Connecting Systems Using Fixed Network Configuration in Oracle® Solaris 11.1

Part No: E28987–01
October 2012

ORACLE®

Copyright © 2011, 2012, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS. Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Ce logiciel et la documentation qui l'accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d'utilisation et de divulgation. Sauf disposition de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, breveter, transmettre, distribuer, exposer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d'interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu'elles soient exemptes d'erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Si ce logiciel, ou la documentation qui l'accompagne, est concédé sous licence au Gouvernement des Etats-Unis, ou à toute entité qui délivre la licence de ce logiciel ou l'utilise pour le compte du Gouvernement des Etats-Unis, la notice suivante s'applique:

U.S. GOVERNMENT END USERS. Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d'applications de gestion des informations. Ce logiciel ou matériel n'est pas conçu ni n'est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer des dommages corporels. Si vous utilisez ce logiciel ou matériel dans le cadre d'applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l'utilisation de ce logiciel ou matériel pour ce type d'applications.

Oracle et Java sont des marques déposées d'Oracle Corporation et/ou de ses affiliés. Tout autre nom mentionné peut correspondre à des marques appartenant à d'autres propriétaires qu'Oracle.

Intel et Intel Xeon sont des marques ou des marques déposées d'Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. AMD, Opteron, le logo AMD et le logo AMD Opteron sont des marques ou des marques déposées d'Advanced Micro Devices. UNIX est une marque déposée d'The Open Group.

Ce logiciel ou matériel et la documentation qui l'accompagne peuvent fournir des informations ou des liens donnant accès à des contenus, des produits et des services émanant de tiers. Oracle Corporation et ses affiliés déclinent toute responsabilité ou garantie expresse quant aux contenus, produits ou services émanant de tiers. En aucun cas, Oracle Corporation et ses affiliés ne sauraient être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'accès à des contenus, produits ou services tiers, ou à leur utilisation.

121119@25097

Contents

Preface

Welcome to *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*. This book is part of the series *Establishing An Oracle Solaris 11.1 Network* that cover basic topics and procedures to configure Oracle Solaris networks. This book assumes that you have already installed Oracle Solaris. You should be ready to configure your network or ready to configure any networking software that is required on your network.

Note – This Oracle Solaris release supports systems that use the SPARC and x86 families of processor architectures. The supported systems appear in the *Oracle Solaris OS: Hardware Compatibility Lists*. This document cites any implementation differences between the platform types.

For supported systems, see the *Oracle Solaris OS: Hardware Compatibility Lists*.

Who Should Use This Book

This book is intended for anyone responsible for administering systems that run Oracle Solaris, which are configured in a network. To use this book, you should have at least two years of UNIX system administration experience. Attending UNIX system administration training courses might be helpful.

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.

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Description	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for shells that are included in the Oracle Solaris OS. Note that the default system prompt that is displayed in command examples varies, depending on the Oracle Solaris release.

TABLE P–2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	`$`
Bash shell, Korn shell, and Bourne shell for superuser	`#`
C shell	`machine_name%`
C shell for superuser	`machine_name#`

CHAPTER 1

Overview of Fixed Network Configuration

On a system that runs Oracle Solaris 11, network configuration can either be reactive or fixed depending on the network configuration profile (NCP) that is active on the system. For an overview of reactive and fixed network configuration, see *Introduction to Oracle Solaris 11 Networking*. For detailed information about how to create and configure NCPs, see *Connecting Systems Using Reactive Network Configuration in Oracle Solaris 11.1*.

This chapter presents a general introduction to fixed network configuration and covers the following topics:

What Is Fixed Network Configuration?

In Oracle Solaris 11, network configuration is managed by network configuration profiles (NCPs). The type of NCP that is operative on a specific system determines that system's network configuration. If the NCP is reactive, network configuration on that system is dynamically implemented. If the NCP is fixed, network configuration is statically implemented.

Fixed network configuration refers to the configuration mode in which a specific network setup is instantiated on the system. Unlike in a reactive network configuration mode, the instantiated configuration in a fixed configuration mode remains unchanged regardless of changes in the system's network environment. If changes in that environment occur, such as addition of interfaces, you must manually reconfigure the system's network setup to have the system adopt to the new environment.

Note – Do not confuse fixed network configuration with simply configuring static IP addresses. In fixed network configuration, you can assign a DHCP address to an interface. Likewise, in reactive network configuration, you can create NCPs where interfaces are configured with static IP addresses. Thus, fixed network configuration has a wider scope and specifically refers to the ability of the system's network configuration to change according to changes in the system's environment.

The following table presents a comparison between the two modes of network configuration.

Features	Reactive Network Configuration	Fixed Network Configuration
Automatically adapts to changes in system's network environment	Supported by means of multiple NCPs that can be configured	Not supported; requires manual reconfiguration as needed
Type of NCP operative on the system	Reactive (`Automatic`, or some other user-created NCP)	Fixed (`DefaultFixed`)
Multiple NCPs	Supported (but only one NCP can be active at a time)	Not supported
User created NCPs	Supported	Only one fixed NCP (`DefaultFixed`) exists, which is generated by the system. However, the contents of `DefaultFixed` is entirely determined by the user.

The following sections describe in more detail profile-managed network configuration and the tools used for network configuration.

