary address is deprecated, default address selection chooses a non-deprecated address as the source address of a packet. A non-deprecated address could be the automatically generated IPv6 address or possibly the interface's IPv4 address. For more information about default address selection, see "Administering Default Address Selection" in "Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 ".

   ---

5. **(Optional) Specify the lead time in advance of address deprecation, during which the host should generate a new temporary address.**

   `ifdefault TmpRegenAdvance` *duration*

   This syntax specifies the lead time in advance of address deprecation for the temporary addresses of all of the interfaces on a host. The default is 5 seconds.

6. **Change the configuration of the `in.ndpd` daemon as follows:**

```
# pkill -HUP in.ndpd
# /usr/lib/inet/in.ndpd
```

7. **Verify that temporary addresses have been created by issuing the `ipadm show-addr` command, as shown in Example 3-4.**

   The command output displays the `t` flag on the `CURRENT` field of temporary addresses.

Example 3-4 Displaying `ipadm show-addr` Command Output With Temporary Addresses Enabled

The following example shows the output of the `ipadm show-addr` command after temporary addresses are created. Note that only IPv6–related information is included in the sample output.

```
# ipadm show-addr -o all
ADDROBJ   TYPE     STATE CURRENT PERSISTENT ADDR
lo0/v6    static   ok    U----   ---        ::1/128
net0/v6   addrconf ok    U----   ---        fe80::a00:20ff:feb9:4c54/10
net0/v6a  static   ok    U----   ---        2001:db8:3c4d:15:a00:20ff:feb9:4c54/64
net0/?    addrconf ok    U--t-   ---        2001:db8:3c4d:15:7c37:e7d1:fc9c:d2cb/64
```

Note that for the address object `net0/?`, the `t` flag is set under the `CURRENT` field, indicating that the corresponding address has a temporary interface ID.

See Also

- To set up name service support for IPv6 addresses, see Chapter 4, “Administering Naming and Directory Services on an Oracle Solaris Client”.
- To configure IPv6 addresses for a server, see “How to Configure a User-Specified IPv6 Token” on page 57.
- To monitor activities on IPv6 nodes, see Chapter 1, “Administering TCP/IP Networks,” in “Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 ”.

Configuring an IPv6 Token

The 64-bit interface ID of an IPv6 address is also referred to as a *token*. During address autoconfiguration, the token is associated with the interface's MAC address. In most cases, non-routing nodes (IPv6 hosts and servers) should use their autoconfigured tokens.

However, using autoconfigured tokens can be a problem for servers with interfaces that are routinely swapped as part of system maintenance. When the interface card is changed, the MAC address is also changed. Servers that depend on having stable IP addresses can experience problems as a result. Various parts of the network infrastructure, such as Domain Name System (DNS) or Network Information System (NIS), might have stored specific IPv6 addresses for the interfaces of the server.

To avoid address change problems, you can manually configure a token to be used as the interface ID in an IPv6 address. To create the token, you specify a hexadecimal number of 64 bits or less to occupy the interface ID portion of the IPv6 address. During subsequent address

autoconfiguration, Neighbor Discovery does not create an interface ID that is based on the interface's MAC address. Instead, the manually created token becomes the interface ID. This token remains assigned to the interface, even when a card is replaced.

Note - The difference between user-specified tokens and temporary addresses is that temporary addresses are randomly generated, rather than explicitly created by a user.

▼ How to Configure a User-Specified IPv6 Token

The following procedure is particularly useful for servers with interfaces that are routinely replaced. You can also follow these steps to configure user-specified tokens on any IPv6 node.

1. **Verify that the interface that you want to configure with a token exists and that no IPv6 addresses are configured on that interface.**

```
# ipadm show-if
IFNAME   CLASS      STATE   ACTIVE   OVER
lo0      loopback   ok       yes     ---
net0     ip         ok       yes     ---

# ipadm show-addr
ADDROBJ      TYPE       STATE   ADDR
lo0/v4       static     ok      127.0.0.1/8
```

   The previous output shows that the network interface `net0` exists with no configured IPv6 address.

2. **Create one or more 64-bit hexadecimal numbers to be used as tokens for the node's interfaces that follows the following format:**

   xxxx:*xxxx*:*xxxx*:*xxxx*

3. **Configure each interface that will have a user-specified interface ID (token).**

```
# ipadm create-addr -T addrconf -i interface-ID  interface
```

   For example, you configure interface `net0` with a token as follows:

```
# ipadm create-addr -T addrconf -i ::1a:2b:3c:4d/64 net0
```

Note - After the address object has been created with the token, you can no longer modify the token.

---

4. **Update the IPv6 daemon with the changes.**

```
# pkill -HUP in.ndpd
```

Example 3-5 Configuring a User-Specified Token on an IPv6 Interface

The following example shows how to configure `net0` with an IPv6 address and a token.

```
# ipadm show-if
IFNAME   CLASS      STATE    ACTIVE     OVER
lo0      loopback   ok       yes        ---
net0     ip         ok       yes        ---

# ipadm show-addr
ADDROBJ      TYPE       STATE   ADDR
lo0/v4       static     ok      127.0.0.1/8

# ipadm create-addr -T addrconf -i ::1a:2b:3c:4d/64 net0
# pkill -HUP in.ndpd
# ipadm show-addr
ADDROBJ      TYPE       STATE   ADDR
lo0/v6       static     ok      ::1/128
net0/v6      addrconf   ok      fe80::1a:2b:3c:4d/10
net0/v6a     addrconf   ok      2002:a08:39f0:1:1a:2b:3c:4d/64
```

After the token is configured, the address object `net0/v6` has both a link-local address, as well as an address with `1a:2b:3c:4d` configured for its interface ID. Note that you can no longer modify the token for this interface after `net0/v6` is created.

See Also

- To update the name services with the IPv6 addresses of the server, see Chapter 4, “Administering Naming and Directory Services on an Oracle Solaris Client”.
- To monitor server performance, see Chapter 1, “Administering TCP/IP Networks,” in “Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 ”.

Configuring IPv6-Enabled Interfaces on Servers

When you plan for IPv6 addresses on a server, you must make a few decisions as you enable IPv6 on the server's interfaces. Your decisions affect the strategy to use for configuring the interface IDs (tokens) of an interface's IPv6 address.

▼ How to Enable IPv6 on a Server's Interfaces

The following procedure describes how to enable IPv6 on your network's servers. Some of the steps might vary depending on the manner in which you want to implement IPv6.

1. **Enable IPv6 on the server's IP interfaces.**

For step-by-step instructions, see “Configuring IPv6 Interfaces” on page 51.

2. **Ensure that an IPv6 subnet prefix is configured on a router that is on the same link as the server.**

   For more information, refer to “Configuring an IPv6 Router” in “Configuring an Oracle Solaris 11.2 System as a Router or a Load Balancer ”.

3. **Choose from one of the following strategies for assigning an interface ID to the server's IPv6-enabled interfaces.**

   By default, IPv6 address autoconfiguration uses the MAC address of an interface when creating the interface ID portion of the IPv6 address. If the IPv6 address of the interface is well known, swapping one interface for another interface can cause problems. The MAC address of the new interface will be different. During address autoconfiguration, a new interface ID is generated.

   - **For an IPv6-enabled interface that you do not plan to replace, use the autoconfigured IPv6 address.**

   - **For IPv6-enabled interfaces that must appear anonymous outside of the local network, consider using a randomly generated token for the interface ID.**

     For more information, see “How to Configure a Temporary IPv6 Address” on page 54

   - **For IPv6-enabled interfaces that you plan to swap on a regular basis, you can use static configuration, or you can create tokens for the interface IDs.**

     For more information, see “How to Configure a User-Specified IPv6 Token” on page 57.

Migrating From an IPv4 Network to an IPv6 Network

Note - Before you migrate from an IPv4 network to an IPv6 network, be aware that most migration plans involve running IPv4 and IPv6 together for an extended period of time, possibly indefinitely.

Prior to migrating from an IPv4 network to an IPv6 network, review the information in Chapter 2, “Planning for Using IPv6 Addresses,” in “Planning for Network Deployment in Oracle Solaris 11.2 ” to determine whether you need to perform any additional tasks.

The basic steps for migrating from an IPv4 network to an IPv6 network involves first removing all of the existing IPv4 DHCP and static IP addresses, then reconfiguring new IPv6 addresses,

as many as are required. If the new IPv6 interface is on the same link as a router that currently advertises an IPv6 prefix, the interface obtains the link prefix. For more information, see "Configuring an IPv6 Router" in "Configuring an Oracle Solaris 11.2 System as a Router or a Load Balancer ".

EXAMPLE 3-6 Migrating IPv4 Addresses to IPv6 Addresses

The following examples show how to migrate your existing IPv4 addresses to IPv6 addresses. The process begins by deleting all of the existing IPv4 DHCP and static IP addresses.

```
# ipadm show-addr net0/
ADDROBJ   TYPE     STATE   ADDR
lo0/v4    static   ok      127.0.0.1/8
net0/v4   static   ok      172.16.27.74/24
# ipadm delete-addr net0/v4
```

For instructions, see "Removing or Modifying an IP Interface Configuration" on page 77.

Next, the new IPv6 address is created by using the `ipadm create addr` command with the appropriate options and arguments.

For example, you can create a link-local and `addrconf` IPv6 address as follows:

```
# ipadm create-addr -T addrconf -p stateless=yes,stateful=yes net0/v6a
```

Create a static IPv6 address without DHCPv6 and an `addrconf` address as follows:

```
# ipadm create-addr -T addrconf -p stateless=no,stateful=no net0/v6a
# ipadm create-addr -T static -a a::b/64 net0/v6b
```

Create a static IPv6 address as follows:

```
# ipadm create-addr -T static -a a::b/64 net0/v6b
```

Display the new IPv6 configuration by using the `ipadm show-addr` command.

For additional IPv6 configuration steps (required and optional) that are not included this example, refer to "Configuring IPv6 Interfaces" on page 51.

Configuring Routing

This section contains the following topics:

Routing Tables and Routing Types

Both routers and hosts maintain a *routing table*. For example, the following routing table lists the IP addresses of networks that the system knows about, including the system's local default network. The table also lists the IP address of a gateway system for each known network. A *gateway* is a system that can receive outgoing packets and forward them one hop beyond the local network.

```
Routing Table: IPv4
Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
default              172.20.1.10          UG       1    532   net0
224.0.0.0            10.0.5.100           U        1      0   net1
10.0.0.0             10.0.5.100           U        1      0   net1
127.0.0.1            127.0.0.1            UH       1     57   lo0
```

You can configure two types of routing on an Oracle Solaris system: static and dynamic. You can configure either or both routing types on a single system. A system that implements *dynamic routing* relies on routing protocols, such as the Routing Information Protocol (RIP) for IPv4 networks and RIPng (RIP next generation) protocol for IPv6 networks, to route network traffic as well as update routing information in the table. With *static routing*, information is maintained manually by using the `route` command. For more information, see the `route`(1M) man page.

When you configure routing for the local network or an autonomous system (AS), consider which type of routing to support on particular routers and hosts. The following table shows the different types of routing and networking scenarios for which each routing type is best applied.

Routing Type	Best Uses
Static	Small networks and hosts that get their routes from a default router and default routers that only need to know about one or two routers on the next few hops.
Dynamic	Larger internetworks, including routers on local networks with many hosts and hosts on large autonomous systems. Dynamic routing is the best option for systems on most networks.
Combined static and dynamic	Routers that connect a statically routed network and a dynamically routed network and border routers that connect an interior autonomous system with external networks. Combining both static and dynamic routing on a system is a common practice.

The topology that is described in “IPv4 Autonomous System Topology” in “Planning for Network Deployment in Oracle Solaris 11.2 ” combines both static and dynamic routing.

Note - Two routes that are going to the same destination does not automatically cause a system to perform load balancing or failover. If you need these capabilities, use IPMP. For more information, see Chapter 2, “About IPMP Administration,” in “Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 ”.

Creating Persistent (Static) Routes

You use the `route` command to manually manipulate the network routing tables. To make the changes persistent across reboots, use the `-p` option. Because the `/etc/defaultrouter` file is deprecated in Oracle Solaris 11, you can no longer manage routes (default or otherwise) by using this file. Using the `route` command is the only way that you can manually make routes persistent across system reboots.

Note - The `route` command manipulates routes for the active profile *only*. The default route, as well as all other routes, potentially might be replaced if the active profile changes. However, this is not a concern if you always use the same profile on your system.

When adding routes persistently care should be taken to make sure that routes that you add do not already exist in the persistent configuration. If these routes already exist in the persistent configuration, the network routing tables could change without updating the persistent route. An example would be a situation where the system's default route is mapped to the system's primary interface (which is frequently the case after an Oracle Solaris installation). If you subsequently change the system's primary interface to another interface, then the system's default route should also be updated persistently. A best practice is to delete the persistent route configuration prior to adding the new route. For more information, see “Troubleshooting Issues When Adding a Persistent Route” in “Troubleshooting Network Administration Issues in Oracle Solaris 11.2 ”.

Note the following additional information about creating and displaying persistent routes:

- Use the `route` command with the `-p` option to persistently add a route:

  ```
  # route -p add default ip-address
  ```

  For routes that are created by using this method, use the `route -p show` command to display all of the persistent static routes:

  ```
  # route -p show
  ```

- Display the currently active routes on a system by using the `netstat` command with the following options:

  ```
  # netstat -rn
  ```

See the `netstat`(1M) and `route`(1M) man pages.

For more information, see the `netstat`(1M) and `route`(1M) man pages.

For information about creating and displaying default routes when using the reactive mode, see Chapter 5, “About Administering Profile-Based Network Configuration in Oracle Solaris”.

▼ How to Add a Static Route to the Routing Table

1. **View the current state of the routing table by using your regular user account.**

```
% netstat -rn
```

The output would be similar to the following:

```
Routing Table: IPv4
  Destination           Gateway           Flags  Ref     Use     Interface
-------------------- -------------------- ----- ----- ---------- ---------
192.168.5.125        192.168.5.10         U     1     5879       net0
224.0.0.0            198.168.5.10         U     1     0          net0
default              192.168.5.10         UG    1     91908
127.0.0.1            127.0.0.1            UH    1     811302     lo0

Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use    If
--------------------------- --------------------------- ----- --- ------- -----
::1                         ::1                         UH      2       0 lo0
```

2. **Become an administrator.**

3. **(Optional) Flush the existing entries in the routing table.**

```
# route flush
```

4. **Add a persistent route.**

```
# route -p add -net network-address -gateway gateway-address
```

`-p`	Creates a route that persists across system reboots. If you want the route to persist only for the current session, do not use the `-p` option.
`-net` *network-address*	Specifies that the route goes to the network with the address that is specified in *network-address*.
`-gateway` *gateway-address*	Indicates that the gateway system for the specified route has the IP address *gateway-address*.

Example 3-7 Adding a Static Route to the Routing Table

The following example shows how to add a static route to a router (Router 2). The static route is needed for the AS's border router, `10.0.5.150`. See Figure 3-1 for an illustration of this particular setup.

You would view the routing table on Router 2 as follows:

```
# netstat -rn
Routing Table: IPv4
Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
default              172.20.1.10          UG        1    249 ce0
224.0.0.0            172.20.1.10          U         1      0 ce0
10.0.5.0             10.0.5.20            U         1     78 bge0
127.0.0.1            127.0.0.1            UH        1     57 lo0

Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use    If
--------------------------- --------------------------- ----- --- ------- -----
::1                         ::1                         UH      2       0 lo0
```

The routing table indicates that there are two routes that Router 2 knows about. The default route uses Router 2's `172.20.1.10` interface as its gateway. The second route, `10.0.5.0`, was discovered by the `in.routed` daemon that is running on Router 2. The gateway for this route is Router 1 and it has the IP address `10.0.5.20`.

You would add a second route to network `10.0.5.0`, which has its gateway as the border router, as follows:

```
# route -p add -net 10.0.5.0/24 -gateway 10.0.5.150
add net 10.0.5.0: gateway 10.0.5.150
```

The routing table now has a route for the border router, which has the IP address `10.0.5.150`.

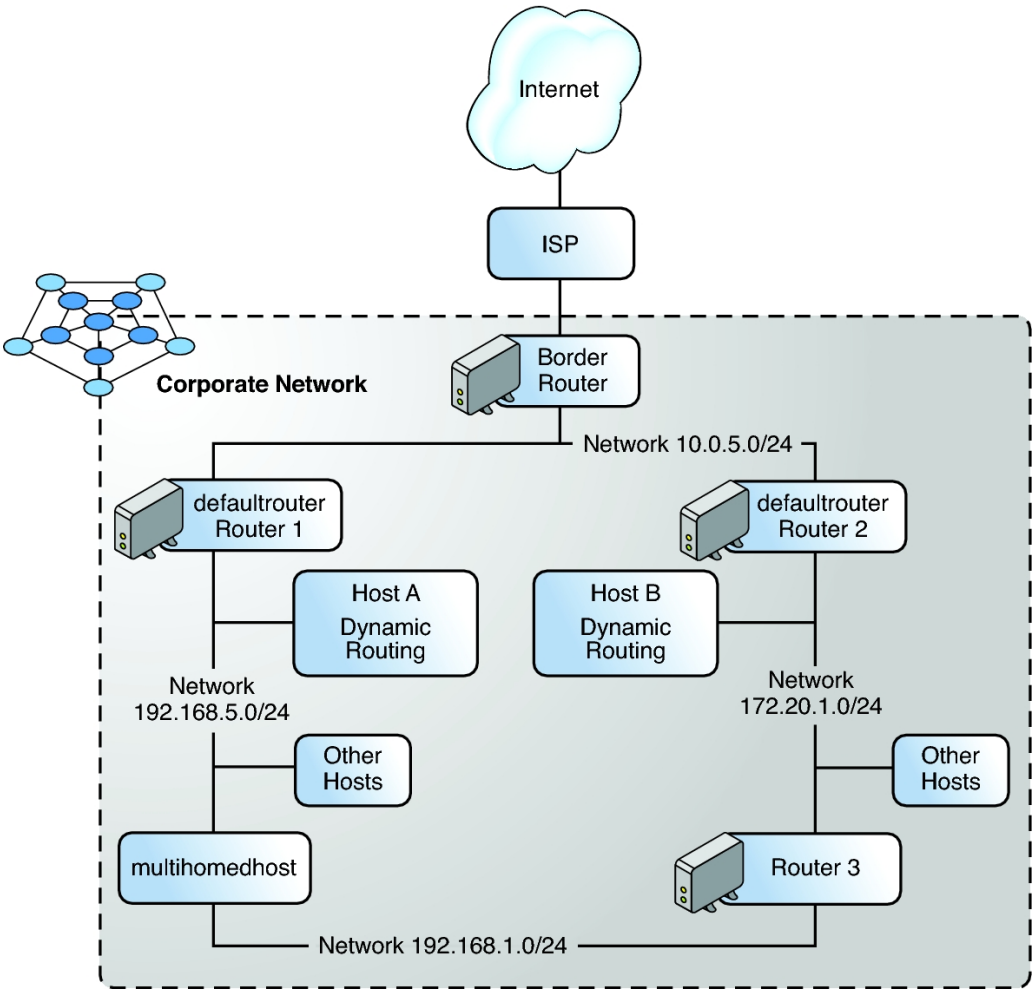
```
# netstat -rn
Routing Table: IPv4
Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
default              172.20.1.10          UG        1    249 ce0
224.0.0.0            172.20.1.10          U         1      0 ce0
10.0.5.0             10.0.5.20            U         1     78 bge0
10.0.5.0             10.0.5.150           U         1    375 bge0
127.0.0.1            127.0.0.1            UH        1     57 lo0

Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use    If
--------------------------- --------------------------- ----- --- ------- -----
::1                         ::1                         UH      2       0 lo0
```

Enabling Routing for Single-Interface Systems

You can configure single-interface systems with either static or dynamic routing. With static routing, the host must rely upon the services of a default router for routing information. Enabling dynamic routing that uses a routing protocol is the easiest way to manage routing on a system.