Highlights of Profile-Managed Network Configuration

In Oracle Solaris 11, network configuration is based on profiles. A system's network configuration is managed by an NCP and a corresponding Location profile. For an introduction to profile-managed network configuration, see "Network Configuration Profiles" in *Introduction to Oracle Solaris 11 Networking*. For details about NCPs, see *Connecting Systems Using Reactive Network Configuration in Oracle Solaris 11.1*.

Note – For network configuration, the principal profile types are NCPs, Location profiles, external network modifiers (ENMs), and wireless local area networks (WLANs). Of these types, the main profile is the NCP. Throughout this documentation, unless specified otherwise, the term *profile* refers to the NCP.

The highlights of profile-based network configuration follow:

- Only one pair of NCP and location profiles can be active at one time to manage a system's network configuration. All other existing NCPs on the system are non-operational.
- The active NCP can be either *reactive* or *fixed*. With a reactive profile, the network configuration is monitored to adapt to changes in the system's network environment. With a fixed profile, the network configuration is instantiated but not monitored.
- If the active NCP is reactive, the system's networking configuration is adaptive. If the active NCP is fixed, the system's networking configuration is constant.
- The values of the different properties of an NCP constitute a policy that governs how the profile manages the network configuration.
- Changes to the NCP's properties are immediately implemented as new property values. These new values become part of the profile's policy that manages the network configuration.

If your system is configured for fixed networking, then the active NCP that manages its network configuration is `DefaultFixed`. This profile is generated by the OS and is the only fixed profile on the system. A system does not support multiple fixed profiles.

The properties of the `DefaultFixed` NCP reflect the persistent configuration that is created or modified while `DefaultFixed` NCP is active.

Network Configuration Tools

In Oracle Solaris 11, four network commands are available to configure the network:

- `netcfg` command
- `netadm` command
- `dladm` command
- `ipadm` command

The `netcfg` and `netadm` commands are used to administer reactive network configuration on the system. You use the `netcfg` command to create and configure profiles that implement reactive network configuration: NCPs, Location profiles, ENMs, and WLANs. However, on a system with fixed network configuration, you can use the `netcfg` command only to view the `DefaultFixed` profile. The `netadm` command is used to administer all the profiles on the system, particularly to list the system's network profiles as well as to replace one active NCP with another.

The `dladm` and `ipadm` commands are used to configure datalinks and IP interfaces respectively. The commands create persistent configurations and are applied to the profile that is active on the system when the commands are used.

For example, if a datalink `net0` is configured with a specific maximum transmission unit (MTU) of 1200, and the active NCP is `Automatic`, then this MTU value becomes persistent for `net0` in the `Automatic` NCP. Suppose then that you activated a second NCP called `myncp`. If you issue the `dladm` command to set the MTU with a different value, then that value would be applied to `myncp`. Thus, net0 can have different MTU values in different profiles. Thus, the `dladm` and `ipadm` commands can also be used to indirectly configure profiles.

When you configure datalinks and IP interfaces with these the `dladm` or `ipadm` commands, be aware of the following scopes of their use:

- The two commands configure only the datalinks and IP interfaces for the active profile. To configure other properties of the profile, such as setting default routes, you use the `netcfg` command to configure the profile's property that refers to default routes. Or, you use the `routeadm` command that directly sets default routes on the system's routing table. In the latter case, the configuration applies to whichever profile is active on the system.
- You can use the `dladm` and `ipadm` commands on any reactive profile, provided that the profile is active. However, you cannot use the `netcfg` command to configure the `DefaultFixed` profile, which is the system's only fixed profile. You can only use the `netadm` and `netcfg` commands to view the properties of the `DefaultFixed` profile, but not to configure them.

The `dladm` and `ipadm` commands are effective on the active profile, either a reactive profile or a fixed profile. Consequently, before you use these commands, you must make sure of the following:

- Know which profile is active on the system to ensure that you make changes to the correct target profile.
- Know whether the target profile is reactive or fixed to avoid causing unexpected configuration behavior after using the commands. A reactive profile manages the network configuration differently than a fixed profile. Accordingly, the behavior of the two profiles also differs when changes are implemented.

The next sections describe the `dladm` and `ipadm` commands in detail.

`dladm` Command

Use the `dladm` command to configure datalinks. You can customize datalink properties by using the `dladm` command, provided that the link's network driver has been converted to the GLDv3 driver configuration framework, such as `e1000g`. To confirm whether your specific driver supports this feature, refer to the driver's man page.

The full implementation of the GLDv3 driver configuration framework has enhanced the configuration of network interface card (NIC) drivers in the following ways:

- Only a single command interface, the `dladm` command, is needed to configure network driver properties.
- A uniform syntax is used regardless of the properties: `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 even after you reboot the system.

To avail of these advantages when you configure datalinks, you should use `dladm` as the configuration tool instead of the customary tools in previous releases, such as the `ndd` command.

For more details about the `dladm` command, refer to the `dladm`(1M) man page. For a list of subcommands to use with the `dladm` command, type the following:

```
# dladm help
The following subcommands are supported:
Bridge           : add-bridge     create-bridge   delete-bridge
                   modify-bridge  remove-bridge   show-bridge
Etherstub        : create-etherstub delete-etherstub show-etherstub
IB               : create-part    delete-part     show-ib         show-part
IP tunnel        : create-iptun   delete-iptun    modify-iptun    show-iptun
Link Aggregation: add-aggr        create-aggr     delete-aggr
                   modify-aggr    remove-aggr     show-aggr
Link             : rename-link    reset-linkprop  set-linkprop
                   show-link      show-linkprop
Secure Object    : create-secobj  delete-secobj   show-secobj
VLAN             : create-vlan    delete-vlan     modify-vlan     show-vlan
VNIC             : create-vnic    delete-vnic     modify-vnic     show-vnic
Wifi             : connect-wifi   disconnect-wifi scan-wifi       show-wifi
Miscellaneous    : delete-phys    show-ether      show-phys       show-usage
For more info, run: dladm help <subcommand>.
```

To use `dladm` command on datalinks, see Chapter 3, "Working With Datalinks."