Sites with multiple routers and networks typically administer their network topology as a single routing domain or an *autonomous system* (AS). The procedures and examples in this section are based on the following figure. In this figure, an AS is divided into three local networks: `10.0.5.0`, `172.20.1.0`, and `192.168.5.0`. The network is comprised of routers and client systems, including the following types of routers: border routers, default routers, and packet-forwarding routers. Client systems include multihomed systems and single-interface systems. For more details about each of these components, see “IPv4 Autonomous System Topology” in “Planning for Network Deployment in Oracle Solaris 11.2 ”.

FIGURE 3-1 Autonomous System With Multiple IPv4 Routers



▼ How to Enable Dynamic Routing on a Single-Interface System

The following procedure assumes that you have already configured the system's IP interface. For more details about planning for routers on a network, see “IPv4 Autonomous System Topology” in “Planning for Network Deployment in Oracle Solaris 11.2 ”.

1. **Become an administrator.**

2. **Configure one of the system's IP interfaces with an IP address for the network to which the system belongs.**

   For instructions, see "How to Configure an IPv4 Interface" on page 46.

3. **Delete all of the persistently defined routes from the system.**

   You perform this step because the presence of any statically defined default routes prevents the system from enabling dynamic routing during a system boot.

   a. **Determine all of the persistently defined default routes as follows:**

      ```
      # route -p show
      ```

   b. **Delete each of the persistently defined routes. For example:**

   ```
   # route -p delete -net default -gateway 172.20.1.10
   ```

4. **Ensure that packet forwarding is disabled.**

   ```
   # routeadm -d ipv4-forwarding -u
   ```

5. **Enable IPv4 routing on the system.**

   ```
   # routeadm -e ipv4-routing -u
   ```

Example 3-8 Running Dynamic Routing on a Single-Interface System

The following example shows how to configure dynamic routing for `hosta`, which is a single-interface system on the network `192.168.5.0`, as illustrated in Figure 3-1. The system uses Router 1 as its default router. The example assumes that you have already configured the system's IP interface.

First, you would log into `hosta` with administrator rights. Then, you would remove all of the persistently defined routes from the system.

```
# route -p show
persistent: route add default 172.20.1.10

# route -p delete default 172.20.1.10
delete net default: gateway 172.20.1.10
delete persistent net default: gateway 172.20.1.10

# routeadm
             Configuration   Current             Current
                    Option   Configuration       System State
---------------------------------------------------------------
              IPv4 routing   disabled            disabled
              IPv6 routing   disabled            disabled
            IPv4 forwarding  disabled            disabled
            IPv6 forwarding  disabled            disabled
```

```
           Routing services   "route:default ripng:default"

Routing daemons:

                     STATE   FMRI
                  disabled   svc:/network/routing/ripng:default
                    online   svc:/network/routing/ndp:default
                  disabled   svc:/network/routing/rdisc:default
                  disabled   svc:/network/routing/legacy-routing:ipv4
                  disabled   svc:/network/routing/legacy-routing:ipv6
                  disabled   svc:/network/routing/route:default

# routeadm -d ipv4-forwarding -u
# routeadm -e ipv4-routing -u
# routeadm
              Configuration   Current             Current
                     Option   Configuration       System State
---------------------------------------------------------------
               IPv4 routing   enabled             enabled
               IPv6 routing   disabled            disabled
             IPv4 forwarding   disabled            disabled
             IPv6 forwarding   disabled            disabled

           Routing services   "route:default ripng:default"

Routing daemons:

                     STATE   FMRI
                  disabled   svc:/network/routing/ripng:default
                    online   svc:/network/routing/ndp:default
                  disabled   svc:/network/routing/rdisc:default
                  disabled   svc:/network/routing/legacy-routing:ipv4
                  disabled   svc:/network/routing/legacy-routing:ipv6
                    online   svc:/network/routing/route:default
```

About IPv6 Routing

IPv6 routing is almost identical to IPv4 routing under CIDR. The only difference is that the addresses are 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. With very straightforward extensions, all of IPv4's routing algorithms, such as Open Shortest Path First (OSPF), RIP, Inter-domain Routing Protocol (IDRP), and the Intermediate System to Intermediate System (IS-IS), can be used to route IPv6.

IPv6 also includes simple routing extensions that support the following powerful new routing capabilities:

- Provider selection that is based on policy, performance, cost, and so on
- Host mobility, route to current location
- Auto-readdressing, route to new address

You obtain the new routing capabilities by creating sequences of IPv6 addresses that use the IPv6 routing option. An IPv6 source uses the routing option to list one or more intermediate nodes or topological group to be visited on the way to a packet's destination. This function is very similar to IPv4's loose source and record route option.

To make address sequences a general function, IPv6 hosts are required (in most instances) to reverse routes in a packet that a host receives. The packet must be successfully authenticated by using the IPv6 authentication header. The packet must contain address sequences in order to return the packet to its originator. This technique forces IPv6 host implementations to support the handling and reversal of source routes. The handling and reversal of source routes enables providers to work with hosts that implement the new IPv6 capabilities such as provider selection and extended addresses.

Configuring Multihomed Hosts

In Oracle Solaris, a system with more than one interface is considered a *multihomed host*. The interfaces of a multihomed host connect to different subnets, either on different physical networks or on the same physical network. For step-by-step instructions on creating a multihomed host, see “How to Create a Multihomed Host” on page 70.

On a system with multiple interfaces that connect to the same subnet, you must configure the interfaces into an IPMP group first. Otherwise, the system cannot be a multihomed host. For more information about IPMP, see Chapter 2, “About IPMP Administration,” in “Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 ”.

A multihomed host does not forward IP packets, but you can configure a multihomed host to run routing protocols. You typically configure the following types of systems as multihomed hosts:

- NFS servers, particularly those servers that function as large data centers, can be attached to more than one network to share files among a large pool of users. These servers do not need to maintain routing tables.
- Database servers can have multiple network interfaces that provide resources to a large pool of users, just like NFS servers.
- Firewall gateways are systems that provide the connection between a company's network and public networks such as the Internet. Administrators set up firewalls as a security measure. When configured as a firewall, the host does not pass packets between the networks that are attached to the host's interfaces. However, the host can still provide standard TCP/IP services such as `ssh` to authorized users.

Note - When multihomed hosts have different types of firewalls on any of their interfaces, take care to avoid unintentional disruption of the host's packets. This problem arises particularly with stateful firewalls. One solution might be to configure stateless firewalls. For more information about firewalls, refer to "Firewall Systems" in "Securing Systems and Attached Devices in Oracle Solaris 11.2 " or the documentation for your third-party firewall.

▼ How to Create a Multihomed Host

1. **Become an administrator.**

2. **Configure each additional network interface that was not configured as part of the installation process.**

   Refer to "How to Configure an IPv4 Interface" on page 46.

3. **If packet forwarding is enabled, disable this service.**

```
# routeadm -p ipv4-forwarding
persistent=enabled default=disabled current=enabled
# routeadm -d ipv4-forwarding -u
# routeadm -p ipv4-forwarding
persistent=disabled default=disabled current=disabled
```

4. **(Optional) Turn on dynamic routing for the multihomed host.**

```
# routeadm -e ipv4-routing -u
# routeadm -p ipv4-routing
persistent=enabled default=enabled current=enabled
```

Example 3-9 Configuring a Multihomed Host

The following example shows how to configure a multihomed host, as illustrated in the figure in "IPv4 Autonomous System Topology" in "Planning for Network Deployment in Oracle Solaris 11.2 ". In this example, the system has the host name `hostc`. This host has two interfaces, which are both connected to network `192.168.5.0`.

To begin, you would display the status of the system's interfaces.

```
# dladm show-link
LINK     CLASS     MTU     STATE   BRIDGE   OVER
net0     phys      1500    up      --       --
net1     phys      1500    up      --       --

# ipadm show-addr
ADDROBJ        TYPE     STATE        ADDR
```

```
lo0/v4         static   ok           127.0.0.1/8
net0/v4        static   ok           192.168.5.82/24
```

The `dladm show-link` command reports that `hostc` has two datalinks. However, only `net0` has been configured with an IP address. To configure `hostc` as a multihomed host, you would configure `net1` with an IP address in the same `192.168.5.0` network. Ensure that the underlying physical NIC of `net1` is physically connected to the network.

```
# ipadm create-ip net1
# ipadm create-addr static -a 192.168.5.85/24 net1
# ipadm show-addr
ADDROBJ        TYPE     STATE        ADDR
lo0/v4         static   ok           127.0.0.1/8
net0/v4        static   ok           192.168.5.82/24
net1/v4        static   ok           192.168.5.85/24
```

Next, you would add the `net1` interface to the `/etc/hosts` file as follows:

```
# vi /etc/inet/hosts
127.0.0.1          localhost
192.168.5.82       hostc    #primary network interface for host3
192.168.5.85         hostc-2  #second interface
```

Then, you would turn off packet forwarding if this service is running on the `hostc` as follows:

```
# routeadm -p ipv4-forwarding
persistent=enabled default=disabled current=enabled

# routeadm
             Configuration   Current             Current
                    Option   Configuration       System State
---------------------------------------------------------------
               IPv4 routing   enabled             enabled
               IPv6 routing   disabled            disabled
            IPv4 forwarding   disabled            disabled
            IPv6 forwarding   disabled            disabled

          Routing services   "route:default ripng:default"

Routing daemons:

                     STATE   FMRI
                  disabled   svc:/network/routing/ripng:default
                    online   svc:/network/routing/ndp:default
                  disabled   svc:/network/routing/rdisc:default
                  disabled   svc:/network/routing/legacy-routing:ipv4
                  disabled   svc:/network/routing/legacy-routing:ipv6
                    online   svc:/network/routing/route:default
```

The `routeadm` command reports that dynamic routing through the `in.routed` daemon is currently enabled.

Implementing Symmetric Routing on Multihomed Hosts

By default, a system with multiple interfaces (also called a *multihomed host*) routes its network traffic based on the longest matching route to the traffic's destination in the routing table. When multiple routes of equal length to the destination exist, Oracle Solaris applies Equal-Cost Multi-Path (ECMP) algorithms to spread the traffic across those routes.

Spreading the traffic in this manner is not always ideal. For example, an IP packet might be sent through an interface on a multihomed host that is not on the same subnet as the IP source address in the packet. Further, if the outgoing packet is in response to a certain incoming request, such as an ICMP echo request, the request and the response might not traverse the same interface. This type of traffic routing configuration is called *asymmetric routing*. If your Internet service provider (ISP) is implementing *ingress filtering*, as described in RFC 3704 (http://www.rfc-editor.org/rfc/bcp/bcp84.txt), an asymmetric routing configuration might cause an outgoing packet to be dropped by the ISP.

RFC 3704 intends to limit denial-of-service (DoS) attacks across the Internet. To comply with this intent, your network must be configured for symmetric routing. The IP `hostmodel` property enables you to meet this requirement. This property controls the behavior of IP packets that are received or transmitted through a multihomed host.

The `hostmodel` property can have one of three possible values:

`strong` — Corresponds to the strong end system (ES) model as defined in RFC 1122. This value implements symmetric routing.

`weak` — Corresponds to the weak ES model as defined in RFC 1122. With this value, a multihomed host uses asymmetric routing.

`src-priority` — Configures packet routing by using preferred routes. If multiple destination routes exist in the routing table, then the preferred routes are those that use interfaces on which the IP source address of an outgoing packet is configured. If no such routes exist, then the outgoing packet will use the longest matching route to the packet's IP destination.

For example, you would implement symmetric routing of IP packets on a multihomed host as follows:

```
# ipadm set-prop -p hostmodel=strong ipv4
# ipadm set-prop -p hostmodel=strong ipv6
# ipadm show-prop -p hostmodel ip
PROTO  PROPERTY   PERM  CURRENT  PERSISTENT  DEFAULT   POSSIBLE
ipv6   hostmodel  rw    strong   --          weak      strong,
src-priority,
weak
ipv4   hostmodel  rw    strong   --          weak      strong,
src-priority,
```

`weak`

Customizing IP Interface Properties and Addresses

There are three `ipadm` subcommands that are used to manage IP interface properties:

- `show-ifprop -p` *property interface* – Displays the properties of an IP interface and its current values. If you do not use the `-p` *property* option, then all of the properties of the IP interface are listed. If you do not specify an IP interface, then all of the properties of all of the IP interfaces are listed.
- `set-ifprop -p` *property=value interface* – Assigns a value to the IP interface's property.
- `reset-ifprop -p` *property interface* – Resets the specific property to its default values.

Like datalinks, IP interfaces also have properties that you can customize for your specific network environment. For each interface, two sets of properties exist, one set for IPv4 and the another set for IPv6.

Setting the MTU Property

Some properties, including the MTU property, are common to both datalinks and IP interfaces. Thus, you can have one MTU value for a datalink and a different MTU value for the interface that is configured over that link. Further, you can have different MTU values that apply to the IPv4 and IPv6 packets that traverse that IP interface.

When setting MTU properties for an IP interface, keep the following key points in mind:

- The value of the MTU setting of an IP interface cannot be larger than the value of the MTU setting of a datalink. In such cases, the `ipadm` command displays an error message.
- If an IP interface's MTU value is different than a datalink's MTU value, IP packets are limited to the MTU value of the IP interface. For example, if a datalink has an MTU value of 9000 bytes and an IP interface as an MTU value of 1500 bytes, IP packets are limited to 1500 bytes. However, other Layer 3 protocols that are using the underlying Layer 2 protocol can send packets up to 9000 bytes.

  For instructions on customizing datalink properties, including information about how the MTU setting of a datalink impacts the MTU setting of an IP interface, see “Customizing Datalink Properties” on page 32.

Enabling Packet Forwarding

On a network, a host can receive data packets that are destined for another host system. By enabling packet forwarding on the receiving local system, that system can forward the data

packet to the destination host. This process is referred to as *IP forwarding* and is disabled by default in Oracle Solaris.

Packet forwarding is managed by a property that can be set on both IP interfaces and of the TCP/IP protocol. If you want to be selective about how packets are forwarded, you can enable packet forwarding on the IP interface. For example, you might have a system that has multiple NICs, where some NICs are connected to the external network, while other NICs are connected to a private network. You would therefore enable packet forwarding only on some of the interfaces, rather than on all of the interfaces.

You can also enable packet forwarding globally on the system by setting the property of the TCP/IP protocol. See "Enabling Packet Forwarding Globally" in "Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 " for more information.

Note - The `forwarding` property of either IP interfaces or protocols is not mutually exclusive. You can set the property for the interface and the protocol at the same time. For example, you could enable packet forwarding globally on the protocol, and then customize packet forwarding for each IP interface on the system. Thus, although enabled globally, packet forwarding can still be selective for the system.

For example, you would enable packet forwarding on the IP interface as follows:

```
# ipadm set-ifprop -p forwarding=on -m protocol-version interface
```

where *protocol-version* is either IPv4 or IPv6. You must type the command separately for IPv4 and IPv6 packets.

The following example shows how you might enable only IPv4 packet forwarding on your system:

```
# ipadm show-ifprop -p forwarding net0
IFNAME   PROPERTY     PROTO   PERM   CURRENT   PERSISTENT   DEFAULT   POSSIBLE
net0     forwarding   ipv4    rw     off       off          off       on,off
net0     forwarding   ipv6    rw     off       --           off       on,off

# ipadm set-ifprop -p forwarding=on -m ipv4 net0
# ipadm show-ifprop net0
IFNAME   PROPERTY         PROTO  PERM  CURRENT  PERSISTENT  DEFAULT   POSSIBLE
...
net0     forwarding       ipv4   rw    on       on          off       on,off
...
```

Customizing IP Address Properties

The `ipadm` command enables you to manage IP address-specific properties.

You can customize IP address properties to manage the following network configuration parameters:

- Netmask length
- Whether an IP address can be used as a source address for outbound packets
- Whether the address belongs to a global or non–global zone
- Whether the address is a private address

Use the following `ipadm` subcommands when working with IP address properties:

- `show-addrprop -p` *property addrobj* – Displays address properties, depending on the options that you use.

  To display the properties of all of the IP addresses, do not specify a property or an address object. To display the values of a single property for all of the IP addresses, specify just that property. To display all of the properties of a specific address object, specify just the address object.
- `set-addrprop -p` *property=value addrobj* – Assigns values to address properties. Note that you can only set one address property at a time.
- `reset-addrprop -p` *property addrobj* – Restores any default values to the address property.

Note - If you want to change the IP address of a specific interface, do not use the `set-addressprop` subcommand. Instead, delete the address object and create a new one with the new IP address. See “Removing or Modifying an IP Interface Configuration” on page 77.

As an example, suppose you want to change the netmask of an IP address. The IP address is configured on the IP interface `net3`and is identified by the address object name `net3/v4`. The following examples show how to revise the netmask:

```
# ipadm show-addr
ADDROBJ     TYPE       STATE     ADDR
lo0/?       static     ok        127.0.0.1/8
net3/v4     static     ok        192.168.84.3/24

# ipadm show-addrprop -p prefixlen net3/v4
ADDROBJ  PROPERTY   PERM  CURRENT  PERSISTENT  DEFAULT   POSSIBLE
net3/v4  prefixlen  rw    24       24          24        1-30,32

# ipadm set-addrprop -p prefixlen=8 net3/v4
# ipadm show-addrprop -p prefixlen net3/v4
ADDROBJ  PROPERTY   PERM  CURRENT  PERSISTENT  DEFAULT   POSSIBLE
net3/v4  prefixlen  rw    8        24          24        1-30,32
```

Disabling, Removing, and Modifying IP Interface Configuration

This section contains the following topics:

Removing an IP Interface Configuration

Use the `delete-ip` subcommand to remove a configured IP interface. This command is particularly important when you are performing certain datalink configuration tasks. For example, renaming a datalink fails if there are IP interfaces configured over that datalink. Before attempting to rename the datalink, you would need to use the `ipadm delete-ip` command to remove the existing IP configuration. The following example shows the commands that you would use to perform this task:

```
# ipadm delete-ip interface
# dladm rename-link old-name new-name
# ipadm create-ip interface
# ipadm create-address parameters
```

See also "Renaming a Datalink" on page 32 for additional information. To reconfigure an IP interface after a datalink has been renamed, see "How to Configure an IPv4 Interface" on page 46.

Disabling an IP Interface Configuration

By default, an IP interface is plumbed and becomes part of the active configuration when you create the interface by using the `ipadm create-ip` command. The interface is flagged as `UP` when the first address is created on the interface.

To remove an interface from active configuration without destroying the configuration, use the `disable-if` subcommand as follows. This subcommand unplumbs the interface in the kernel.

```
# ipadm disable-if -t interface
```

You would make an IP interface operational and its flag displayed as `UP` as follows:

```
# ipadm enable-if -t interface
```

Tip - To display the current status of IP interfaces, see "Obtaining Information About IP Interfaces" on page 79.

Removing or Modifying an IP Interface Configuration

The `ipadm delete-addr` command deletes a specific address configuration from an IP interface. This command is useful when you simply want to remove an IP address from the system or as part of changing an IP address that is configured on an interface. If you want to change the IP address that is configured on an interface, then you must first remove the original address configuration before assigning a new address configuration. See "How to Modify an Existing IP Address" on page 77.

For instructions on creating an IP address for an interface, see "How to Configure an IPv4 Interface" on page 46.

Note - An interface can have multiple IP addresses. Each address is identified by an address object. To ensure that you are removing the correct address, you must know the address object. Use the `ipadm show-addr` command to display the IP addresses that are configured on an interface. For an explanation of the address object, see "How to Configure an IPv4 Interface" on page 46. For more information about displaying IP addresses, see "Obtaining Information About IP Addresses" on page 81

▼ How to Modify an Existing IP Address

The following procedure describes the steps for reconfiguring a system's existing IP address.

1. **Become an administrator**

2. **Delete the address object that represents the IP address that you want to reconfigure.**

   ```
   # ipadm delete-addr addrobj
   ```

3. **Assign a new IP address by using the same address object name.**

   ```
   # ipadm create-addr -a IP-address addrobj
   ```

   To add another interface to the system, see How to Configure an IPv4 Interface.

4. **(Optional) If necessary, modify the system's host name as follows:**

   ```
   # hostname new-hostname
   ```

5. **(Optional) If the subnet mask has changed, modify the subnet entries.**

6. **(Optional) If the subnet address has changed, change the IP address of the default router.**

   See "Creating Persistent (Static) Routes" on page 62 for instructions.

7. **Reboot the system for the changes to take effect.**

Monitoring IP Interfaces and Addresses

Use the `ipadm` command to monitor and obtain information about IP interfaces and their properties. By itself, the command displays general information about IP interfaces on the system. However, you can also use various subcommands to restrict the information that you want to display by using the following command syntax:

`ipadm show-*` *other-arguments interface*

- To obtain only interface information, use the `show-if` subcommand.
- To obtain only address information, use the `show-addr` subcommand.
- To obtain information about interface properties, use the `show-ifprop` subcommand.
- To obtain information about address properties, use the `show-addrprop` subcommand.

For an explanation of all of the fields that are displayed by the `ipadm show-*` commands, see the `ipadm`(1M) man page.

Obtaining General Information About IP Interfaces

The `ipadm` command provides a comprehensive picture of the system's interfaces. Using the command without accompanying subcommands displays default information about all of the system's IP interfaces. For example:

```
# ipadm
NAME         CLASS/TYPE STATE     UNDER  ADDR
lo0          loopback   ok        --     --
lo0/v4       static     ok        --     127.0.0.1/8
lo0/v6       static     ok        --     ::1/128
net0         ip         ok        --     --
net0/v4      static     ok        --     10.132.146.233/23
net0/v4      dhcp       ok        --     10.132.146.234/23
ipmp0        ipmp       degraded  --     --
ipmp0/v6     static     ok        --     2001:db8:1:2::4c08/128
net1         ip         failed    ipmp0  --
net1/v6      addrconf   ok        --     fe80::124:4fff:fe58:1831/10
net2         ip         ok        ipmp0  --
```

```
net2/v6      addrconf   ok        --     fe80::214:4fff:fe58:1832/10
iptun0       ip         ok        --     --
iptun0/v4    static     ok        --     172.16.111.5->172.16.223.75
iptun0/v6    static     ok        --     fe80::10:5->fe80::223:75
iptun0/v6a   static     ok        --     2001:db8:1a0:7::10:5->2001:db8:7a82:64::223:75
```

The previous output displays the following information:

- IP interfaces.
- Class of each interface.
- State of each interface.
- Status of the interface: either a “stand alone” IP interface or an underlying interface for another type of interface configuration. In the example, `net1` and `net2` are underlying interfaces of `ipmp0`, as indicated in the `UNDER` column.
- Address objects that are associated with the interface. Address objects identify a specific IP address. These address objects are listed and indented under the `NAME` heading to distinguish them from interface names.
- Type of IP address, which is indented under the `CLASS/TYPE` heading and which can be `static`, `dhcp` and so on.
- Actual addresses listed under the `ADDRESS` column.

Obtaining Information About IP Interfaces

For information about IP interfaces, use the `ipadm show-if` *interface* command. If you do not specify an interface, the information covers all of the interfaces on the system.

The fields in the command output refer to the following information:

`IFNAME` — Refers to the interface whose information is being displayed.

`CLASS` — Refers to the class of interface, which can be one of four:

- `ip` refers to an IP interface
- `ipmp` refers to an IPMP interface
- `vni` refers to a virtual interface
- `loopback` refers to a loopback interface, which is automatically created. Except for the loopback interface, you can manually create the remaining 3 interface classes.

`STATE` — Refers to the status of the interface, which can be one of the following: `ok`, `offline`, `failed`, `down`, or `disabled`.

The status `failed` applies to IPMP groups and can refer to a datalink or an IP interface that is down and cannot host traffic. If the IP interface

belongs to an IPMP group, then the IPMP interface can continue to receive and send traffic by using other active IP interfaces in the group.

The status `down` refers to an IP interface that is switched offline by the administrator.

The status `disable` refers to the IP interface that is unplumbed by using the `ipadm disable-if` command.

`ACTIVE` Indicates whether the interface is being used to host traffic, and is set to either `yes` or `no`.

`OVER` Applies only to the IPMP class of interfaces and refers to the underlying interfaces that constitute the IPMP interface or group.

The following is an example of the information that the command displays:

```
# ipadm show-if
IFNAME       CLASS        STATE     ACTIVE     OVER
lo0          loopback     ok        yes        --
net0         ip           ok        yes        --
net1         ip           ok        yes        --
tun0         ip           ok        yes        --
```

Obtaining Information About IP Interface Properties

Use the `ipadm show-ifprop` *interface* command to obtain information about the properties of IP interfaces. If you do not specify a property or an interface, then information about all of the properties of all of the IP interfaces on the system is displayed.

The fields in the command output refer to the following:

`IFNAME` Refers to the IP interface whose information is being displayed.

`PROPERTY` Refers to a property of the interface. An interface can have several properties.

`PROTO` Refers to the protocol to which the property applies, which can be either IPv4 or IPv6.

`PERM` Refers to the allowed permissions of a given property, which can be read only, write only, or both.

`CURRENT` Indicates the current value of the property in the active configuration.

PERSISTENT	Refers to the value of the property that is reapplied when the system is rebooted.
DEFAULT	Indicates the default value of the specified property.
POSSIBLE	Refers to a list of values that can be assigned to the specified property. For numeric values, a range of acceptable values is displayed.

Note - If any field value is unknown, such as when an interface does not support the property whose information is being requested, the value is displayed as a question mark (?).

The following example shows the type of information that the `show-ifprop` subcommand displays:

```
# ipadm show-ifprop -p mtu net1
IFNAME  PROPERTY  PROTO  PERM  CURRENT  PERSISTENT  DEFAULT  POSSIBLE
net1    mtu       ipv4   rw    1500     --          1500     68-1500
net1    mtu       ipv6   rw    1500     --          1500     1280-1500
```

Obtaining Information About IP Addresses

For information about IP addresses, use the `ipadm show-addr` *interface* command. If you do not specify an interface, then the information about all of the IP addresses that are on the system is displayed.

The fields in the command output refer to the following:

ADDROBJ	Specifies the address object whose IP address is being listed.
TYPE	Indicates whether the IP address is `static`, `dhcp`, or `addrconf`. The `addrconf` value indicates that the address was obtained by using stateless or stateful address configuration.
STATE	Describes the status of the address object in the active configuration. For a full list of these values, see the `ipadm`(1M) man page.
ADDR	Specifies the IP address that is configured over the interface. The address can be IPv4 or IPv6. A tunnel interface displays both local and remote addresses. For more information about tunnels, see Chapter 5, “Administering IP Tunnels,” in “Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2 ”.

The following is an example of the information that the `show-addr` subcommand provides:

```
# ipadm show-addr
ADDROBJ           TYPE       STATE     ADDR
lo0/v4            static     ok        127.0.0.1/8
net0/v4           static     ok        192.168.84.3/24
tun0/v4           static     ok        172.16.134.1-->172.16.134.2
```

If you specify an interface with the command and the interface has multiple addresses, information that is similar to the following is displayed:

```
# ipadm show-addr net0
ADDROBJ           TYPE       STATE     ADDR
net0/v4           static     ok        192.168.84.3/24
net0/v4a          static     ok        10.0.1.1/24
net0/v4bc         static     ok        172.16.10.1
```

An address object that is displayed as *interface*/*?* indicates that the address was configured on the interface by an application that did not use `libipadm` APIs. Such applications are not under the control of the `ipadm` command, which requires that the address object name use the format *interface*/*user-defined-string*. For examples of assigning IP addresses, see "How to Configure an IPv4 Interface" on page 46.

Obtaining Information About IP Address Properties

For information about IP address properties, use the `ipadm show-addrprop` *addrobj* command. To list all of the properties, omit the *addrobj* option. To list a single property for all of the IP addresses, specify just that property. To display all of the properties of a specific address, specify just the *addrobj* option.

The fields in the command output refer to the following:

`ADDROBJ`	Refers to the address object whose properties are being listed.
`PROPERTY`	Refers to a property of the address object. An address object can have several properties.
`PERM`	Refers to the allowed permissions of a given property, which can be read only, write only, or both.
`CURRENT`	Refers to the actual value of the property in the present configuration.
`PERSISTENT`	Refers to the value of the property that is reapplied when the system is rebooted.

`DEFAULT` Indicates the default value of the specified property.

`POSSIBLE` Refers to a list of values that can be assigned to the specified property. For numeric values, a range of acceptable values is displayed.

The following is an example of the type of information that the `show-addrprop` subcommand displays:

```
# ipadm show-addrprop net1/v4
ADDROBJ   PROPERTY     PERM   CURRENT          PERSISTENT  DEFAULT          POSSIBLE
net1/v4   broadcast    r-     192.168.84.255   --          192.168.84.255   --
net1/v4   deprecated   rw     off              --          off              on,off
net1/v4   prefixlen    rw     24               24          24               1-30,32
net1/v4   private      rw     off              --          off              on,off
net1/v4   transmit     rw     on               --          on               on,off
net1/v4   zone         rw     global           --          global           --
```

♦ ♦ ♦ CHAPTER 4

Administering Naming and Directory Services on an Oracle Solaris Client

This chapter describes how to configure naming services for an Oracle Solaris host client system. For a complete overview of naming and directory services, as well as server-side administration, see "Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ".

For information about troubleshooting naming and directory services configuration, see Chapter 3, "Troubleshooting Naming Services Issues," in "Troubleshooting Network Administration Issues in Oracle Solaris 11.2 ".

This chapter contains the following topics:

Note - The tasks that are described in this chapter apply to both IPv4 and IPv6 networks unless otherwise noted.

What's New in Naming Service Configuration

The following features are new or changed:

- **Naming services and system configuration migration to SMF** – In this release, naming services are managed through the Service Management Facility (SMF). The previous behavior where you modified a certain file to configure naming services, for example, `/etc/nsswitch.conf` and `/etc/resolv.conf`, no longer works. Legacy configuration files are retained in this Oracle Solaris release only for the purpose of compatibility with

previous Oracle Solaris releases. The contents of these files are generated by the SMF service that pertains to the specific naming service.

If no network configuration exists, naming services default to `files only` behavior, rather than `nis files`. The `svc:/system/name-service/cache` SMF service should be enabled at all times. Note also that if you make configuration changes to these services by using SMF commands, the services must be enabled, refreshed, or both for any changes to take effect. See the `svccfg`(1M) and `svcadm`(1M) man pages.

- `resolv.conf` **error-checking capabilities** – Prior to the naming services migration to SMF, errors in the `resolv.conf` file configuration were processed silently and went undetected without producing any warnings. As a result, the `resolv.conf` file did not behave according to how it was configured. Oracle Solaris 11 introduces basic error checking through SMF templates so that error conditions are now properly reported. See the `resolv.conf`(4) man page.
- **Domain Name System (DNS) server setup** – The process for setting up a DNS server has changed. For detailed instructions, see "Administering DNS (Tasks)" in "Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ".

Overview of Naming and Directory Services Configuration

A naming service performs lookups of stored information, such as host names and addresses, user names, passwords, access permissions, and so on. This information is made available so that users can log in to their host, access resources, and be granted permissions. The naming service information can be stored in files, maps, or various forms of database files. These information repositories can be local to the system or located in a central network-based repository or database. Without a central naming service, each host would have to maintain its own copy of this information. If you centralize all data, administration becomes easier. Naming services are fundamental to any computing network.

The following naming and directory services are supported:

- **Domain Name System (DNS)**

  DNS is a hierarchical distributed database that is implemented on a TCP/IP network. It is primarily used to look up IP addresses for Internet host names and host names for IP addresses. The data is distributed across the network and is located by using period-separated names that are read from right to left. DNS is also used to store other Internet-related host information, such as mail exchange routing information, location data, and available services. The hierarchical nature of the service enables the local administration of local domains, while providing international coverage of other domains that are connected to the Internet, an intranet, or both. For more information, see "Description of the DNS Naming Service" in "Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ".

  Two extensions to the DNS protocol are managed by the `svc:network/dns/multicast` service. Multicast DNS (mDNS) implements DNS in a small network where no

conventional DNS server has been installed. DNS Service Discovery (DNS-SD) extends Multicast DNS to also provide simple service discovery (network browsing). See “Description of Multicast DNS and Service Discovery” in “Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ”.

- **Network Information System (NIS)**

  NIS (pronounced "niss" in this guide) was developed independently of DNS. NIS focuses on making network administration more manageable by providing centralized control over a variety of network information. NIS stores information about the network, machine names and addresses, users, and network services. This collection of network information is referred to as the *NIS namespace*. For more information, see “Description of the NIS Naming Service” in “Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ”.

- **Lightweight Directory Access Protocol (LDAP)**

  LDAP is the secure network protocol that is used to access directory servers for distributed naming and other directory services. This standard based protocol supports a hierarchal database structure. The same protocol can be used to provide naming services in both UNIX and multi-platform environments. Oracle Solaris supports LDAP in conjunction with the Oracle Directory Server Enterprise Edition (formerly Sun Java System Directory Server), as well as other LDAP directory servers. For more information, see “Description of the LDAP Naming Services” in “Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ”.

For a complete overview of naming services support (server-side and client-side) in Oracle Solaris, see Chapter 1, “About Naming and Directory Services,” in “Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ” and “Working With Oracle Solaris 11.2 Directory and Naming Services: LDAP ”.

About the `name-service/switch` SMF Service

The `name-service/switch` SMF service is a configurable selection service that enables you to specify which name information service or source to use for each type of network information.

The name service switch is used by client applications that call any of the following interfaces:

- `gethostbyname`
- `getpwuid`
- `getpwnam`
- `getaddrinfo`

The `name-service/switch` SMF service defines which naming service or services should be used for each network database. This information was previously stored in the `/etc/nsswitch.conf` file,. Although this file still exists, the configuration settings that the file contains must be modified by changing the appropriate properties in this SMF service.

You can display these properties as follows:

```
$ svccfg -s name-service/switch listprop config
  config                      application
  config/default              astring     files
  config/value_authorization astring     solaris.smf.value.name-service.switch
  config/password             astring     "files ldap"
  config/group                astring     "files ldap"
  config/host                 astring     "files dns"
  config/automount            astring     "files ldap"
```

The `config/default` property specifies which default source or sources are to be searched. If a particular database does not have its own property set, then the default source or sources are used. In the previous example, all of the databases, with the exception of `password`, `group`, `host`, and `automount`, use local files as their source. If a source or sources other than the default are required, a property is created for the specific database. In this example, `password`, `groups`, and `automount` are searched for in local files first and then in LDAP. Host lookups are searched for in local files first and then in DNS.

If you change the naming services that are enabled on your system, you must update the appropriate properties of the `name-service/switch` SMF service to use the correct naming service. For example, suppose that your `name-service/switch` was configured similarly to the previous example, and you then disabled LDAP and enabled NIS instead.

- In this case, you would need to set the following properties of the `name-service/switch` service to use files and NIS:
- `config/password`
- `config/group`
- `config/automount`

You would type the following commands to set these properties correctly:

```
# svccfg -s name-serivce/switch setprop config/password = astring: "'files nis'"
# svccfg -s name-serivce/switch setprop config/group = astring: "'files nis'"
# svccfg -s name-serivce/switch setprop config/automountconfig/password = astring:
 "'files nis'"
# svccfg -s name-service/switch:default refresh
```

For complete details, see Chapter 2, “About the Name Service Switch,” in “Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ” and “Configuring the Name Service Switch” in “Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ”.

Configuring a System for Local Files Mode

When running in local files mode, a system obtains all TCP/IP configuration information from files that are located in a local directory. In network client mode, the configuration information is provided to all of the systems in the network by a remote network configuration server.