`ipadm` Command

Advances in Oracle Solaris have surpassed the capabilities of traditional tools to efficiently administer various aspects of network configuration. The `ifconfig` command, for example, has been the customary tool to configure network interfaces. However, this command does not implement persistent configuration. Over time, `ifconfig` has undergone enhancements for added capabilities in network administration. However, as a consequence, the command has become complex and confusing to use.

Another issue with interface configuration and administration is the absence of simple tools to administer TCP/IP properties or tunables. The `ndd` command has been the prescribed customization tool for this purpose. However, like the `ifconfig` command, `ndd` does not implement persistent configuration. Previously, persistent configuration could be simulated for a network scenario by editing the boot scripts. With the introduction of service management facility (SMF) in Oracle Solaris, using such workarounds can become risky because of the complexities of managing SMF dependencies, particularly in light of upgrades to an Oracle Solaris installation.

The `ipadm` command is introduced to eventually replace the `ifconfig` command for interface configuration. The command also replaces the `ndd` command to configure protocol properties.

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

- It manages IP interfaces and IP addresses more efficiently by being the tool uniquely for IP interface administration, unlike the `ifconfig` command, which is used for purposes other than interface configuration.
- It implements persistent interface and address configuration.

For a list of `ifconfig` options and their equivalent `ipadm` subcommands, see Appendix A, "Comparison Map: `ifconfig` and `ipadm` Commands."

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

- It can set temporary or persistent properties for these protocols: IP, Address Resolution Protocol (ARP), Stream Control Transmission Protocol (SCTP), and Internet Control Messaging Protocol (ICMP), as well as upper layer protocols such as TCP and User Datagram Protocol (UDP).
- It provides information about each TCP/IP property, such as a property's current and default value, as well as the range of possible values. Thus, debugging information is more easily obtained.
- It also follows a consistent command syntax and is therefore easier to use.

For a list of `ndd` options and their equivalent `ipadm` subcommands, see Appendix B, "Comparison Map: `ndd` and `ipadm` Commands."

For more details about the `ipadm` command, refer to the `ipadm`(1M) man page. For a list of subcommands to use with the `ipadm`, type the following:

```
# ipadm help
The following subcommands are supported:
Address                 : create-addr    delete-addr     disable-addr
                          down-addr      enable-addr     refresh-addr
                          reset-addrprop set-addrprop    show-addr
                          show-addrprop  up-addr
Interface               : disable-if     enable-if       reset-ifprop
                          set-ifprop     show-if         show-ifprop
IP interface            : create-ip      delete-ip
IPMP interface          : add-ipmp       create-ipmp     delete-ipmp
                          remove-ipmp
Protocol property       : reset-prop     set-prop        show-prop
VNI interface           : create-vni     delete-vni
For more info, run: ipadm help <subcommand>.
```

CHAPTER 2

Configuring a System for the Network

This chapter provides procedures you follow to configure an IP interface on a system that uses fixed network configuration. The following topics are discussed:

Configuring the Network (Task Map)

This section describes basic configuration procedures for an IP interface. The following table describes configuration tasks and maps these tasks to their corresponding procedures.

Task	Description	For Instructions
Configure a system to support unique MAC addresses.	Configures a SPARC based system to allow unique MAC addresses for interfaces.	"SPARC: How to Ensure That the MAC Address of Each Interface Is Unique" on page 16
Determine which NCP is active on the system	Displays the active NCP on the system and enables `DefaultFixed`.	"How to Change the Active NCP On the System" on page 17
Perform basic IP interface configuration by using the `ipadm` command.	Creates an IP interface and assigns valid IP addresses, either static or DHCP, to the interface.	"How to Configure an IP Interface" on page 18
Customize datalinks.	Customize datalinks further by setting link properties.	"Customizing Datalink Properties" on page 28
Customize IP interfaces.	Customizes IP interfaces further by setting interface properties.	"Setting IP Interface Properties" on page 41
Customize IP addresses.	Customizes IP addresses further by setting address properties.	"Setting IP Address Properties" on page 42

Task	Description	For Instructions
Customize protocols.	Customizes protocols further by setting protocol properties.	"Setting TCP/IP Protocol Properties" on page 43
Configure a wireless network.	Connect a laptop to the network using wireless networking.	Chapter 5, "Configuring Wireless Networking on Laptops Running Oracle Solaris"

▼ 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 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 their unique MAC address. The next procedure explains how to use the `eeprom` command to check the current value of `local-mac-address?` and change it, if necessary.

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

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

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

In the example, the response to the `eeprom` command, `local-mac-address?=false`, indicates that all interfaces do use the system-wide MAC address. The value of `local-mac-address?=false` must be changed to `local-mac-address?=true` before the interfaces can become members of an IPMP group. You should also make this change for link aggregations.