Typically, the following servers on the network run in local files mode:

- Network configuration servers
- NFS servers
- Name servers that supply NIS, LDAP, or DNS services
- Mail servers
- Routers

Since clients can run in either network client mode or local files mode, on any given network, you can have a combination of these modes with which different systems are configured.

▼ How to Configure a System for Local Files Mode

1. **Become an administrator.**

2. **Configure the system's IP interfaces with the assigned IP addresses.**

   Refer to How to Configure an IPv4 Interface.

3. **Verify that the host name was set correctly.**

   ```
   # hostname
   ```

   For more information, see the `hostname`(1) man page.

4. **Verify that the entries in the `/etc/inet/hosts` are current.**

   Oracle Solaris creates entries for the primary network interface, the loopback address, and any additional interfaces that are configured during the installation, if applicable. If the entries are not current, add the IP addresses and corresponding names for any network interfaces that were added to the system after the installation.

5. **Specify the system's fully qualified domain as a property of the `nis/domain` SMF service.**

   For example, you would specify `deserts.worldwide.com` as the value for the `domainname` property of the `nis/domain` SMF service as follows:

   ```
   # domainname domainname
   ```

This step makes the change persistent.

6. **Add the routing information.**

Note - If you are using DHCP services, you can skip this step.

For instructions, see “Configuring Routing” on page 60.

7. **Add the netmask information, if applicable.**

Note - If you are using DHCP services, you can skip this step.

---

   a. **Type the network number and the netmask in the `/etc/inet/netmasks` file.**

      To create entries, use the format *network-number netmask*. For example, to specify the Class C network number `192.168.83`, you would type the following information:

      ```
      192.168.83.0    255.255.255.0
      ```

      For CIDR addresses, convert the network prefix to the equivalent dotted decimal representation. For example, type the following information to express the CIDR network prefix `192.168.3.0/22`.

      ```
      192.168.3.0     255.255.252.0
      ```

   b. **Change the lookup source for the netmask in the `name-service/switch` property so that only local files are searched, then refresh the instance.**

      ```
      # svccfg -s name-service/switch setprop config/netmask = astring: "files"
      # svccfg -s name-service/switch:default refresh
      ```

8. **Reboot the system.**

Configuring a DNS Client

DNS has two parts: a service that provides answers and a client that queries the service. In Oracle Solaris, the default DNS service is provided by the Berkeley Internet Name Domain (BIND) from the Internet Systems Consortium (ISC) and its associated `named` server daemon. The DNS client consists of a collection of utilities and libraries.

For additional task-related information, see “Administering DNS (Tasks)” in “Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ”.

How to Enable a DNS Client

1. **Become an administrator.**

2. **List the domains to search and the IP addresses for the DNS name servers, then update the SMF repository. For example:**

```
# svccfg -s network/dns/client
   svc:/network/dns/client> setprop config/search = astring: ("example.com"
 "sales.example.com")
   svc:/network/dns/client> setprop config/nameserver = net_address: (192.168.1.10
 192.168.1.11)
   svc:/network/dns/client> select network/dns/client:default
   svc:/network/dns/client:default> refresh
   svc:/network/dns/client:default> quit
```

   Make sure to refresh the service for the changes to take effect.

3. **Update the name service switch information to use DNS.**

   The first command updates the DNS configuration information in the SMF repository.

```
# svccfg -s system/name-service/switch
   svc:/system/name-service/switch> setprop config/host = astring: "files dns"
   svc:/system/name-service/switch> select system/name-service/switch:default
   svc:/system/name-service/switch:default> refresh
   svc:/system/name-service/switch:default> quit
```

4. **Start the services that are needed to run the DNS client.**

```
# svcadm enable network/dns/client
# svcadm enable system/name-service/switch
```

5. **Verify that the DNS client is enabled by using one or both of the following commands:**

   `# dig` *knownserver.example.com*

   `# getent hosts` *knownserver.example.com*

   The `dig` command when used alone checks that the DNS client is enabled. The `getent hosts` command verifies the `/etc/nsswitch.conf` file's use of the DNS client.

Example 4-1 Setting Multiple DNS Options for a Client Simultaneously

The following example shows how you would set multiple `/etc/resolv.conf` options.

```
# svccg
svc:> select /network/dns/client
svc:/network/dns/client> setprop config/options = "ndots:2 retrans:3 retry:1"
svc:/network/dns/client> listprop config/options
```

```
config/options astring     ndots:2 retrans:3 retry:1
svc:/network/dns/client> exit
# svcadm refresh dns/client
# grep options /etc/resolv.conf
options ndots:2 retrans:3 retry:1
```

Enabling Multicast DNS

For Multicast DNS (mDNS) and DNS Service Discovery to function, mDNS must be deployed on all of the systems that will participate in mDNS. The mDNS service is used to advertise the availability of services that are provided on the system.

Prior to enabling mDNS, make sure that the software package is installed on your system. If necessary, install the package as follows:

```
# pkg install pkg:/service/network/dns/mdns
```

Part of the process of enabling mDNS is to first update the name service switch information. To be able to resolve local hosts, you must change the `config/host` property of the `name-service/switch` SMF service to include `mdns` as a source as follows:

```
#  /usr/sbin/svccfg -s svc:/system/name-service/switch
svc:/system/name-service/switch> setprop config/host = astring: "files dns mdns"
svc:/system/name-service/switch> select system/name-service/switch:default
svc:/system/name-service/switch:default> refresh
svc:/system/name-service/switch> quit
```

Enable the mDNS SMF service as follows:

```
# svcadm enable svc:/network/dns/multicast:default
```

When you enable mDNS by using this method your changes persist through upgrades and reboots. For more information, see the `svcadm`(1M) man page.

Advertising Resources for DNS

You can use the `dns-sd` command to browse and discover services similarly to how you use the `ping` or `traceroute` commands. The `dns-sd` command is primarily used interactively, mainly because its command-line arguments and output format can change over time, which makes invoking the command from a shell script unpredictable and risky. Additionally, the asynchronous nature of DNS service discovery (DNS-SD) does not easily lend itself to script-oriented programming.

For examples, see “Advertising Resources for DNS” in “Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ”. See also the `dns-sd`(1M) man page.

Configuring a NIS Client

NIS is a distributed naming service that is used to identify and locate network objects and resources. NIS provides a uniform storage and retrieval method for network-wide information in a transport-protocol, media-independent fashion. NIS was developed independently of DNS and has a slightly different focus. While DNS focuses on making communication simpler by using machine names instead of numerical IP addresses, NIS focuses on making network administration more manageable by providing centralized control over a variety of network information.

By running NIS, you can distribute administrative databases (maps) among a variety of servers (master and slaves). You can then update those databases from a centralized location in an automatic and reliable fashion, which ensures that all clients share the same naming service information in a consistent manner across the network. For a detailed overview, see Chapter 5, "About the Network Information Service," in "Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ".

There are two SMF services that provide NIS client service on a system. Use the `svcadm` command to enable, disable, restart, and refresh these services. Display the state of the NIS services on a system as follows:

```
# svcs \*nis\*
STATE          STIME    FMRI
online         Oct_09   svc:/network/nis/domain:default
online         Oct_09   svc:/network/nis/client:default
```

For additional task-related information, see Chapter 6, "Setting Up and Configuring Network Information Service," in "Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ".

▼ How to Configure a NIS Client in Broadcast Mode

Broadcast mode is the simplest method for establishing a NIS client. When you start the `nis/client` SMF service, the service runs the `ypbind` command, which searches the local subnet for a NIS server. If a subnet is found, the `ypbind` command binds to it. This search is referred to as broadcasting. If no NIS server exists on the client's local subnet, the `ypbind` command fails to bind and the client machine cannot obtain namespace data from the NIS service. See "How to Configure a NIS Client by Using Specific NIS Servers" on page 94.

1. **Become an administrator.**

2. **Set the NIS domain name.**

   ```
   # domainname example.com
   ```

3. **If needed, make changes to the name service switch.**

   See “Configuring the Name Service Switch” in “Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ”.

4. **Start the NIS client services.**

```
# svcadm enable network/nis/domain
# svcadm enable network/nis/client
```

How to Configure a NIS Client by Using Specific NIS Servers

Before You Begin The following procedure requires that the host names (servers) that are specified in Step 3 can be resolved by DNS. If you are not using DNS, and you type a host name instead of an IP address, make sure to add an appropriate entry for each NIS server to the `/etc/hosts` file on the client. For more information, see the `ypinit`(1M) man page.

1. **Become an administrator.**

2. **Set the NIS domain.**

```
# domainname example.com
# svcadm enable network/nis/domain
```

3. **Run the client configuration script.**

```
# ypinit -c
```

   You are prompted to name the NIS servers from which the client obtains naming service information. You can list the master server and as many slave servers as you want. The servers that you list can be located anywhere in the domain. It is a better practice to first list the servers that are closest (in network terms) to the machine than servers that are located in more distant parts of the network.

4. **Enable the NIS client.**

```
# svcadm enable network/nis/client
```

How to Disable NIS Client Services

1. **Become an administrator.**

2. **Stop the NIS client services as follows:**

```
# svcadm disable network/nis/domain
# svcadm disable network/nis/client
```

Configuring an LDAP Client

For an Oracle Solaris client to use LDAP as a naming service, the following requirements must be met:

- Client's domain name must be served by the LDAP server.
- Name service switch must point to LDAP for the required services.
- Client must be configured with all of the given parameters that define its behavior.
- `ldap_cachemgr` must be running on the client.
- At least one server for which a client is configured must be up and running.

The `ldapclient` command is key to setting up an LDAP client, as this command performs all of the previously listed tasks, except for starting the server.

When using the fixed mode for network configuration, the easiest way to set up LDAP on a client system is to enable the `DefaultFixed` profile and perform persistent network configuration. Then, you can use the `ldapclient` command to complete the LDAP setup, either by using a profile or a manual setup. See the `ldapclient`(1M) man page for more information.

For a detailed overview of LDAP, see “Working With Oracle Solaris 11.2 Directory and Naming Services: LDAP ”.

Importing Naming Services Configuration

The `nscfg` command transfers legacy file configuration for the `name-service` switch components into the SMF repository. When you upgrade to Oracle Solaris 11, the system's naming service configuration is automatically migrated to SMF. However, if necessary, you can manually migrate this configuration to SMF by using the `nscfg` command.

The following command imports the legacy file and then converts and pushes the configuration to SMF:

```
# /usr/sbin/nscfg import -f FMRI
```

Using the `nscfg` command is the simplest way to populate the DNS configuration with information from a previously existing `resolv.conf` file. In the following example, the `nscfg` command reads the information in the `/etc/resolv.conf` file, converts it, then stores the information in the `svc:/network/dns/client` SMF service:

```
# cp resolv.conf /etc/resolv.conf
# /usr/sbin/nscfg import -f dns/client
# svcadm enable dns/client
```

When you change a system's naming service, you need to also modify the name service switch information accordingly and possibly flush any stale information from the name service cache, as shown in the following example:

```
# cp /etc/nsswitch.dns /etc/nsswitch.conf
# /usr/sbin/nscfg import -f name-service/switch
# svcadm refresh name-service/switch
# svcadm refresh name-service/cache
```

For more information, see the `nscfg`(1M) man page.

Resetting SMF Naming Services Configuration

You can reset the configuration properties of an SMF naming service back to the `files only` mode as follows:

```
# /usr/sbin/nscfg unconfig FMRI
# svcadm refresh name-service/switch
```

For example, you would revert the changes that were made to the SMF configuration in “Importing Naming Services Configuration” on page 95 as follows:

```
# svcadm disable dns/client
# /usr/sbin/nscfg unconfig dns/client
# /usr/sbin/nscfg unconfig name-service/switch
# svcadm refresh name-service/switch
# svcadm refresh name-service/cache
```

♦ ♦ ♦ CHAPTER 5

About Administering Profile-Based Network Configuration in Oracle Solaris

This chapter provides an overview of profile-based network configuration, which is primarily used when the system is using the reactive mode. The reactive mode supports various types of profiles for managing network configuration on your Oracle Solaris system. This mode is most often used for notebook PCs and in situations where network conditions change often. In the reactive mode, a network daemon (`nwamd`) monitors the system's state and dynamically adjusts the network configuration if conditions change.

For more information about the fixed mode, which is the mode that is most often used in corporate environments, see Chapter 2, “Administering Datalink Configuration in Oracle Solaris” and Chapter 3, “Configuring and Administering IP Interfaces and Addresses in Oracle Solaris”.

For step-by-step instructions on configuring and administering network profiles, see Chapter 6, “Administering Profile-Based Network Configuration in Oracle Solaris”.

This chapter contains the following topics:

About the Reactive Mode

Reactive mode means the system automatically adapts to any changes in network conditions and then adjusts the system's current network configuration without requiring any manual reconfiguration. For example, if your wired network interface becomes unplugged, if a new wireless network becomes available, or if you change physical locations, the system adapts its network configuration accordingly.

With the primary focus on mobility, the reactive network configuration policy in Oracle Solaris enables the system's network configuration to change dynamically in response to different network events or at your request. This type of network configuration works best for notebook PC use and in situations where network conditions change often. When using the reactive mode, basic Ethernet and WiFi configuration of a system are performed automatically. The system automatically connects to a wired or wireless network at startup and displays notifications about the status of the currently active network connection on the desktop. You can configure reactive profiles with properties that determine the conditions under which a particular profile is enabled. These properties enable the profile's configuration to be applied dynamically to the system by the network management daemon, `nwamd`, as needed.

Reactive network configuration changes are typically triggered by the following events and activities:

- Connecting or disconnecting an Ethernet cable
- Connecting or disconnecting a wireless local area network (WLAN) card
- Booting a system when a wired interface or a wireless interface is available
- Resuming from suspend when a wired interface or a wireless interface is available (if supported)
- Acquiring or losing a DHCP lease

You use two commands to administer network configuration when in the reactive mode: the `netcfg` command for making network configuration changes to profiles and the `netadm` command for displaying information about profiles, as well as enabling and disabling profiles. For a complete description, see the `netcfg`(1M) and `netadm`(1M) man pages. For task-related information, see Chapter 6, “Administering Profile-Based Network Configuration in Oracle Solaris”.

Fixed mode, on the other hand, is the opposite of reactive mode. When you are using the fixed mode, the network daemon instantiates a specific network configuration on the system, but does not automatically adjust that configuration to varying network conditions. For more information about the fixed mode, see “About Network Configuration Modes” on page 16.

For more information about all of the network administration commands that are supported and when to use them, see “Oracle Solaris Network Administration Commands” on page 17.

About Profile-Based Network Configuration

Oracle Solaris provides a predetermined set of system-defined profiles, as well as the capability for creating various types of user-defined reactive profiles with properties and activation conditions that you specify to meet your particular networking needs. You can use profiles to simplify the basic configuration of datalinks and IP addresses on your system, as well as define more complex system-wide network configuration, for example, naming services, IP Filter, and IP Security (IPsec) configurations.

The following profile types are supported:

- **Network configuration profiles (NCPs)** – An NCP is the principal profile type that is used to specify the configuration of network datalinks and IP interfaces. NCPs are configured with property values that specify how the network is configured when that particular NCP is activated on the system. NCPs can be reactive or fixed. You can have multiple reactive NCPs configured, but Oracle Solaris only supports one fixed NCP named `DefaultFixed`.
- **Network configuration units (NCUs)** – The individual configuration information (properties) that defines an NCP are specified within NCUs. An NCU can represent a physical link or an interface and contains properties that specify the configuration for that link or interface.
- **Location profiles** – A Location profile (also referred to as a Location) specifies system-wide network configuration, for example, naming services, domain, IP Filter configuration, and IPsec configuration.
- **External network modifiers (ENMs)** – An ENM is a profile that manages applications that are responsible for creating network configuration that is external to the system's primary network configuration, for example, a VPN application.
- **Known WLANs** – A Known WLAN is a profile that stores information about wireless networks that are discovered by your system.

Profile Type Descriptions

The following is a detailed description of each of profile types that are supported in the Oracle Solaris release.

Description of an NCP

An NCP defines system-specific network configuration, for example datalinks and IP interfaces and addresses. The various NCUs (network configuration units) that are part of each NCP specify how to configure the various network links and interfaces, for example, which interface or interfaces should be brought up, and under what conditions that interface should be brought up, as well as how the IP address for the interface is obtained.

The `Automatic` NCP represents all of the network links and interfaces that are currently in the system. The content of the `Automatic` NCP changes if network devices are added or removed. The `Automatic` NCP provides access to a profile that utilizes DHCP address autoconfiguration, which makes it possible to obtain IP addresses for the system. This NCP also implements a link selection policy that favors wired links over wireless links. If the specification of an alternate IP configuration policy or an alternate link selection policy is required, you would need to create another NCP on your system. You cannot delete the `Automatic` NCP. You can copy this NCP and make changes to the copy. See Example 6-7.

Description of an NCU

NCUs contain the property values that define an NCP. NCUs represent the individual physical links and interfaces that are on a system. The process of configuring a user-defined NCP includes creating NCUs that specify how and under what conditions each link and interface should be configured.

There are two types of NCUs:

- Link NCUs – Represent physical devices (Layer 2 entities in the Open Systems Interconnection (OSI) model)
- Interface NCUs – Represent IP interfaces (Layer 3 entities)

Link NCUs represent the following datalink layer classes:

- Aggregations
- Bridges
- Etherstubs
- Ethernet over IB (EoIB),
- Physical links (Ethernet or WiFi)
- Tunnels
- Virtual eXtensible local areal networks (VXLANs)
- Virtual local area networks (VLANs)
- Virtual network interface cards (VNICs)

Interface NCUs represent the following IP layer classes:

- IP interfaces
- IPMP interfaces
- VNI interfaces

For information about the properties that you can set for the various object types, see the `netcfg`(1M) man page.

Description of a Location Profile

A Location profile (also referred to simply as a *Location*) consists of network configuration information such as naming services and firewall settings that are applied together to specify system-wide network configuration when that Location is active. Because a Location does not necessarily correspond to a physical location, you can set up several Location profiles to meet different networking needs. For example, one Location can be used when you are connected to the company intranet. Another Location can be used when you are connected to the public Internet by using a wireless access point that is located in your office.

By default, there are three Locations that are predefined by the system:

- **DefaultFixed**

  The `DefaultFixed` Location is enabled whenever the `DefaultFixed` NCP is active. The `DefaultFixed` Location cannot be directly modified by the using the `netcfg` command. When this Location is enabled (as part of enabling the `DefaultFixed` NCP), the relevant Service Management Facility (SMF) properties are updated to reflect the settings of the Location. When the system is shut down or another Location is enabled, the relevant SMF properties are saved as part of the `DefaultFixed` Location's configuration.