3 If necessary, change the value of `local-mac-address?` as follows:

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

When you reboot the system in Step 6, the interfaces with 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 the interfaces on the system.**

Look for cases where multiple interfaces have the same MAC address. In this 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             DEFAULT           POSSIBLE
net0   mac-address    rw   8:0:20:0:0:1      8:0:20:0:0:1      --
net1   mac-address    rw   8:0:20:0:0:1      8:0:20:0:0:1      --
net3   mac-address    rw   0:14:4f:45:c:2d   0:14:4f:45:c:2d   --
```

Note – Continue to the next step only if two or more network interfaces have the same MAC address. Otherwise, proceed to the final step.

---

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

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

In the example in the previous step, 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
```

Refer to the `dladm(1M)` man page for details about this command.

6 **Reboot the system.**

▼ How to Change the Active NCP On the System

The type of NCP enabled on the system determines whether the system's network configuration is reactive or fixed. The system with reactive configuration behaves differently than with fixed network configuration. All the procedures in this book create persistent configurations which are applied to the active NCP. Therefore, before performing any procedure, you must know which NCP is active to apply the configuration to the correct profile. Thus, the system's network configuration behaves as you expect after completing the procedures.

1 **List the profiles on the system.**

```
# netadm list
TYPE        PROFILE        STATE
ncp         DefaultFixed   online
ncp         Automatic      disabled
loc         Automatic      offline
loc         NoNet          offline
loc         User           offline
loc         DefaultFixed   online
```

The profile whose status is listed as online is the active NCP on the system.

For more detailed information about the NCPs on the system, use the `-x` option with the `netadm` command.

```
netadm list -x
TYPE        PROFILE        STATE          AUXILIARY STATE
ncp         DefaultFixed   online         active
ncp         Automatic      disabled       disabled by administrator
loc         Automatic      offline        conditions for activation are unmet
loc         NoNet          offline        conditions for activation are unmet
loc         User           offline        conditions for activation are unmet
loc         DefaultFixed   online         active
```

2 To switch between profile types, for example from a reactive profile to a fixed profile, type the following command:

`# netadm enable -p ncp` *NCP-name*

where *NCP-name* is the name of a type of NCP.

For example, suppose that your system's network configuration is reactive. If you want the configurations that are created by the procedures in this book to apply to the `DefaultFixed` NCP, you would type the following:

```
# netadm enable -p ncp defaultfixed
```

Caution – When you switch active profiles, the existing network configuration is removed, and a new configuration is created. Any persistent configurations that were implemented on a previously active NCP are excluded in the new active NCP.

How to Configure an IP Interface

The following procedure provides the basic steps that you use to configure a system's IP 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.

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

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 that you create when you perform network configuration. To create this interface class, use the `create-ip` subcommand.
- STREAMS virtual network interface driver (VNI interface). To create this interface class, use the `create-vni` subcommand. For more information about VNI devices or interfaces, see the `vni(7d)` man page.
- IPMP interface. This interface 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 5, "Introduction to IPMP," in *Managing Oracle Solaris 11.1 Network Performance*.

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 know the datalinks on the system, use the `dladm show-link` command.

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

- To configure a static address, type the following:

  `# 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, as well as tunnel configuration, see Chapter 6, "Configuring IP Tunnels," in *Configuring and Administering Oracle Solaris 11.1 Networks*.

  For a numeric IP address, use CIDR notation. If you do not use CIDR notation, the netmask is computed according to the sequence listed for `netmask` in the `name-service/switch` service 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, the creation of IP addresses precedes naming services being brought online. Therefore, you must ensure that any host name that is used in the network configuration must be 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 create the address object name automatically, specify the interface name for *interface*. Otherwise, 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. Multiple addresses on the same interface have alphabetic letters appended to the address object names, such as `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.

- To configure a non-static address, type the following:

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

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

  For a fuller explanation about `[`*interface* `|` *addrobj*`]`, 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 *interface*, information for all interfaces on the system is displayed.

For more information about the output of the `ipadm show-*` subcommand, see "Monitoring IP Interfaces and Addresses" on page 50.

5 If you are configuring a static IP address that uses a hostname, add entries for the IP address in 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 route(1M) and routeadm(1M) man pages. See also "Routing Tables and Routing Types" in *Configuring and Administering Oracle Solaris 11.1 Networks*.

Example 2–1 Configuring a Network Interface With a Static IP Address

This example explains how to configure an interface with a static IP address. The example begins with enabling the `DefaultFixed` NCP on the system to allow you to use the `dladm` and `ipadm` commands for fixed network configuration.

```
# 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     BRIDGE   OVER
net3     phys      1500    up        --       --

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

# ipadm
NAME      CLASS/TYPE     STATE     UNDER     ADDR
lo0       loopback       ok        --        --
   l0/v4       static    ok        --        127.0.0.1/8
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    campus01
```

```
# route -p add default 192.168.84.1
# netstat -r
Routing Table: IPv4
  Destination           Gateway           Flags  Ref     Use      Interface
-------------------- -------------------- ----- ----- ---------- ---------
default              some.machine.com          UG       2       10466
192.168.84.0         192.168.84.3         U        3        1810  net3
localhost            localhost            UH       2          12  lo0

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

Note that if `campus01` is already defined in the `/etc/hosts` file, you can use that host name when assigning the following address:

```
# ipadm create-addr -a campus01 net3
ipadm: net3/v4
```

Example 2–2 Automatically Configuring a Network Interface With an IP Address

In this example, the IP interface is configured to receive its address from a DHCP server.

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

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

# ipadm create-ip net3
# ipadm create-addr -T dhcp net3
ipadm: 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
```