- **Automatic**

  The `Automatic` Location is activated if there are networks available but no other Location supersedes it. You can modify the `Automatic` Location by using the `netcfg` command.

Note - The `Automatic` Location should not be confused with the `Automatic` NCP. The `Automatic` Location defines system-wide network properties after the initial network configuration of a system takes place. The `Automatic` NCP specifies link and interface network configuration on a system.

- **NoNet**

  The `NoNet` Location has very specific activation conditions. This Location is applied by the system to a stand-alone system when no local interfaces have an assigned IP address. You can modify the `NoNet` Location by using the `netcfg` command.

User-defined Locations are identical to system-defined Locations, with the exception that a user-defined Location is configured with custom values that you specify, while system-defined Locations have preset values.

Description of an ENM

ENMs enable you to specify when applications or scripts should perform network configuration that is external to the configuration that is specified in the NCP and Location profiles. ENMs can also be defined as services or applications that directly modify your network configuration when they are enabled or disabled. You can specify the conditions under which an ENM should be enabled or disabled. You can also enable or disable an ENM manually. Unlike an NCP or a Location profile, where only one of each profile type can be active on the system at any given time, multiple ENMs can potentially be active on the system at the same time. The ENMs that are active on a system at any given time are not necessarily dependent on the NCP or Location profile that is also enabled on the system at the same time.

Although there are several external applications and services for which you can create an ENM, the obvious example is the VPN application. After you install and configure VPN on your system, you can create an ENM that automatically activates and deactivates VPN under the conditions that you specified.

Note - The reactive network configuration mode cannot automatically detect external applications that are capable of directly modifying the network configuration on a system. To manage the activation or deactivation of a VPN application, or any external application or service, you must first install the application, then create an ENM for it by using either the command-line interface (CLI) or the network administration GUI.

Persistent information about any network configuration that is performed by an ENM is not stored or tracked in exactly the same way that information about an NCP or a Location profile is stored. However, the system is capable of noting an externally initiated network configuration, and then based on any configuration changes that are made to the system by an ENM, reevaluating which Location should be active, and subsequently activating that Location. An example would be switching to a Location that is activated conditionally when a certain IP address is in use. If the `svc:/network/physical:default` service is restarted at any time, the network configuration that is specified by the active NCP is reinstated. ENMs are restarted as well, possibly tearing down and recreating network configuration in the process.

Description of a Known WLAN

Known WLANs are profiles that are used to manage wireless networks that are known to the system. A global list of these known wireless networks is then maintained by the system. This information is used to determine the order in which attempts to connect to available wireless networks are made. If a wireless network that exists in the Known WLAN list is available, the system automatically connects to that network. If two or more known wireless networks are available, the system attempts to connect to the wireless network with the highest priority (lowest number). Any new wireless network that you connect to is automatically added to the top of the Known WLAN list and becomes the current highest priority wireless network.

The default behavior is to prefer more recently connected WLANs over WLANs that you connected to previously. At no time can any Known WLAN share the same priority. If a new WLAN is added to the list with the same priority value as an existing WLAN, the existing entry is shifted to a lower priority value. Subsequently, the priority value of every other WLAN in the list is dynamically shifted to a lower priority value.

One key name can also be associated with a Known WLAN. A *Key name* enables you to create your own key by using the `dladm create-secobj` command. You can then associate this key with WLANs by adding the secure object names to the WLAN `keyname` property. For more information, see the `dladm`(1M) man page.

For more information managing WLANs from the command line, see “Administering Known WLANs in Reactive Mode” on page 147.

System-Defined and User-Defined Profiles

The `Automatic` NCP is a system-defined profile that is made up of one link NCU and one interface NCU for each physical link that is present in the system. The NCU activation policy in this NCP is to prefer connected, wired links over wireless links and to plumb both IPv4 and IPv6 on each enabled link. DHCP is used to obtain IPv4 addresses. Stateless Autoconf and DHCP are used to obtain IPv6 addresses. The `Automatic` NCP changes dynamically when new links are inserted or removed from the system. All NCUs that correspond to the inserted or removed link are also added or removed at the same time. The profile is updated automatically by the `nwamd` daemon.

When active, the `Automatic` NCP implements the following basic policy:

- Configure all available (connected) Ethernet interfaces by using DHCP.
- If no Ethernet interfaces are connected, or if none can obtain an IP address, enable one wireless interface, automatically connecting to the best available WLAN from the Known WLAN list. See "Description of a Known WLAN" on page 102.
- Until at least one IP4 address has been obtained, keep the `NoNet` Location active. See "Description of a Location Profile" on page 100. This Location provides a strict set of IP Filter rules that only pass data that is relevant to IP address acquisition (DHCP and IPv6 autoconf messages). All of the properties of the `NoNet` Location, with the exception of the activation conditions, can be modified.
- When at least one IP address has been assigned to one of the system's interfaces, activate the `Automatic` Location. This Location has no IP Filter or IPsec rules. The Location applies the domain name system (DNS) configuration data that is obtained from the DHCP server. As with the `NoNet` Location, all of the properties of the `Automatic` Location, with the exception of its activation conditions, can be modified.

You can optionally configure user-defined NCPs. You must explicitly add and remove NCUs from the specified NCP. You can also create NCUs that do not correlate to any link that is currently present in the system. In addition, you can determine the policy for the user-defined NCP. For example, you can allow multiple links and interfaces to be enabled on the system at a given time, as well as specify different dependency relationships between NCUs and static IP addresses.

Guidelines for Using Profile-Based Network Configuration

Profile-based network configuration adheres to the following guidelines:

- Only one network configuration profile (NCP) and one Location profile can be active at any given time on a system. All other existing NCPs on the system are non-operational.
- The active NCP can be either reactive or fixed (`DefaultFixed`). With a reactive NCP, the system monitors the network configuration to adapt to changes in the system's network

environment. With the `DefaultFixed` NCP (the system's only fixed profile), the network configuration is instantiated but not monitored.

- The values of the different properties of an NCP constitute a policy that governs how the profile manages the network configuration of the system.
- Any changes to an NCP's properties are immediately implemented as new property values, which then become part of the profile's policy for managing the network configuration whenever that profile is active.
- If your system is using the reactive mode, then the active NCP that manages its network configuration is either the `Automatic` NCP or a user-defined reactive NCP that you create. When a reactive NCP is active, you administer network configuration with the `netcfg` and `netadm` commands.

If your system is using the fixed mode, then the active NCP that manages its network configuration is always `DefaultFixed`. When this NCP is active, you administer network configuration by using the `dladm` and `ipadm` commands. See Chapter 2, “Administering Datalink Configuration in Oracle Solaris” and Chapter 3, “Configuring and Administering IP Interfaces and Addresses in Oracle Solaris” for more information.

Profile Activation Policy

Activation modes for profiles are either manual, automatic, or conditional. When a reactive profile is active, changes in the network environment cause the system to reevaluate the network configuration and then make a “best guess” at which reactive NCP and Location to activate, based on current conditions. Changes in network conditions include plugging or unplugging an Ethernet cable, obtaining or losing a DHCP lease, and detecting a new wireless network. At all times, there must be at one NCP and Location that is active on the system.

The reactive network configuration mode enables you to specify the activation policy for reactive NCPs. This policy determines when to enable NCUs. Each Location profile also contains properties that define its activation criteria.

NCUs, Locations and ENMs both have an `activation-mode` property. The allowable values for each of these profile types differs. Also how the `activation-mode` property is validated differs for each profile type, as well as the conditions under which each profile type is enabled.

Note - The `activation-mode` property for an NCU can either be set to `manual` or `prioritized`. The `activation-mode` property for a Location profile can be set to `manual`, `conditional-any`, `conditional-all`, or `system`.

The NCP activation policy is enforced through the use of properties and conditions that can be specified for each NCU. Examples of policy that you might specify include: “prefer

wired connections over wireless connections" or "activate one interface at a time." How and when NCPs are enabled is defined in the properties that are set for each NCU type. For more information about activation conditions, see the `netcfg`(1M) man page.

Note - An interface NCU is associated with an underlying link NCU. Each interface NCU becomes active when its associated link NCU is enabled. The dependency on the underlying link NCU can never be removed. If you enable an interface NCU without enabling its associated link NCU, the interface NCU does not actually become active until the underlying link NCU for that interface is enabled.

Profile Activation Modes

User-defined NCUs, Location profiles, and ENMs all have `activation-mode` properties. The `activation-mode` property is set when you create or modify a profile by using the `netcfg` command. NCPs do not have an `activation-mode` property. All NCPs are manually enabled.

The following table describes the possible values for the `activation-mode` property for the different profile types.

TABLE 5-1 `activation-mode` Property Values

Profile Type	`activation-mode` Value
NCU	`manual` or `prioritized`
Location	`manual`, `conditional-any`, `conditional-all`, or `system`
ENM	`manual`, `conditional-any`, or `conditional-all`

For more information about enabling and disabling profiles, see "Enabling and Disabling Profiles" on page 109.

Security Requirements for Using Profile-Based Network Configuration

The `netcfgd` daemon controls the repository that stores all of the network configuration information. The `netcfg` command, the network administration GUI, and the `nwamd` daemon each send requests to the `netcfgd` daemon to access the repository..

The current network configuration implementation uses the following authorizations to perform specific tasks:

- `solaris.network.autoconf.read` – Enables the reading of network profile data, which is verified by the `netcfgd` daemon.
- `solaris.network.autoconf.write` – Enables the writing of network profile data, which is verified by the `netcfgd` daemon.
- `solaris.network.autoconf.select` – Enables new configuration data to be applied, which is verified by the `nwamd` daemon.
- `solaris.network.autconf.wlan` – Enables the writing of Known WLAN configuration data.

These authorizations are registered in the `auth_attr` database. See the `auth_attr`(4) man page.

The `solaris.network.autoconf.read` authorization is included in the Basic Solaris User rights profile, which is assigned to all users by default. Anyone with this authorization is therefore able to view the current state of the network and the contents of all network profiles.

Two additional rights profiles are provided: Network Autoconf User and Network Autoconf Admin. The Network Autoconf User profile has `read`, `select`, and `wlan` authorizations. The Network Autoconf Admin profile adds the `write` authorization. The Network Autoconf User profile is assigned to the Console User profile. By default, anyone who is logged into the console can view, enable, and disable profiles. Because the Console User profile is not assigned the `solaris.network.autoconf.write` authorization, a user with this authorization cannot create or modify NCPs, NCUs, locations, or ENMs. However, the Console User profile can view, create, and modify WLANs.

The `netcfg` and `netadm` commands can be used to view network profiles by anyone who has the Basic Solaris User rights profile. This profile is assigned to all users by default.

The `netadm` command can also be used to enable profiles by any user who has the Network Autoconf User or Console User profile. The Console User profile is automatically assigned to the user who is logged into the system from `/dev/console`.

To modify network profiles by using the `netcfg` command, you need the `solaris.network.autoconf.write` authorization or the Network Autoconf Admin profile.

For example, you would determine the privileges that are associated with the Console User rights profile as follows:

```
$ profiles -p "Console User" info
 name=Console User
 desc=Manage System as the Console User
 auths=solaris.system.shutdown,solaris.device.cdrw,solaris.devinde.mount.removable,
 solaris.smf.manage.vbiosd,solaris.smf.value.vbiosd
 profiles=Suspend To RAM,Suspend To Disk,Brightness,CPU
```

```
Power Management,Network Autoconf User,Desktop Removable Media User
help=RtConsUser.html
```

How Profile-Based Network Configuration Works With Other Oracle Solaris Features

Reactive network configuration mode works with other Oracle Solaris networking technologies as follows:

- Virtual machines: Oracle VM Server for SPARC (formerly Logical Domains) and Oracle VM VirtualBox

  Reactive profiles work with both Oracle Solaris hosts and guests. However, reactive network configuration mode manages only the interfaces that belong to the specified virtual machines without interfering with other virtual machines on the system.
- Oracle Solaris Zones and stack instances

  Reactive profiles work in global zones or in exclusive stack non-global zones. However you cannot configure reactive profiles for a shared stack zone, as the network configuration for shared stack zones is always managed in the global zone.
- Dynamic Reconfiguration (DR)

  The system's network configuration supports the dynamic reconfiguration (DR) feature and hot-plug features for systems that have these capabilities. You can use these features to add or remove a device, regardless of which NCP is currently active (a reactive NCP or the `DefaultFixed` NCP). However, the behavior of the system varies depending on the type of NCP that is currently active.

  When the `Automatic` NCP or another reactive NCP is active and a device is plugged in, the NCP automatically creates IP configuration for the newly added device. If the device is removed from the system while a reactive profile is currently active, the IP interface for the device is unconfigured. When the `DefaultFixed` NCP is the active on the system, you must explicitly configure the IP interface after you add the device. You must also explicitly remove the IP configuration prior to removing the device.

  For more information about dynamically configuring devices, see “Managing Devices in Oracle Solaris 11.2 ”. For more information about performing dynamic reconfiguration when you are using a fixed profile, see How to Replace a Network Interface Card With Dynamic Reconfiguration.

♦ ♦ ♦ CHAPTER 6

Administering Profile-Based Network Configuration in Oracle Solaris

This chapter describes how to administer network configuration by using various types of profiles. For an overview of profile-based network configuration and the reactive mode, see Chapter 5, “About Administering Profile-Based Network Configuration in Oracle Solaris”.

For information about configuring datalinks and IP interfaces when using the fixed mode, see Chapter 2, “Administering Datalink Configuration in Oracle Solaris” and Chapter 3, “Configuring and Administering IP Interfaces and Addresses in Oracle Solaris”.

This chapter contains the following topics:

Enabling and Disabling Profiles

You use the `netadm` command to enable and disable all profiles, regardless of the profile type or whether the profile is fixed for reactive. The basic command syntax is as follows:

```
# netadm enable [ -p profile-type ] [ -c ncu-class ] profile-name
```

For example, you would enable the system-defined Automatic NCP as follows:

```
# netadm enable -p ncp Automatic
```

For background information about enabling and disabling profiles, see “Guidelines for Using Profile-Based Network Configuration” on page 103.

Refer to the following additional guidelines when enabling and disabling the various types of profiles:

- **NCPs** – At any given time, there *must* be one active NCP and one active Location profile on the system. The active NCP remains active until you explicitly enable a different

NCP. Enabling a different NCP implicitly disables the currently active NCP. You cannot explicitly disable the NCP that is currently active on a system. “Description of an NCP” on page 99.

When you switch to the fixed mode by enabling the `DefaultFixed` NCP, the `DefaultFixed` Location is also automatically enabled and cannot be changed.

When you enable the `Automatic` NCP, the activation policy selects an appropriate corresponding Location, based on the current network conditions and then enables that Location.

- **NCUs** – You can manually enable and disable individual NCUs that are a part of the currently active NCP if the activation mode for the NCU is set to `manual`. If the link or NCU class is not specified, both NCUs are enabled or disabled. See “Description of an NCU” on page 100.
- **Locations** – By default, the system selects the best Location profile to enable. The system selects a Location from the set of locations with the `system` or `conditional` activation mode. However, the user can at any time override the system's selection by manually enabling any Location, regardless of the Location's activation mode. When you enable a Location manually, the system does not automatically change the active Location. The automatic selection of a Location is disabled. You must explicitly disable the manually enabled Location to restore the conditional Location selection by the system. See “Description of a Location Profile” on page 100.
- **ENMs** – These profiles can have a `manual` or `conditional` activation mode. If you set the `activation-mode` property to `conditional`, the system enables or disables the ENM based on the specified conditions. If you set the activation mode to `manual`, you can enable or disable the ENM by using the `netadm` command. There are no constraints on ENM activation. Zero or more ENMs can be active on a system at any given time. Enabling or disabling an ENM has no effect on other currently active ENMs. See “Profile Activation Modes” on page 105.

EXAMPLE 6-1 Enabling an NCP

In the following example, a user-defined NCP named `myncp` is enabled.

```
$ netadm enable -p ncp myncp
Enabling ncp 'myncp'
```

EXAMPLE 6-2 Enabling a Location Profile

In the following example, a Location profile named `office` is enabled.

```
$ netadm enable -p loc office
Enabling loc 'office'
```

Note that when you specify profile names, the `netadm` command is case-insensitive.

EXAMPLE 6-3 Disabling a Link NCU

In the following example, a link NCU named `net1` is disabled.

```
$ netadm disable -p ncu -c phys net1
```

EXAMPLE 6-4 Disabling a Location

In the following example, a Location named office is disabled.

```
$ netadm disable -p loc office
Disabling loc 'office'
```

EXAMPLE 6-5 Enabling and disabling ENMs

In the following example, an ENM `test-enm1` is enabled and another ENM `test-enm2` is disabled.

```
$ netadm enable -p enm test-enm1
Enabling enm 'test-enm1'
$ netadm disable -p enm test-enm2
Disabling enm 'test-enm2'
```

EXAMPLE 6-6 Switching Between the Fixed and Reactive Modes

The following example shows how to switch to the fixed mode by enabling the system-defined `DefaultFixed` NCP.

```
$ netadm enable -p ncp DefaultFixed
Enabling ncp 'DefaultFixed'
```

Configuring Profiles

You can configure reactive profiles by using the `netcfg` command in the following ways:

- Interactively
- Command-line mode
- Command-file mode

Working in the `netcfg` Interactive Mode

When used interactively, the concept of a *scope* is used for the `netcfg` command. When you use the command interactively, the scope that you are in at any given time depends on the profile

type and the particular task that you are performing. When you type the `netcfg` command by itself in a terminal window, as shown in the following example, a prompt is displayed at the *global scope*:

```
$ netcfg
netcfg>
```

To create or select a profile, you must first initiate the `netcfg` interactive session.

From the global scope prompt, you can use the `select` or `create` subcommands to view, modify, or create the following profile types, which are the top-level profiles:

- NCPs
- Locations
- ENMs
- Known WLANs

After you have created or selected a profile, the syntax of the `netcfg` interactive prompt looks similar to the following example:

```
netcfg:object-type:object-name>
```

Use the `netcfg` command in the interactive mode to perform the following tasks:

- Create a profile.
- Select and modify a profile.
- Verify that all of the required information about a profile is set and valid.
- Commit the changes for a new profile.
- Cancel the current profile configuration without committing any changes to persistent storage.
- Revert the changes that you made for a profile.

For Location profiles and ENMs, selecting or creating a top-level profile while in the `netcfg` interactive mode results in a command prompt that is displayed at the *profile scope*, as shown in the following example:

```
netcfg> select loc test-loc
netcfg:loc:test-loc>
```

When an NCP is selected, the command prompt is at the *NCP scope*. NCUs are selected and created at this scope. Selecting or creating an NCU moves the session to the *profile scope* for the selected NCP. In this scope, all of the properties that are associated with the currently selected profile can be viewed and set.

In the following example, the `office` NCP is selected, which moves the interactive session into the NCP scope for the NCP, from where an NCU is then selected. This action results in the profile scope for the selected NCU. In this scope, the properties of the NCU can be viewed or set.

```
$ netcfg
netcfg> select ncp office
netcfg:ncp:office> select ncu phys net2
netcfg:ncp:office:ncu:net2>
```

At any given scope, the command prompt indicates the currently selected profile. Any changes that you make to the profile in this scope can be *committed*, which means the changes are saved to the persistent storage. Changes are implicitly committed upon exiting the scope. If you do not want to commit the changes that you made, you can revert to the previously committed state for that profile. Doing so reverts any changes that you made to the profile at that level. The `revert` and `cancel` subcommands work similarly.

For instructions, see "Creating NCPs" on page 114.

Note - The `walkprop` subcommand is used to display each property that is associated with a profile individually. This command is meaningful only when used in the interactive mode.

Working in the `netcfg` Command-Line Mode

In command-line mode, any `netcfg` subcommand that affects a selected profile or property must be performed within the particular scope in which the selected profile or property exists. Selecting a profile in command-line mode is the same as selecting it in the interactive mode. The only difference is that in command-line mode all of the subcommands are written on the command-line. For example, to list the contents of an NCP, you would first select the NCP and then use the `list` subcommand to display the NCUs for that NCP, as shown in the following example:

```
$ netcfg "select ncp myncp; list"
ncp:myncp
 management-type reactive
NCUs:
 phys net0
 phys net1
 ip   net0
 ip   net1
```

Refer to the following guidelines when using the `netcfg` command in this mode:

- Separate each subcommand by a semi-colon.
- Specify the `select` subcommand from the global scope to move to the NCP scope.
- Specify the `list` subcommand from the NCP scope to list the properties that are within that scope.
- Use straight quotation marks to prevent the shell from interpreting the semi-colons.

Note - In command-line mode, you must type the complete command on a single line. Changes that you make to a selected profile by using the `netcfg` command in command-line mode are committed to persistent storage when the command is executed.

You can use any of the `netcfg` subcommands in command-line mode, except for the `walkprop` subcommand.

Working in the `netcfg` Command-File Mode

In command-file mode, profile configuration information and commands are extracted from a file. The commands in the file are same as those that are used in the interactive mode and by the `export` subcommand. The export subcommand with the `-f` option is used to produce the file. For example, the following command exports the current profile configuration to a file:

```
$ netcfg export -f /tmp/nwam.config
```

To import a profile configuration from a file, type the following command:

```
$ netcfg -f /tmp/nwam.config
```

The `export` subcommand can also be used interactively. For more information, see “Exporting a Profile Configuration” on page 135.

Creating NCPs

An NCP defines the network configuration of the system. You can only create reactive NCPs, not fixed NCPs.

To create an NCP in the interactive mode, you begin by initiating the interactive session. Then, you use the `create` subcommand to create the new NCP, as shown in the following example:

```
$ netcfg
netcfg> create ncp myncp
netcfg:ncp:myncp>
```

EXAMPLE 6-7 Creating an NCP by Cloning the `Automatic` NCP

You can create an NCP by cloning any existing NCP other than the `DefaultFixed` NCP. You can then modify the NCP's properties to specify new configuration parameters. In the following example, a new NCP named `newncp` is created by cloning the system-defined `Automatic` NCP.

```
$ netcfg
```

```
netcfg> create -t Automatic ncp newncp
netcfg:ncp:newncp> list
ncp:newncp
 management-type reactive
NCUs:
 phys net0
 phys net1
 ip   net0
 ip   net1
```

The `management-type` property is a read-only property that is always set to `reactive`. In the previous example, the `list` subcommand is used to display the contents of the newly copied NCP (`newncp`).

You can also directly clone the `Automatic` NCP by using the `netcfg` command-line mode, as shown in the following example. Both the interactive and command-line methods copy the existing NCUs from the `Automatic` NCP to the newly created NCP.

```
$ netcfg create -t Automatic ncp newncp
$ netcfg list ncp newncp
ncp:newncp
 management-type reactive
NCUs:
 phys net0
 phys net1
 ip   net0
 ip   net1
```

Creating NCUs for an NCP

The NCP is essentially a container that consists of a set of NCUs that are configured with properties that define the network configuration for the NCP. All NCPs contain both link and interface NCUs.

Link NCUs specify both the link configuration and the link selection policy for the NCP. Interface NCUs specify the interface configuration policy for the NCP. If IP connectivity is required, both a link and an interface NCU are required. NCUs must be explicitly added or removed by using the `netcfg` command or by using the Network Administration GUI. For more information about adding and removing NCUs by using the Network Administration GUI, see the “Administering Network Configuration From the Desktop” on page 137.

You create NCUs by using the `netcfg` command either interactively or in command-line mode. Because creating an NCU involves several operations, it is easier and more efficient to create NCUs interactively, rather than trying to construct a single-line command that creates the NCU and all of its properties. NCUs can be created when you initially create an NCP or afterward. The process of creating or modifying an NCU involves setting general NCU properties, as well as setting properties that specifically apply to each NCU type.

When you create an NCU interactively, the `netcfg` command walks you through each relevant property for the NCU, displaying both the default value (when a default exists) and all possible values. For example, if you specify `dhcp` for the `ipv4-addrsrc` property of an interface NCU, you are not prompted to specify a value for the `ipv4-addr` property because this property is used for a static IP address configuration only. You can specify alternate values for each property at the interactive prompt. Pressing Return without specifying a value re-applies the default value or leaves the property empty if there is no default value.

There are several NCU properties that you can specify when creating or modifying an NCU. Some properties apply to both NCU types, while others apply to either a link NCU or an interface NCU. For a complete description of all of the NCU properties, including rules and conditions that might apply when specifying these properties, see the `netcfg`(1M) man page.

▼ How to Create NCUs for an NCP Interactively

The following procedure describes how to select an existing NCP and then create NCUs for the NCP interactively.

Note - The “walk” process that is performed during the initial profile creation ensures that when modifying the NCP you are prompted for only those properties that are applicable, given the choices you made during its creation.

---

1. Initiate the `netcfg` interactive session.

```
$ netcfg
netcfg>
```

2. Select an existing NCP.

In the following example, the NCP `myncp` is selected:

```
netcfg> select ncp myncp
netcfg:ncp:myncp>
```

Selecting the NCP automatically takes you into that NCP's scope. For a Location, an ENM, or a WLAN object, the command prompt would take you into the profile scope for that profile.

3. Create the link and interface NCUs for the NCP.

In the following example, a link NCU is created:

```
netcfg:ncp:myncp> create ncu phys net0
Created ncu `net0'. Walking properties ...
activation-mode (manual) [manual|prioritized]>
mac-address>
```

```
autopush>
mtu> 1600
```

where `ncu` is the object type, `phys` is the class of NCU, and `net0` is the object name.

Creating an NCU moves you into that object's scope and walks you through the default properties for the object.

In this example, the following properties are specified:

- The `activation-mode` property, which defaults to `manual`, is accepted by pressing Return.
- The `mac-address` and `autopush` properties are left empty.
- The `mtu` property is set to a value of `1600`.

The following example shows how to create an interface NCU:

```
netcfg:ncp:myncp> create ncu ip net0
Created ncu `net0'. Walking properties ...
ip-version (ipv4,ipv6) [ipv4|ipv6]> ipv4
ipv4-addrsrc (dhcp) [dhcp|static]> dhcp
ipv4-default-route>
```

where `ncu` is the object type, `ip` is the object class, and `net0` is the object name.

Creating an NCU moves you into that object's scope and walks you through the default properties for the object.

In this example, the following properties are specified:

- The `ip-version` property is set to `ipv4`.
- The `ipv4-addrsrc` property is set to `dhcp`.

During the creation of an NCU, the `class` option is used to differentiate between the two types of NCUs.

4. **(Optional) Verify that the configuration is correct as follows:**

```
netcfg:ncp:myncp:ncu:net0> verify
All properties verified
```

The `verify` subcommand verifies the configuration and notifies you if any of the required values are missing.

5. **Save each NCU that you create.**

   - **Use the `commit` subcommand:**

     ```
     netcfg:ncp:myncp:ncu:net0> commit
     Committed changes
     netcfg:ncp:myncp:ncu:net0>
     ```

The `commit` subcommand implicitly verifies the properties.

- **Use the `end` subcommand:**

```
netcfg:ncp:myncp:ncu:net0> end
Committed changes
netcfg:ncp:myncp>
```

The `end` subcommand implicitly commits the changes.

In this instance, if you are done adding NCUs to the NCP, the `end` subcommand moves the session to the NCP scope.

In the interactive mode, changes are not saved to persistent storage until you commit them. When you use the `commit` subcommand, the entire profile is committed. To maintain the consistency of persistent storage, the `commit` operation also includes a verification step. If the verification fails, the `commit` operation also fails. If an implicit commit fails, you are given the option of ending or exiting the interactive session without committing the current changes. Alternatively, you can remain in the current scope and continue making changes to the profile.

Note - To cancel the changes that you made, use the `cancel` or the `revert` subcommand.

The `cancel` subcommand ends the current profile configuration without committing the current changes to persistent storage, then moves the interactive session up one scope level. The `revert` subcommand undoes the changes that you made and rereads the previous configuration. When you use the `revert` subcommand, the interactive session stays within the same scope.

---

6. **When you are finished, exit the interactive session.**

```
netcfg:ncp:myncp> exit
```

The `exit` subcommand is similar to the `end` subcommand but also exits the interactive session.

Creating Locations

A Location profile contains properties that define network configuration values that are not directly related to basic link and IP connectivity. Some examples include naming service and IP Filter settings that are applied together, when required. Exactly one Location profile and one NCP must be active on the system at all times.

You create Location profiles by using the `netcfg` command either interactively or in command-line mode. When you create a Location profile, you set properties for the Location by specifying values that define the particular configuration parameters when that particular

Location is enabled. Location properties are categorized by group, where the group signifies a particular class of configuration preferences.

Location properties are also stored in a repository. When a particular Location is enabled, its properties are automatically applied to the running system. Creating and modifying Location profiles also involves setting properties that define when a particular Location is enabled. The properties that you are presented with during the configuration process are based on property values that were previously set.

For a complete description of all of the Location profile properties, including any rules, conditions, and dependencies that might apply when specifying any of these properties, see the `netcfg`(1M) man page.

▼ How to Create a Location Interactively

The following procedure describes how to create a Location profile. The “walk” process that is performed during the initial profile creation only prompts you for those properties that are applicable, given the values that you entered previously.

1. Initiate the `netcfg` interactive session.

```
$ netcfg
netcfg>
```

2. Create the Location profile.

In the following example, a Location named `office` is created:

```
netcfg> create loc office
Created loc 'office'. Walking properties ...
activation-mode (manual) [manual|conditional-any|conditional-all]> conditional-any
conditions> ncu ip:net0 is active
nameservices (dns) [dns|files|nis|ldap]>
nameservices-config-file ("/etc/nsswitch.dns")>
dns-nameservice-configsrc (dhcp) [manual|dhcp]>
nfsv4-domain>
ipfilter-config-file> /export/home/test/wifi.ipf.conf
ipfilter-v6-config-file>
ipnat-config-file>
ippool-config-file>
ike-config-file> /etc/inet/ike/ikev1.config
ikev2-config-file>
ipsecpolicy-config-file>
```

Creating the Location automatically moves you to into the profile scope for the Location.

In this example, the following properties were set:

- The `activation-mode` property was set to `conditional-any`, which resulted in a command prompt that enabled the conditions for activation to be specified.
- The condition for activation of the location was specified as `ncu ip:net0 is active`.
- For the `ipfilter-config-file` property, the `/export/home/test/wifi.ipf.conf` file was specified.
- For the `ike-config-file` property, the `/etc/inet/ike/ikev1.config` file was specified.
- For the remaining properties, the default values were accepted by pressing Return.

3. (Optional) Display the profile configuration by using the `list` subcommand as follows:

```
netcfg:loc:office> list
loc:office
        activation-mode                 conditional-any
        conditions                      "ncu ip:net0 is active"
        enabled                         false
        nameservices                    dns
        nameservices-config-file        "/etc/nsswitch.dns"
        dns-nameservice-configsrc       dhcp
        ipfilter-config-file            "/export/home/test/wifi.ipf.conf"
        ike-config-file                 "/etc/inet/ike/ikev1.config"
```

4. Verify that the profile configuration is correct.

```
netcfg:loc:office> verify
All properties verified
```

The `verify` subcommand verifies the configuration and notifies you if any required values are missing.

5. When you have verified the configuration, save the Location.

```
netcfg:loc:office> commit
Committed changes
```

Alternatively, you can use the `end` subcommand to end the session, which also saves the profile configuration and moves the session to the global scope.

```
netcfg:loc:office> end
Committed changes
netcfg>
```

6. Exit the interactive session.

```
netcfg> exit
```

Creating ENMs

ENMs enable you to specify when applications or scripts such as a VPN application, should perform network configuration that is separate (external) from the configuration that is specified in the NCP and Location profiles. For more information about ENMs, see "Description of an ENM" on page 101.

Note - The system does not automatically recognize an application for which you might create an ENM. These applications must first be installed and then configured on the system before creating an ENM for them by using the `netcfg` command.

For more information about properties that you might specify when creating an ENM, see the `netcfg`(1M) man page.

▼ How to Create an ENM Interactively

1. **Initiate the `netcfg` interactive session.**

```
$ netcfg
netcfg>
```

2. **Create the ENM.**

```
netcfg> create enm test-enm
Created enm 'test-enm'.  Walking properties ...
activation-mode (manual) [manual|conditional-any|conditional-all]>
fmri> svc:/application/test-enm:default
start>
stop>
netcfg:enm:test-enm>
```

Creating the ENM automatically moves you into the profile scope for the ENM and walks each of its properties.

In this example, the following properties are specified for the `test-enm` ENM:

- The `activation-mode` property, which is set to manual, is accepted by pressing Return. Because this value is set to `manual`, the `conditions` property is not made available for setting.
- The `fmri` property is set to `svc:/application/test-enm:default`.
- The `start` and `stop` properties are not set for this ENM.

3. **(Optional) Display the profile configuration.**

```
netcfg:enm:test-enm> list
enm:test-enm
 activation-mode manual
 enabled   false
 fmri      "svc:/application/test-enm:default"
```

4. **Verify that the profile configuration is correct.**

```
netcfg:enm:test-enm> verify
All properties verified
```

The `verify` subcommand verifies the configuration and notifies you if any required values are missing.

5. **Save the ENM.**

```
netcfg:enm:test-enm> commit
Committed changes
netcfg>
```

The `commit` subcommand verifies and saves the configuration.

Alternatively, you can use the `end` subcommand to end the session, which also saves the profile configuration.

```
netcfg:enm:test-enm> end
Committed changes
```

6. **Exit the interactive session.**

```
netcfg> exit
```

Creating Known WLANs

Known WLANs are profiles that store information about wireless networks that are known to your system. NCPs are able to automatically configure wireless interfaces based on configuration information that is supplied by each of the wireless networks to which the system is connected. For more information, see “Description of a Known WLAN” on page 102.

For information about the properties that you might specify when creating or modifying WLANs, see the `netcfg`(1M) man page.

▼ How to Create a Known WLAN Interactively

1. **Initiate the `netcfg` interactive session.**

```
$ netcfg
netcfg>
```

2. Create the Known WLAN.

The following example creates a Known WLAN that connects to a wireless network named ESSID. The Known WLAN must have the same name as the wireless network name or its ESSID:

```
netcfg> create wlan mywifi
Created wlan 'mywifi'.  Walking properties ...
priority (0)> 100
bssids>
keyname> mywifi-key
keyslot>
security-mode [none|wep|wpa]> wpa
netcfg:wlan:mywifi>
```

Creating the WLAN automatically moves you into the profile scope for the WLAN and walks you through each of its properties.

In this example, the following properties are specified for the `mywifi` Known WLAN:

- The value of the `priority` property is changed from a default value of `0` to `100`.
- The `keyname` property is set to `mywifi-key` and specifies the name of the secure object for this wireless network. See “Establishing Secure WiFi Communications” on page 144 for more information.
- The `security-mode` property is set to `wpa`. This property specifies the type of encryption that is used by this wireless network.
- The `keyslot` and `bssid` property values are left empty.

3. (Optional) Display the profile configuration.

```
netcfg:wlan:mywifi> list
known wlan:mywifi
 priority   100
 keyname "mywifi-key"
 security-mode    wpa
netcfg:wlan:mywifi>
```

4. Verify that the profile configuration is correct.

```
netcfg:wlan:mywifi> verify
All properties verified
```

The `verify` subcommand verifies the configuration and notifies you if any required values are missing.

5. Save the Known WLAN.

```
netcfg:wlan:mywifi> commit
```

```
Committed changes
```

The `commit` subcommand verifies and saves the configuration.

Alternatively, you can use the `end` subcommand to end the session, which also saves the profile configuration and moves the session to the global scope.

```
netcfg:wlan:mywifi> end
Committed changes
netcfg>
```

6. **Exit the interactive session.**

```
netcfg> exit
```

Administering Profiles

This section contains the following topics:

Setting Property Values for Profiles

Property values for reactive profiles are set or modified by using the `set` subcommand. This subcommand can be used interactively or in command-line mode. If a property value is set or changed in command-line mode, the change is immediately committed to persistent storage.

Note - You cannot modify the `DefaultFixed` NCP or the `DefaultFixed` Location profile by using the `set` subcommand. Whenever the `DefaultFixed` NCP is active, use the `dladm` and `ipadm` commands to make configuration changes. When the `DefaultFixed` Location is active, make changes directly to the relevant SMF properties by using the `svccfg` and `svcadm` commands. See the `svccfg`(1M) and `svcadm`(1M) man pages.

The syntax for the `set` subcommand is as follows:

```
netcfg> set prop-name=value1[,value2,...]
```

▼ How to Set Profile Property Values Interactively

The following procedure describes how to set property values for a Location profile interactively. When setting property values interactively, you must first select a profile from the current scope, which moves the interactive session into that profile's scope. The selected profile is then loaded into memory from persistent storage. In this scope, you can then modify the properties of the profile.

For example purposes only, the following procedure shows how to set the `ipfilter-config-file` property of the `test-loc` Location interactively.