Other Network Configuration and Administration Tasks

This book describes basic network configuration that connects your system to the network. Specifically, the information focuses on configuration of the system's datalinks and interfaces. Other network configuration and administration tasks can be performed that are described in other networking books. Assuming that your system is configured for fixed network configuration, you can refer to the following books for these other tasks:

- To configure systems as routers, network configuration servers, and so on, see *Configuring and Administering Oracle Solaris 11.1 Networks*.
- To perform advance datalink and IP interface configuration, see *Managing Oracle Solaris 11.1 Network Performance*.
- To build on the basic configuration and improve network performance, such as configuring link aggregations, IPMP groups, and so on, see *Managing Oracle Solaris 11.1 Network Performance*.
- To establish security for your network, see *Securing the Network in Oracle Solaris 11.1*.
- To implement network virtualization, see *Using Virtual Networks in Oracle Solaris 11.1*.

Other books that specialize on specific networking areas, such as DHCP, name services, and so on are also available in the library.

CHAPTER 3

Working With Datalinks

This chapter discusses the `dladm` command and explains how to use the command on datalinks to display their current configuration, change the default values of their properties, or delete datalinks from the system. The following topics are discussed:

Basic `dladm` Commands

This section describes basic `dladm` commands that you might regularly use on the system's datalinks. More `dladm` subcommands are supported than those listed in this section. For other subcommands, see the `dladm`(1M) man page.

Note – Except for the `dladm` subcommands that display datalink information, all other subcommands first require the removal of any existing interface configuration over the datalink. To remove IP interface configuration, see "Removing an IP Interface Configuration (`ipadm delete-ip`)" on page 39.

Displaying General Information About Datalinks (`dladm`)

If used by itself, the `dladm` command displays general information about the system's datalinks, including their class, state, and 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
```

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

For information about link aggregations and VLANs, see *Managing Oracle Solaris 11.1 Network Performance*. For information about VNICs, see *Using Virtual Networks in Oracle Solaris 11.1*.

Displaying a System's Datalinks (`dladm show-link`)

Use `dladm show-link` to display the datalinks on a system. A system has as many datalinks as installed NICs. You can use options with this command to customize the information you obtain. For example, using the `-P` option includes persistent configuration information about the datalinks. Based on the information 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 issue the command, information similar to the following is displayed:

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

This example shows that a 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 Physical Attributes of Datalinks (`dladm show-phys`)

Use `dladm show-phys` 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 similar to the following:

```
# 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 output shows, among other details, the physical NICs with which the datalinks with generic link names are associated. For example, `net0` is the datalink name of the NIC `e1000g0`. To see 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.

Another useful option for the command is `-L`, which shows 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
```

Deleting a Datalink (`dladm delete-phys`)

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

Removing a datalink is only loosely connected to the removal of a physical NIC. For example, a physical NIC is removed from the system. The datalink configuration associated with that NIC remains because the software layer is no longer bound to the hardware layer, as described in "Network Stack in Oracle Solaris" in *Introduction to Oracle Solaris 11 Networking*. 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.

Renaming a Datalink (`dladm rename-link`)

Use `dladm rename-link` to rename a datalink. On an Oracle Solaris 11 system, the OS automatically provides generic names to all datalinks. Generic datalink names are described in "Default Generic Link Names" in *Introduction to Oracle Solaris 11 Networking*.

By default, these generic names use the naming format `net`*n*, such as `net0`, `net1`, `net2`, and so on. Because the OS manages the 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 Switch Primary Interfaces on a System" on page 34.

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.

Three `dladm` subcommands are used for datalink properties:

- `dladm show-linkprop [-p` *property*`] [datalink]` displays the properties of a datalink and their current values. If you do not use the `-p` *property* option, then all the properties of a datalink are listed. If you do not specify a datalink, then all the properties of all datalinks are listed.
- `dladm set-linkprop -p` *property*=*value datalink* assigns a value to the datalink's property.
- `dladm reset-linkprop -p` *property datalink* resets the specific property to its default value.

Overview of Datalink Properties

Datalink properties that can be customized 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* that can be applied to any driver of the given media type such as link speed, autonegotiation for Ethernet, or the maximum transmission unit (MTU) size that can be applied to all datalink drivers.
- *Private properties* that 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 you to change specific property values. For example, a NIC might be communicating with an old switch that does not properly perform autonegotiation. Or, a switch might have been configured to support Jumbo frames. Or, driver specific properties that regulate packet transmission or packet receiving might need to be modified for the specific driver. The following sections describe selected properties and explains how to change their values to function in your network environment.

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 through the network, the default value is insufficient. Support for Jumbo frames requires the MTU size to be at least 9000.

To change the MTU size from its default value, type the following command:

```
# 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 the steps to enable support for Jumbo frames. The example assumes that you have already removed any existing IP interface configuration over the datalink.

```
# dladm show-linkprop -p mtu net1
LINK     PROPERTY     VALUE     DEFAULT     POSSIBLE
net1     mtu          1500      1500        --
# dladm set-linkprop -p mtu=9000 net1
# dladm show-link web1
LINK     CLASS     MTU      STATE     BRIDGE     OVER
web1     phys      9000     up        --         --
```

Modifying Link Speed Parameters

Most network setups consist of a combination of systems with varying speed capabilities. Each system advertises speed capabilities to other systems in 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 the *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 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 autonegotiation 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 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
```

The command switches off the advertisement of the gigabit capabilities of the system for full-duplex capability and half-duplex capability.

To display the new values of these properties, use the `dladm show-linkprop` command.