1. **Initiate the `netcfg` interactive session.**

```
$ netcfg
netcfg>
```

2. **Select the profile or configuration object to modify.**

```
netcfg> select loc test-loc
netcfg:loc:test-loc>
```

3. **Set the property value.**

In the following example, the `ipfilter-config-file` property is set:

```
netcfg:loc:test-loc> set ipfilter-config-file = /path/to/ipf-file
```

4. **(Optional) List the configuration information.**

```
netcfg:loc:test-loc> list
loc:test-loc
 activation-mode   manual
 enabled        false
 nameservices        dns
 dns-nameservice-configsrc  dhcp
 nameservices-config-file  "/etc/nsswitch.dns"
 ipfilter-config-file   "/path/to/ipf-file"
```

5. **End the session.**

```
netcfg:loc:test-loc> end
Committed changes
netcfg>
```

The `end` subcommand saves and moves the session to the global scope.

6. **Exit the interactive session.**

```
netcfg> exit
```

Example 6-8 Setting Property Values for a Profile in Command-Line Mode

The previous example that shows how to interactively set the `ipfilter-config-file` property can also be performed in command-line mode as follows:

```
$ netcfg "select loc test-loc; set ipfilter-config-file = /path/to/ipf-file"
```

The command-line mode is best suited for when you only need to perform a simple action. However, you can also use the command-line mode to perform more complex actions by carefully specifying the appropriate subcommands on the command line. As with the interactive example, in command-line mode you must also first select the Location to move into that profile's scope. You can then specify the `set` subcommand to set the individual property values.

When using the command-line mode, multiple values can be set for a given property at the same time. When setting multiple values in this manner, each value must be separated by a comma (`,`). If individual values for a specified property also contain a comma, the comma that is part of the property value must be preceded by a backslash (`\`). Commas within properties that only have a single value are not interpreted as delimiters and therefore do not need to be preceded by a backslash.

For example, you would set the `ip-version` property to use both IPv4 and IPv6 for the `net0` NCU in the `myncp` NCP as follows:

```
$ netcfg "select ncp myncp; select ncu ip net0; set ip-version=ipv4,ipv6"
```

Obtaining Information About Profile Configuration

Use the `list` subcommand to list all of the profiles, property-value pairs, and resources that exist at the current or specified scope, either interactively or in command-line mode.

In the global scope, the `list` subcommand lists all of the system-defined and user-defined profiles on a system, as shown in the following example:

```
$ netcfg list
netcfg list
NCPs:
 DefaultFixed
 Automatic
 myncp
Locations:
 Automatic
```

```
 NoNet
 DefaultFixed
 office
ENMs:
 test-enm
WLANs:
 mywifi
```

Note - In the interactive mode, using the `list` subcommand in the global scope lists the same information.

Note that the `list` subcommand does not list the state of each profile. To display information about profiles and their states, use the `netadm` command with the `list` subcommand. For more information, see "Displaying the Current State of a Profile" on page 132.

Listing Property Values for an Individual Profile

The `list` subcommand in the profile scope lists all of the property values for a specified profile. The syntax is as follows:

```
netcfg> list [ object-type [ class ] object-name ]
```

▼ How to List All of the Property Values for a Profile Interactively

The following procedure describes how to list all of the property values for a profile by using the `list` subcommand interactively. The example in the following procedure shows how to list the configuration information for an IP NCP name `net0`.The values that are listed for each profile vary, depending on the profile.

1. **Initiate the `netcfg` interactive session.**

```
$ netcfg
netcfg>
```

2. **Select the NCP.**

```
netcfg> select ncp myncp
netcfg:ncp:myncp>
```

3. **List the configuration for the IP NCU by using one of the following methods:**

   - **List the configuration from the global scope:**

```
netcfg:ncp:myncp> list ncu ip net0
ncu:net0
        type                    interface
        class                   ip
        parent                  "myncp"
        enabled                 false
        ip-version              ipv4,ipv6
        ipv4-addrsrc            dhcp
        ipv6-addrsrc            dhcp,autoconf
netcfg:ncp:myncp>
```

- **List the configuration from the profile scope:**

```
netcfg:ncp:myncp> select ncu ip net0
netcfg:ncp:myncp:ncu:net0> list
ncu:net0
        type                    interface
        class                   ip
        parent                  "myncp"
        enabled                 false
        ip-version              ipv4,ipv6
        ipv4-addrsrc            dhcp
        ipv6-addrsrc            dhcp,autoconf
netcfg:ncp:myncp:ncu:net0>
```

4. **Exit the interactive session.**

```
netcfg:ncp:myncp:ncu:net0> exit
```

Example 6-9 Listing Property Values in Command-Line Mode

The previous example that shows how to list property values interactively can be performed in command-line mode as well. The output is identical, regardless of which mode you use.

For example, you would list the properties of the `net0` IP NCU from the NCP scope as follows:

```
$ netcfg "select ncp myncp; list ncu ip net0"
```

To list the properties of the `net0` IP NCU from the profile scope, you would use the following command:

```
$ netcfg "select ncp myncp; select ncu ip net0; list"
```

Obtaining a Specific Property Value for a Profile

Use the `get` subcommand to obtain a specific property value for a profile. This subcommand can be used either interactively or in command-line mode. The command syntax is as follows:

```
netcfg> get [ -V ] prop-name
```

How to Obtain a Specific Property Value for a Profile

The following procedure describes how to obtain a specific property value for a profile by using the `get` subcommand interactively. The example in the following procedure shows how to obtain the `ip-version` property of an IP NCU.

1. **Initiate the `netcfg` interactive session.**

```
$ netcfg
netcfg>
```

2. **Select the NCP, then select the IP NCU. For example:**

```
netcfg> select ncp myncp
netcfg:ncp:myncp> select ncu ip net0
netcfg:ncp:myncp:ncu:net0>
```

3. **Obtain the specific property value by using one of the following commands:**

   - **Use the `get` subcommand to display the property name and the property value as follows:**

```
netcfg:ncp:myncp:ncu:net0> get ip-version
 ip-version ipv4,ipv6
```

   - **If you only want to obtain the property value, without displaying the property name, use the `-V` option with the `get` subcommand, as follows:**

```
netcfg:ncp:myncp:ncu:net0> get -V ip-version
ipv4,ipv6
```

4. **Exit the interactive session.**

```
netcfg:ncp:myncp:ncu:net0> exit
```

Example 6-10 Obtaining a Specific Property Value for a Profile in Command-Line Mode

The previous interactive example can also be performed in command-line mode. The output is identical, regardless of which mode you use.

For example, you would obtain the value of the `ip-version` property for an IP NCU in command-line mode as follows:

```
$ netcfg "select ncp myncp; select ncu ip net0; get ip-version"
 ip-version ipv4,ipv6
```

The following example shows how to use the `get` subcommand with the `-V` option to obtain a specific property value. This method is useful for scripts, where the property name does not need to be parsed.

```
$ netcfg "select ncp myncp; select ncu ip net0; get -V ip-version"
ipv4,ipv6
```

Setting Property Values for a Profile by Using the `walkprop` Subcommand

Use the `walkprop` subcommand to interactively view and change individual property values for a profile. After initiating the interactive session, typing the `walkprop` subcommand enables you to display the name and current value for each of the properties of the profile, one property at a time. As you view the various properties, you can set or change the current or default value, as desired.

Note - The `walkprop` subcommand is meant to be used in the interactive mode only.

▼ How to "Walk" and Set Property Values for a Given Profile

The following procedure describes how to use the `walkprop` subcommand to interactively view and change property values for a given profile. As shown in the following examples, when you use the `walkprop` subcommand to set the properties of a profile, you do not have to use the `set` subcommand.

1. **Initiate the `netcfg` interactive session.**

   ```
   $ netcfg
   netcfg>
   ```

2. **Select the profile for which you want to view and change properties.**

   In the following example, a Location named `test-loc` is selected:

   ```
   netcfg> select loc test-loc
   netcfg:loc:test-loc>
   ```

3. **Type the `walkprop` subcommand to start the walk.**

   In the following example, after the `walkprop` subcommand is issued, the first property that is displayed is the `activatation-mode` property. Note the default value of the property, which is currently set to `manual` (shown in parentheses).

   ```
   netcfg:loc:test-loc> walkprop
   activation-mode (manual) [manual|conditional-any|conditional-all]>
   ```

4. **To modify a property's value, type the new value at interactive prompt, then press Return.**

   For example, you would change the `activation-mode` property for the Location from `manual` to `conditional-all` as follows:

```
netcfg:loc:test-loc> walkprop
activation-mode (manual) [manual|conditional-any|conditional-all]> conditional-all
```

   Pressing Return saves the current setting and walks the next property.

5. **Repeat the walk process until all of the properties for the profile have been displayed, making modifications as needed, per the instructions in Step 4.**

```
netcfg:loc:test-loc> walkprop
activation-mode (manual) [manual|conditional-any|conditional-all]> conditional-all
conditions> advertised-domain is example.com
nameservices (dns) [dns|files|nis|ldap]>
nameservices-config-file ("/etc/nsswitch.dns")>
dns-nameservice-configsrc (dhcp) [manual|dhcp]>
nfsv4-domain>
ipfilter-config-file>
ipfilter-v6-config-file>
ipnat-config-file>
ippool-config-file>
ike-config-file>
ikev2-confg-file>
ipsecpolicy-config-file>
```

   Pressing Return without making any changes to a property retains the existing default value and then advances the walk to the next property.

Note - Only the relevant properties for a given profile are displayed, as described in "How to Create a Location Interactively" on page 119.

---

6. **List the current default property values for the profile. For example:**

```
netcfg:loc:test-loc> list
loc:test-loc
 activation-mode    conditional-all
 conditions      "advertised-domain is example.com"
 enabled      false
 nameservices       dns
 nameservices-config-file    "/etc/nsswitch.dns"
 dns-nameservice-configsrc   dhcp
```

   Note that in the previous output, the `activation-mode` property is now set to `conditional-all`.

7. **Exit the interactive session to commit the changes.**

```
netcfg:loc:test-loc> exit
```

```
Committed changes
```

Displaying Information About Profiles

Use the `netadm` command with the `list` subcommand to display information about any or all of the profiles that are on a system, including the current state of each profile.

The syntax for the `list` subcommand is as follows:

```
$ netadm list [ -p object-type ] [ -c ncu-class ] [ object-name ]
```

You would display all of the profiles on a system and the current state of each profile as follows:

```
$ netadm list
TYPE   PROFILE    STATE
ncp    DefaultFixed  disabled
ncp    Automatic  online
ncu:phys  net0    online
ncu:phys  net1    offline
ncu:ip    net0    online
ncu:ip    net1    offline
loc    Automatic  online
loc    NoNet   offline
loc    DefaultFixed  offline
enm    test-enm   disabled
```

Note that the `list` subcommand displays the enabled NCP and all of the NCUs that make up that particular NCP.

Displaying the Current State of a Profile

The profile type and NCU class can be included with the `list` subcommand to identify a specific profile. If only a profile type is provided, all of the profiles that are of that type are displayed. If a profile is specified by its name, the current state of that profile is displayed. If the profile name is not unique, all of the profiles with that name are listed.

EXAMPLE 6-11 Displaying the Current State of a Specified Profile

The following example lists the current state of all of the profiles that are on the system with the name `Automatic`.

```
$ netadm list Automatic
TYPE         PROFILE        STATE
ncp          Automatic      online
ncu:ip       net1           offline
ncu:phys     net1           offline
ncu:ip       net0           online
```

```
ncu:phys     net0           online
loc          Automatic      online
```

In the following example, the `list` subcommand is used with the `-p` option to display all of the locations that are currently on the system. In the particular example, the **`Automatic`** Location is online (enabled).

```
$ netadm list -p loc
TYPE  PROFILE   STATE
loc  DefaultFixed  offline
loc  NoNet   offline
loc  Automatic  online
```

In the following example, the `list` subcommand is used with the `-c` option to display all of the interface NCUs in the active NCP.

```
$ netadm list -c ip
TYPE         PROFILE         STATE
ncu:ip       net0            online
ncu:ip       net1            offline
```

Displaying Auxiliary State Values of Profiles

The auxiliary state of a profile provides an explanation about why a given profile is online or offline (enabled or disabled). To list auxiliary state values, use the `-x` option with the `list` subcommand as follows:

```
$ netadm list -x
TYPE        PROFILE        STATE          AUXILIARY STATE
ncp         DefaultFixed   disabled       disabled by administrator
ncp         Automatic      online         active
ncu:phys    net0           online         interface/link is up
ncu:phys    net1           offline        interface/link is down
ncu:ip      net0           online         interface/link is up
ncu:ip      net1           offline        conditions for activation are unmet
loc         Automatic      offline        conditions for activation are unmet
loc         NoNet          offline        conditions for activation are unmet
enm   test-enm    disabled    disabled by administrator
loc         DefaultFixed   offline        conditions for activation are unmet
```

Auxiliary state values vary, depending on the profile type. For detailed information about auxiliary states, see the nwamd(1M) man page.

Removing Profiles

Use the `destroy` subcommand to remove user-defined profiles and configuration objects such as NCUs. You *cannot* remove system-defined profiles, which include the following: the

`Automatic` and `DefaultFixed` NCPs and the `Automatic`, `NoNet`, and `DefaultFixed` Locations. Note also that you cannot remove a profile that is currently active. You must first disable the profile, and then remove it.

The syntax for the `destroy` subcommand is as follows:

```
netcfg> destroy [ -a | object-type [ class ] object-name]
```

The `-a` option removes all of the user-defined profiles from the system, except for any currently active user-defined profiles.

▼ How to Remove a Profile Interactively

The following procedure describes how to remove a user-defined profile interactively. For example purposes only, this procedure shows how to remove an IP NCU from the `myncp` user-defined NCP.

1. **Initiate the `netcfg` interactive session.**

```
$ netcfg
netcfg>
```

2. **Select the profile.**

For example, to select the `myncp` NCP, you would type the following command:

```
netcfg> select ncp myncp
netcfg:ncp:myncp>
```

3. **Remove the profile or configuration object.**

In the following example, the `net1` IP NCU is removed from the `myncp` NCP:

```
netcfg:ncp:myncp> destroy ncu ip net1
netcfg:ncp:myncp>
```

4. **Exit the interactive session. For example:**

```
netcfg:ncp:myncp> exit
```

Example 6-12 Removing a Profile in Command-Line Mode

The previous example that shows how to interactively remove a profile can also be performed in command-line mode as follows:

```
$ netcfg "select ncp myncp; destroy ncu ip net1"
```

Exporting a Profile Configuration

Use the `export` subcommand to save a profile configuration. Exporting a profile can be useful if you are responsible for maintaining multiple servers that require identical network configurations. The `export` subcommand can be used either interactively or in command-line mode. When a profile is exported, the output is displayed as a series of subcommands that the `netcfg` command is capable of interpreting. These subcommands are similar to commands that you would type in the interactive or command-line mode.

Note - The `export` feature is of limited use for some configurations. You can only export configuration objects that were initially created by using the `netcfg` command. You cannot export configuration objects that were created by using the `dladm` or `ipadm` command, for example, link aggregations or IPMP groups. Also, you cannot export the `DefaultFixed` NCP and `DefaultFixed` Locations.

The syntax for the `export` subcommand is as follows:

```
netcfg> export [ -d ] [ -f output-file ] [ object-type  [ class ] object-name ]
```

Note - The `-d` and `-f` options of the `export` subcommand can be used independently of each other. The `-f` option prints the current configuration at the current or specified scope to a specified file. The `-d` option adds the `destroy -a` command as the first line of output.

EXAMPLE 6-13 Exporting a Profile Configuration Interactively

The following example shows how to display a profile configuration on-screen by using the `export` subcommand interactively.

```
$ netcfg
netcfg> export
create ncp "myncp"
create ncu ip "net0"
set ip-version=ipv4
set ipv4-addrsrc=dhcp
set ipv6-addrsrc=dhcp,autoconf
end
create ncu phys "net0"
set activation-mode=manual
set mtu=5000
end
end
create loc "test-loc"
set activation-mode=conditional-all
```

```
set conditions="system-domain is example.com"
set nameservices=dns
set nameservices-config-file="/etc/nsswitch.dns"
set dns-nameservice-configsrc=dhcp
end
create enm "test-enm"
set activation-mode=conditional-all
set conditions="ip-address is-not-in-range 10.2.3.4"
set fmri="svc:/application/test-enm:default"
end
create wlan "mywifi"
set priority=100
set keyname="mywifi-key"
set security-mode=wpa
end
```

In the command-line mode, you would type the following command:

```
$ netcfg export
```

You can use the `-d` option with the `export` subcommand to add the `destroy -a` command as the first line of the `netcfg export` output, as shown in the following example, which has been truncated for the sake of brevity:

```
$ netcfg
netcfg> export -d
destroy -a
create ncp "myncp"
create ncu ip "net0"
set ip-version=ipv4
set ipv4-addrsrc=dhcp
.
.
.
```

In the command-line mode, you would type the following command:

```
$ netcfg export -d
```

EXAMPLE 6-14 Exporting a Profile Configuration to a File

In the following examples, the configuration information for the `myncp` NCP is written to a file by using the `export` subcommand with the `-f` option. In the following example, the `-f` option writes the output to a new file named `myncp2`. The `-d` option is used to add the `destroy -a` command as the first line of the `netcfg export` output.

You would export the profile configuration to a file interactively as follows:

```
$ netcfg
netcfg> export -d -f myncp2
```

You would perform the same task in command-line mode as follows:

```
$ netcfg export -d -f myncp2
```

The following truncated example shows how you would display the profile configuration:

```
$ cat myncp2
destroy -a
create ncp "myncp"
create ncu ip "net0"
.
.
.
```

Restoring an Exported Profile Configuration

You can import a profile configuration that was generated with the `export` subcommand back into the system. The exported file has a series of subcommands that the `netcfg` command is capable of interpreting. The `export` subcommand is useful if you need to restore a profile configuration or copy a profile configuration to another system. Note that you can only restore a profile configuration by using the `netcfg` command-line mode.

Use the following command to restore an exported configuration:

```
$ netcfg -f file
```

For example, you would restore the file that was exported in Example 6-14 as follows:

```
$ netcfg -f myncp2
Configuration read.
```

Administering Network Configuration From the Desktop

You can manage network configuration from the desktop by using the Network Administration GUI (formerly NWAM). The tool is similar to using the `netcfg` and `netadm` commands. With the GUI, you can connect to a wired or wireless network, configure a new wired or wireless connection, create Location profiles, and activate or deactivate profiles. Managing network configuration from the desktop works best for users of notebook PCs and in situations where network conditions change often, for example when switching from a home office to a wireless network at work, or when traveling.

For task-related information about managing wireless networks from the desktop, see “Administering Wireless Networks From the Desktop” on page 147.

When managing network configuration from the desktop, follow these general guidelines and best practices:

- When managing network configuration from the desktop, the simplest solution is to enable the system-defined `Automatic` NCP. At home, you can use this NCP to connect to your wireless network.
- If you decide that you want to use a wired connection, plug in the Ethernet cable. Do not switch from the default `Automatic` NCP to another NCP. The network connection will automatically adapt from a wireless network connection to a wired network connection without having to make any other changes to your existing network configuration.
- At the office, the same rules apply. If no Ethernet cable is plugged into the network, and the `Automatic` NCP is enabled, reactive networking is used and a wireless network connection is automatically established.
- If the `DefaultFixed` NCP is currently active, you can view its status *only*. To configure the network when this NCP is active, you must use the `dladm` and `ipadm` commands. See Chapter 2, "Administering Datalink Configuration in Oracle Solaris" and Chapter 3, "Configuring and Administering IP Interfaces and Addresses in Oracle Solaris".
- Keep in mind that for both the home and office scenario, you must initially choose a wireless network and save it to your list of favorite wireless networks, if you have not already done so.

  Choose a wireless network by using the Network Administration GUI or by running the `netadm select-wifi` command. See "Administering Known WLANs in Reactive Mode" on page 147.
- To view the status of your current network connection, hover your mouse over the Network Status notification icon that is located on the desktop or just click the icon. The Network Status notification icon also includes a contextual menu for creating and managing network configuration with the GUI.

  If the Network Status notification icon is not visible in the desktop, start it by choosing System → Administration → Network. To start the GUI from the command line, run the `nwam-manager` command. See the `nwam-manager`(1M) man page in the JDS/GNOME man page collection for more information.
- IP-related configuration is managed in the Network Profile section of the Network Preferences dialog box. Access the Network Preferences dialog box by clicking the Network Status notification icon that is located on the desktop or by choosing the Network Preferences option from the Network Status notification icon's contextual menu.

♦ ♦ ♦ CHAPTER 7

Administering Wireless Networks in Oracle Solaris

This chapter includes tasks for administering wireless networks in the Oracle Solaris release.

The IEEE 802.11 specifications define wireless communications for local area networks. These specifications and the networks they describe are referred to collectively as *WiFi*, a term that is trademarked by the Wi-Fi Alliance trade group. WiFi networks are reasonably easy to configure by both providers and prospective clients. Therefore, they are increasingly popular and in common use throughout the world. WiFi networks use the same radio wave technology as cellular phones, televisions, and radios.

Note - Oracle Solaris does not contain features for configuring WiFi servers or access points.

This chapter contains the following topics:

Administering Wireless Networks by Using the Command Line

The following tasks are described in this section:

How to Connect to a WiFi Network

Before You Begin Perform the following steps to connect your notebook PC to a WiFi network.

1. **Become an administrator.**

2. **Display the physical attributes of the datalinks that are on the system.**

```
# dladm show-phys
LINK            MEDIA               STATE    SPEED  DUPLEX    DEVICE
net0            Ethernet            up       1500   full      ath0
net1            Ethernet            up       1500   full      e1000g0
```

In the previous example, the output indicates that two links are available. `net0` over the device `ath0` link supports WiFi communications. The `e1000g0` link enables you to connect the system to a wired network.

3. **Configure the WiFi interface.**

   a. **Create the interface that supports WiFi. For example:**

```
# ipadm create-ip net0
```

   b. **Verify that the link has been plumbed.**

```
# ipadm show-if
IFNAME      CLASS        STATE     ACTIVE     OVER
lo0         loopback     ok        yes        --
net0        ip           ok        yes        --
```

4. **Check for available networks.**

```
# dladm scan-wifi
LINK       ESSID        BSSID/IBSSID       SEC     STRENGTH   MODE   SPEED
net0       ofc          00:0e:38:49:01:d0  none     good       g      54Mb
net0       home         00:0e:38:49:02:f0  none     very weak  g      54Mb
net0       linksys      00:0d:ed:a5:47:e0  none     very good  g      54Mb
```

The `scan-wifi` command displays information about the available WiFi networks at the current location. The output includes the following information:

`LINK` Refers to the link name to be used in the WiFi connection.

`ESSID` Refers to the Extended Service Set ID. The `ESSID` is the name of the WiFi network, which can be randomly named by the administrator of the specific wireless network.

`BSSID/IBSSID` Refers to the Basic Service Set ID (BSSID), which is a unique identifier for a particular ESSID. The BSSID is the 48-bit MAC address of the nearby access point that serves the network with a particular ESSID.

SEC	Refers to the type of security that is required to access the wireless network. The values are none, WEP, and WPA. For more information, see "Establishing Secure WiFi Communications" on page 144.
STRENGTH	Refers to the strength of the radio signals from the WiFi networks that are available at your location.
MODE	Refers to the version of the 802 .11 protocol that is run by the network. The modes are a, b, and g, or any combination of these modes.
SPEED	Refers to the speed (in megabits per second) of the particular network.

5. **Connect to a WiFi network by using either of the following methods:**

   - **Connect to an unsecured WiFi network with the strongest signal.**

     ```
     # dladm connect-wifi
     ```

   - **Connect to an unsecured network by specifying its ESSID.**

     ```
     # dladm connect-wifi -e ESSID
     ```

     For more information about using the `dladm connect-wifi` command, see "Establishing Secure WiFi Communications" on page 144 and the `dladm`(1M) man page.

6. **Check the status of the WiFi network to which the system is connected as follows:**

   ```
   # dladm show-wifi
   LINK       STATUS       ESSID       SEC     STRENGTH   MODE   SPEED
   net0       connected    ofc         none    very good   g      36Mb
   ```

   The previous output indicates that the system is connected to the `ofc` network. The `scan-wifi` output from Step 4 of this procedure indicated that `ofc` had the strongest signal of the available networks. The `dladm connect-wifi` command automatically chooses the WiFi network with the strongest signal, unless you explicitly specify a different wireless network.

7. **Configure an IP address for the interface by using either of the following methods:**

   - **Obtain an IP address from a DHCP server.**

     ```
     # ipadm create-addr -T dhcp interface
     ```

     If the WiFi network does not support DHCP, the following message is displayed:

     ```
     ipadm: interface: interface does not exist or cannot be managed using DHCP
     ```

- **Configure a static IP address.**

  ```
  # ipadm create-addr -a address interface
  ```

  Use this option if you have a dedicated IP address for the system.

8. **Access the Internet through the WiFi network in one of the following ways:**

   - **If the access point offers free service, you can run a browser or any application of your choice.**
   - **If the access point is on a commercial WiFi network that requires a fee, follow the instructions that are provided at that location.**

     Typically, you need to supply a key and a payment method for this option.

9. **End the session in one of the following ways:**

   - **Terminate the WiFi session but leave the system running.**

     ```
     # dladm disconnect-wifi
     ```

   - **Terminate a particular WiFi session when more than one session is currently running.**

     ```
     # dladm disconnect-wifi link
     ```

     where *link* represents the interface that is being used for the session.

   - **Cleanly shut down the system while the WiFi session is running.**

     ```
     # shutdown -g0 -i5
     ```

     You do not need to explicitly disconnect the WiFi session prior to shutting down the system.

Example 7-1 Connecting to a Specific WiFi Network

The following example combines the different steps that you would take to connect your Oracle Solaris system to a wireless network. The example also shows how you can force the system to connect to a specific and preferred wireless network instead of allowing the OS to randomly select the wireless network. In the following example, assume that you have the static IP address 10.192.16.3/24 assigned for use on your notebook PC.

```
# dladm show-phys
LINK            MEDIA               STATE    SPEED  DUPLEX    DEVICE
net0            Ethernet            up       1500   full      ath0
net1            Ethernet            up       1500   full      e1000g0
```

```
# ipadm create-ip net0
# ipadm show-if net0
IFNAME      CLASS        STATE     ACTIVE     OVER
lo0         loopback     ok        yes        --
net0        ip           ok        yes        --

# dladm scan-wifi
LINK       ESSID        BSSID/IBSSID       SEC     STRENGTH   MODE   SPEED
net0       wifi-a       00:0e:38:49:01:d0  none     weak       g      54Mb
net0       wifi-b       00:0e:38:49:02:f0  none     very weak  g      54Mb
net0       ofc-net      00:0d:ed:a5:47:e0  wep      very good  g      54Mb
net0       citinet      00:40:96:2a:56:b5  none     good       b      11Mb

# dladm connect-wifi -e citinet
# dladm show-wifi
LINK       STATUS        ESSID         SEC     STRENGTH   MODE   SPEED
net0       connected     citinet       none    good       g      11Mb

# ipadm create-addr -a 10.192.16.3/24 net0
ipadm: net0/v4
# ipadm show-addr net0
ADDROBJ           TYPE     STATE        ADDR
net0/v4           static   ok           10.192.16.3/24
```

Start a browser or another application to commence your work over the WiFi network.

```
# firefox
```

Terminate the session but leave the PC running.

```
# dladm disconnect-wifi
# dladm show-wifi
LINK       STATUS        ESSID         SEC     STRENGTH   MODE   SPEED
net0       disconnected   --            --       --       --       --
```

The output of `show-wifi` verifies that you have disconnected the `net0` link from the WiFi network.

▼ How to Monitor the WiFi Link

The following procedure describes how to monitor the status of a WiFi link through standard networking tools and also how to change a selected link property.

1. **Become an administrator.**
2. **Connect to a WiFi network, as described in "How to Connect to a WiFi Network" on page 140.**
3. **View the properties of the system's datalinks.**

```
# dladm show-linkprop link
```

4. **Set a fixed speed for the link.**

Caution - Oracle Solaris automatically selects the optimal speed for a WiFi connection. Modifying the initial speed of the link might diminish performance or prevent the establishment of certain WiFi connections.

You can modify the link speed to one of the possible values that is listed in the output of the `show-linkprop` subcommand, as shown in the following example:

```
# dladm set-linkprop -p speed=value link
```

5. **Check the packet flow over the link.**

```
# netstat -I net0 -i 5
input   net0      output       input  (Total)    output
packets errs  packets errs  colls  packets errs  packets errs  colls
317     0     106     0     0      2905    0     571     0     0
14      0     0       0     0      20      0     0       0     0
7       0     0       0     0      16      0     1       0     0
5       0     0       0     0      9       0     0       0     0
304     0     10      0     0      631     0     316     0     0
338     0     9       0     0      722     0     381     0     0
294     0     7       0     0      670     0     371     0     0
306     0     5       0     0      649     0     338     0     0
289     0     5       0     0      597     0     301     0     0
```

Example 7-2 Setting the Speed of a Link

The following example shows how you would set the speed of a link after you have connected to a WiFi network.

```
# dladm show-linkprop -p speed net0
LINK     PROPERTY        PERM VALUE        EFFECTIVE    DEFAULT   POSSIBLE
net0     speed           r-   25           0            0         1,2,5,6,9,11,12,18,24,36,48,54
# dladm set-linkprop -p speed=36 net0

# dladm show-linkprop -p speed net0
LINK     PROPERTY        PERM VALUE        EFFECTIVE    DEFAULT   POSSIBLE
net0     speed           r-   36           0            0         1,2,5,6,9,11,12,18,24,36,48,54
```

Establishing Secure WiFi Communications

Radio wave technology makes WiFi networks readily available and often freely accessible to users. As a result, connecting to a WiFi network can be an insecure undertaking.

The following types of WiFi connections are more secure:

- Connecting to a private restricted-access WiFi network.

Private networks, such as internal networks that are established by corporations or universities, restrict network access to users who can provide the correct security challenge. Potential users must supply a key during the connection sequence or log in to the network through a secure Virtual Private Network (VPN) application.

- Encrypting your connection to a WiFi network.

  You can encrypt communications between your system and a WiFi network by using a secure key. Your access point to the WiFi network must be a router that is in your home or office with a secure key-generating feature. Your system and the router establish and then share the key before creating the secure connection.

The `dladm` command can use a Wired Equivalent Privacy (WEP) or a Wi-Fi Protected Access (WPA) key for encrypting connections through an access point. The WEP protocol is defined in the IEEE 802.11 specifications for wireless connections. The WPA protocol is defined in the IEEE 802.11i specifications for wireless connections. Oracle Solaris supports versions 1 and 2 of the WPA standard. For more information about `dladm` command options that are related to WEP and WPA, refer to the `dladm`(1M) man page.

How to Set Up an Encrypted WiFi Network Connection by Specifying a WEP Key

The following procedure describes how to set up secure communications between a system and a router in the home. Many wireless and wired routers for the home have an encryption feature that is capable of generating a secure key.

Before You Begin If you are connecting to a home wireless network, make sure that you have configured your router and have generated a WEP key. Follow the router manufacturer's documentation for generating and saving the key configuration.

1. **Become an administrator.**

2. **Create a secure object that contains the WEP key as follows:**

   ```
   # dladm create-secobj -c wep keyname
   ```

   where *keyname* represents the name you want to give to the key.

3. **Supply the value for the WEP key to the secure object.**

   The `create-secobj` subcommand then runs a script that requests the value for the key.

   ```
   provide value for keyname: 5-or-13-byte key
   confirm value for keyname: Retype key
   ```

   This value is the key that was generated by the router. The script accepts either a 5–byte or 13–byte string, in ASCII or hexadecimal format for the key value.

4. **View the contents of the key that you just created.**

```
# dladm show-secobj
OBJECT              CLASS
keyname               wep
```

where *keyname* is the name of the secure object.

5. **Make an encrypted connection to the WiFi network.**

```
# dladm connect-wifi -e network -k keyname interface
```

6. **Verify that the connection is secure.**

```
# dladm show-wifi
LINK       STATUS        ESSID         SEC     STRENGTH   MODE   SPEED
net0       connected     wifi-1        wep     good       g      11Mb
```

In the previous output, the `wep` value that is located under the `SEC` column indicates that the WEP encryption for the connection is in place.

Example 7-3 Setting Up Encrypted WiFi Communications by Using a WEP Key

The following example assumes that you have already done the following:

- Followed the router manufacturer's documentation and created the WEP key.
- Saved the key so that you can use it to create the secure object on your system.

A secure object is created as follows:

```
# dladm create-secobj -c wep mykey
provide value for mykey: *****
confirm value for mkey: *****
```

When you supply the WEP key that is generated by the router, asterisks mask the value that you type.

The following command establishes an encrypted connection to the WiFi network `citinet` by using the secure object `mykey`.

```
# dladm show-secobj
OBJECT              CLASS
mykey               wep
# dladm connect-wifi -e citinet -k mykey net0
```

The following command verifies that you are connected to the `citinet` wireless network through a WEP encryption.

```
# dladm show-wifi
LINK       STATUS        ESSID         SEC     STRENGTH   MODE   SPEED
net0       connected     citinet       wep     good       g      36Mb
```

Administering Known WLANs in Reactive Mode

If you are using a reactive profile, you might also want to take advantage of the system's ability to automatically connect to *Known WLANs*. When you connect to a WLAN, information about that WLAN is preserved in a network profile of type `known-wlan`. You can also manually create `known-wlan` profiles by using the `netcfg` command. For overview information about Known WLANs, see "Description of a Known WLAN" on page 102 For task-related information, see "Creating Known WLANs" on page 122.

Administering Wireless Networks From the Desktop

By default, when wireless network connections are enabled, the reactive networking daemon (`nwamd`) attempts to connect to any available network in the Favorites list without asking, in the priority order in which those connections are listed. If no favorite networks are available, the Wireless Chooser dialog box opens. In this dialog box you can choose a wireless network to join.

You can also modify the way in which connections to wireless networks are attempted in the Wireless tab of the Network Preferences dialog's Connection Properties view of the Network Administration GUI. If required, you can also manually connect to a different wireless network by right-clicking the Network Status notification icon that is located on the desktop.

Tip - You can access the Connection Properties view for a selected network through the Network Preferences dialog. This dialog contains a drop-down list that is labeled Show. This list enables you to switch between views for a given network. In each view, there are different tasks that you can perform, as well as information about the selected network, which is specific to that view.

The following views exist for every network connection in each network profile that is on the system:

- Connection status
- Network profile
- Connection properties

For more information about profile-based network configuration, see Chapter 5, "About Administering Profile-Based Network Configuration in Oracle Solaris" and the GUI online help.

How to Join a Wireless Network

Wireless networks are joined by choosing the Join Wireless Network option that is available by right-clicking the Network Status notification icon. The Wireless Chooser dialog is where a list of wireless networks that are available to connect to is displayed.

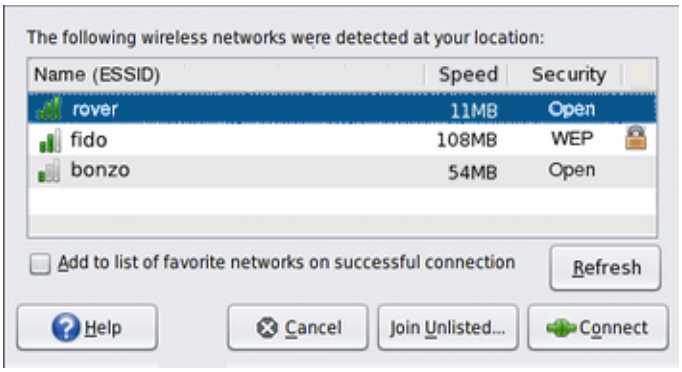
1. **To manually connect to a different wireless network, do one of the following:**

   - **Select an available wireless network from the Network Status notification icon's right-click menu.**

   - **Select the Join unlisted wireless network option from the Network Status notification's icon menu.**

     An unlisted wireless network is one that has been configured so that it does not broadcast its network name, yet is still available to join.

   - **Select an available wireless network from the Wireless Chooser dialog. This dialog is displayed automatically, when there is a choice of available wireless networks to join.**



2. **If the Join Wireless Network dialog opens, provide all of the necessary information for the wireless network that you have chosen.**

Managing Favorite Wireless Networks From the Desktop

By default, when you join a wireless network for the first time, a check box labeled, Add to List of Favorite Networks on Successful Connection, is displayed in the Join Wireless Network dialog.

To add the wireless network to your Favorites list (if the connection is successful) select this box. If you do not want the network to be added to your list of favorites, deselect the box. The box is selected by default.

To add a wireless network that is not currently available, or not currently broadcasting its network name to your Favorites list, go to the Wireless tab of the Connection Properties view, then click the Add button. To add the network, you will need to know its network name, security type, and security key.

Index

E

F

H

I

O

P

R

S

T

U

V

W