```
# 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 on to 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` subcommand.

A separate `autopush` command can also be used to push modules on to the datalink's stream on a per-driver basis. The 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 becomes lost as well.

Therefore, the `dladm` command is a preferable tool for this purpose than the `autopush` command. If both per-driver and per-link types of `autoputsh` configuration exist for a specific datalink, the per-link information that is set with `dladm set-linkprop` is used, and the per-driver information is ignored.

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

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

Setting the e1000g Driver to Use Direct Memory Access Binding

This section and the following section show how to configure private properties. Both sections apply to properties specific to the `e1000g` driver. However, the general information in these sections applies when you configure private properties of other NIC drivers.

Bulk traffic, such as file transfers, normally involves the negotiation of large packets across the network. In such cases, you can obtain better performance from the `e1000g` driver by configuring it to automatically use direct memory access (DMA) binding, where a threshold is defined for packet fragment sizes. If a fragment size surpasses the threshold, then DMA binding is used for transmitting the packets. If a fragment size is within the threshold, then `bcopy` mode is used, where the fragment data is copied to the preallocated transmit buffer.

```
# dladm set-linkprop -p _tx_bcopy_threshold=value  datalink
```

For this property, the valid values for the threshold range from 60 through 2048.

Note – All datalinks are automatically named with generic names. You must ensure that this private property is configured on the datalink whose underlying NIC is `e1000g`. Use `dladm show-phys` to verify before setting the property.

As with configuring public properties, any IP interface must also be deleted before private property values can be modified.

You might perform steps similar to the following:

```
# dladm show-phys
LINK   MEDIA      STATE   SPEED   DUPLEX   DEVICE
net0   Ethernet   up      100Mb    full     nge0
net1   Ethernet   up      100Mb    full     e1000g0

# dladm set-linkprop -p _tx_bcopy_threshold=1024 net1
```

Manually Setting the Interrupt Rate

Properties that regulate the rate at which interrupts are delivered by the `e1000g` driver also affect network and system performance. Typically network packets are delivered to the upper layer of the stack by generating an interrupt for every packet. In turn the interrupt rate, by default, is automatically adjusted by the GLD layer in the kernel. However, this mode might not be desirable in all network traffic conditions. For a discussion of this issue, refer to this document (`http://www.stanford.edu/class/cs240/readings/mogul.pdf`) that was presented at the USENIX technical conference in 1996. Thus, in certain circumstances, setting the interrupt rate manually becomes necessary to obtain better performance.

To define the interrupt rate, you set the following properties:

- `_intr_throttling_rate` determines the delay between interrupt assertions regardless of network traffic conditions.
- `_intr_adaptive` determines whether automatic tuning of the interrupt throttling rate is enabled. By default, this property is enabled.

You first turn off the automatic tuning of the interrupt throttling rate. Then, you manually set the interrupt throttling rate property.

Suppose you have an x86 based system with an `e1000g` NIC whose interrupt throttling rate needs to be modified. Suppose further that the datalink name of `e1000g0` is `net1`. You would type the following commands.

```
# dladm set-linkprop -p _intr_adaptive=0 net1
# dladm set-linkprop -p _intr-throttling_rate=1024 net1
```

Obtaining Status Information About 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 (`dladm show-linkprop`)

This method is explained in "Customizing Datalink Properties" on page 28. To display a complete list of datalink properties, type the command without specifying a property. For example:

```
# dladm show-linkprop net1
LINK      PROPERTY              VALUE       DEFAULT     POSSIBLE
net1      speed                 1000        --          --
net1      autopush              --          --          --
net1      zone                  --          --          --
net1      duplex                half        --          half,full
net1      state                 unknown     up          up,down
net1      adv_autoneg_cap       1           1           1,0
net1      mtu                   1500        1500        --
```

```
net1     flowctrl            no          bi          no,tx,rx,bi
net1     adv_1000fdx_cap     1           1           1,0
net1     en_1000fdx_cap      1           1           1,0
net1     adv_1000hdx_cap     1           1           1,0
net1     en_1000hdx_cap      1           1           1,0
net1     adv_100fdx_cap      0           0           1,0
net1     en_100fdx_cap       0           0           1,0
net1     adv_100hdx_cap      0           0           1,0
net1     en_100hdx_cap       0           0           1,0
net1     adv_10fdx_cap       0           0           1,0
net1     en_10fdx_cap        0           0           1,0
net1     adv_10hdx_cap       0           0           1,0
net1     en_10hdx_cap        0           0           1,0
```

Displaying Ethernet Property Values (`dladm show-ether`)

If no options are used with the `dladm show-ether` command, then only current Ethernet property values of the datalink are displayed. To obtain more information beyond what is provided by default, use the `-x` option. The following is an example of how the command is used:

```
# 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 explains the displayed information in the preceding example:

- For the Ethernet device's current state, the link is up and functioning at 1 gigabits per second at full duplex. Its autonegotiation 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 shown in the first row of the output.
- Subsequent rows 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.

Other Configuration Tasks With the `dladm` Command

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

▼ How to Switch Primary Interfaces on a System

Changing a system's primary interface is a case where you rename datalinks. The following procedure is based on the following system configuration:

- The system has two datalinks: `net0` and `net1`.
- The underlying NICs are `e1000g0` and `nge0`, respectively.
- An IP interface is configured over `net0`. The IP interface always takes the name of the underlying datalink.

The system's primary interface is `net0` based on its instance number of zero (`0`). The primary interface is configured over `e1000g0`. The following steps guide you to make the datalink configuration over `nge0` to become the configuration of the primary interface.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Display the physical attributes of the system's datalinks.**

```
# dladm show-phys
```

3 **Delete the primary IP interface.**

```
# ipadm delete-ip interface
```

Note – For more information about the `ipadm` command, see Chapter 4, "Working With IP Interfaces," as well as the `ipadm(1M)` man page.

---

4 **Replace the name of the primary link with a name that is not used by other datalinks on the system.**

```
# dladm rename-link primary-link unused-name
```

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

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

Example 3–1 Switching the Primary Interface

The following example combines all the steps in the procedure to change the primary interface on the system. At the end of the example, the primary interface configured over `e1000g0` is replaced the interface configured over `nge0`. After you have switched the primary link to a different NIC, you can configure an interface over the new NIC's datalink.

```
# dladm show-phys
LINK   MEDIA      STATE   SPEED   DUPLEX   DEVICE
net0   Ethernet   up      100Mb    full     e1000g0
net1   Ethernet   up      100Mb    full     nge0

# ipadm delete-ip net0
# dladm rename-link net0 oldnet0
# dladm rename-link net1 net0

# ipadm create-ip net0
# ipadm create-addr -a 192.168.10.10/24 net0
ipadm: net0/v4

# dladm show-phys
LINK      MEDIA      STATE   SPEED   DUPLEX   DEVICE
oldnet0   Ethernet   up      1000    full     e1000g0
net0      Ethernet   up      1000    full     nge0
```

▼ How to Replace a Network Interface Card With Dynamic Reconfiguration

This procedure applies only to systems that support dynamic reconfiguration (DR). It specifically refers to configuration steps after DR s completed. In Oracle Solaris 11, you no longer need to reconfigure your network links after you complete DR. Instead, you just transfer the link configurations of the removed NIC to the replacement NIC.

The procedure does not detail the steps to perform DR itself. Consult your system documentation for these steps.

For an introduction to DR, see Chapter 4, "Dynamically Configuring Devices (Tasks)," in *Oracle Solaris 11.1 Administration: Devices and File Systems*.

Before You Begin Procedures to perform DR vary with the type of system. Make sure that you complete the following 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.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **(Optional) Display information about 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 `dladm show-phys -L`, refer to the `dladm`(1M) man page.

3 **Perform the DR as detailed in your system's documentation.**

Consult your system's DR documentation to perform this step.

After you have installed the replacement NIC, proceed to the next step.

4 **Perform one of the following steps depending on the circumstance that applies:**

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

  With the new NIC using the same location that the old NIC previously occupied, the new NIC inherits the link name and 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, type:

  ```
  # 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 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 was removed, and 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`. For the datalink of the new NIC to inherit the datalink configuration of the old NIC, you would type:

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

5 Complete the DR process by enabling the new NIC's resources to become available for use by Oracle Solaris.

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 `dladm show-phys` or `dladm show-link` to show information about the datalinks.

Example 3–2 Performing Dynamic Reconfiguration by Installing a New Network Card

This 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
```

The administrator performs the DR-specific steps such as using `cfgadm` to remove `bge` and then installing `e1000g` in its place. After the card is installed, the datalink of `e1000g0` automatically assumes the name `net0` and inherits the link configurations.

```
# 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     ---
```

CHAPTER 4

Working With IP Interfaces

This chapter discusses the ipadm command and explains how the command is used on IP interfaces. For an overview of the ipadm command and its benefits, see "ipadm Command" on page 13.

The following topics are discussed:

Basic ipadm Commands

In "How to Configure an IP Interface" on page 18, the three principal ipadm subcommands were introduced:

- ipadm
- ipadm create-ip
- ipadm create-addr

This section describes other selected uses of the ipadm command on IP interfaces. The list is not exhaustive. For a complete description of the ipadm command and all possible subcommands and options, see the ipadm(1M) man page.

Removing an IP Interface Configuration (ipadm delete-ip)

Use this command to remove a configured IP interface over a datalink. This command is particularly important when you are preforming certain datalink configurations. For example, renaming a datalink fails if IP interfaces are configured over that datalink. You must issue the `ipadm delete-ip` first before you rename the datalink.

Typically, this command is used together with other `ipadm` and `dladm` subcommands, such as changing the system's primary interface. This task would require you to delete the interface, rename the link, and then reconfigure the interface over the renamed datalink. The sequence is as follows:

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

See the example for changing the primary interface on "Renaming a Datalink (dladm rename-link)" on page 27. To reconfigure an IP interface after the datalink has been renamed, see "How to Configure an IP Interface" on page 18.

Disabling an IP Interface Configuration (`ipadm disable-ip`)

By default, an IP interface is flagged as `UP` and becomes part of the active configuration when you create the interface with `ipadm create-ip`. You can remove the interface from active configuration without destroying its configuration by using the `ipadm disable-ip` subcommand. This command flags the specific interface as `DOWN`.

```
# ipadm disable-ip interface
```

To make the IP interface operational and its flag to be `UP`, you would type:

```
# ipadm enable-ip interface
```

Tip – To display the status of interfaces, use `ipadm`. See "Obtaining Information About IP Interfaces" on page 51

Removing an Interface's Address (`ipadm delete-addr`)

This command deletes a specific address configuration of an IP interface. This command is useful when you want to change the IP address of a specific interface. You must remove the original address configuration before assigning a new address configuration. You would perform the following general steps:

```
# ipadm delete-addr addrobj
# ipadm create-addr parameters
```

For an example of creating an IP address for an interface, see "How to Configure an IP Interface" on page 18.

Note – An interface can have multiple 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` subcommand to display the interface addresses on the system. For an explanation of the address object, see "How to Configure an IP Interface" on page 18. For more information about displaying addresses, see "Obtaining Information About IP Addresses" on page 53

Setting IP Interface Properties

This section explains how to use the `ipadm` command to set selected IP interface properties.

IP interfaces, like datalinks, have properties that you can customize for your specific network environment. For each interface, two sets of properties exist, one set for the IPv4 and the other set for the IPv6 protocols. Some properties, such as MTU, 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 configured over that link. Further, you can have different MTU values that apply to IPv4 and IPv6 packets that traverse that IP interface.

Three `ipadm` subcommands are used to set IP interface properties:

- The `ipadm show-ifprop -p` *property interface* subcommand displays the properties of an IP interface and their current values. If you do not use the `-p` *property* option, then all the properties of the IP interface are listed. If you do not specify an IP interface, then all the properties of all IP interfaces are listed.
- The `ipadm set-ifprop -p` *property*=*value interface* subcommand assigns a value to the IP interface's property.
- The `ipadm reset-ifprop -p` *property interface* subcommand resets the specific property to its default values.

Enabling Packet Forwarding

In a network, a host can receive data packets that are destined for another host system. By enabling packet forwarding in the receiving local system, that system can forward the data packet to the destination host. By default, IP forwarding is disabled.

Packet forwarding is managed by a property that can be set on both IP interfaces and on the TCP/IP protocol. If you want to be selective in how packets are forwarded, then you enable packet forwarding on the IP interface. For example, you might have a system that has multiple

NICs. Some NICs are connected to the external network, while other NICs are connected to the private network. You would therefore enable packet forwarding only on some of the interfaces, rather than on all 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" on page 44.

Note – The `forwarding` property of either IP interfaces or protocols is not 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.

To enable packet forwarding on the IP interface, use the following command:

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

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

The following is an example of 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
...
```

Setting IP Address Properties

The `ipadm` command enables you to set IP address-specific properties after these addresses are assigned to interfaces. By setting these properties, you can determine the following:

- The 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

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

- The `ipadm show-addrprop [-p` *property*`] [`*addrobj*`]` subcommand displays address properties depending on the options that you use.

  To list the properties of all IP addresses, do not specify a property or an address object. To list the values of a single property for all IP addresses, specify only that property. To list all the properties of a specific address object, specify only the address object.
- The `ipadm set-addrprop -p` *property*=*value addrobj* subcommand assigns values to address properties. Note that you can only set one address property at a time.
- The `ipadm reset-addrprop -p` *property addrobj* subcommand 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 an Interface's Address (`ipadm delete-addr`)" on page 40.

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 commands 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
```

Setting TCP/IP Protocol Properties

Use the `ipadm` command to configure protocol properties, also known as *tunables*. The `ipadm` replaces the `ndd` command, which was commonly used in previous releases to set tunables.

TCP/IP properties can be either interface based or global. Properties can be applied to a specific interface or globally to all interfaces in a zone. Global properties can have different values in different non-global zones. For a list of supported protocol properties, refer to the `ipadm`(1M) man page.

Typically, the default values of the TCP/IP internet protocol suffice for the network to function. However, if the default values are insufficient for your network topology, then you can customize these properties as needed.

Three `ipadm` subcommands are used to set TCP/IP interface properties:

- The `ipadm show-prop -p` *property protocol* command displays the properties of a protocol and their current values. If you do not use the `-p` *property* option, then all the properties of the protocol are listed. If you do not specify a protocol, then all the properties of all protocols are listed.
- The `ipadm set-prop -p` *property=value protocol* subcommand assigns a value to the IP interface's property.
- The `ipadm reset-prop -p` *property protocol* subcommand resets the specific protocol property to its default values.

Note – If a property can receive multiple values, then you assign multiple values to the property with the += qualifier as follows:

```
ipadm set-prop -p property+=value1 [value2 value3 ...].
```

To remove one value from a set of values for a property, you use the −= qualifier as follows:

```
ipadm set-prop -p property-=value2
```

Enabling Packet Forwarding Globally

"Enabling Packet Forwarding" on page 41 shows how to enable packet forwarding on the interface. Setting packet forwarding on the IP interface property enables you to implement this feature selectively. You can enable this property only on specific interfaces on the system.

If you want to enable packet forwarding on the entire system regardless of the number of IP interfaces, then you use the protocol property: In protocols, the property name is the same as in IP interfaces, which is `forwarding`. You must issue the command separately to enable packet forwarding on IPv4 and IPv6 protocols.

The following example shows how to enable packet forwarding for all IPv4 and IPv6 traffic on the system:

```
# ipadm show-prop -p forwarding ip
PROTO   PROPERTY     PERM   CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ipv4    forwarding   rw     off       --           off       on,off
ipv6    forwarding   rw     off       --           off       on,off
#
# ipadm set-prop -p forwarding=on ipv4
# ipadm set-prop -p forwarding=on ipv6
#
# ipadm show-prop ip
PROTO   PROPERTY     PERM   CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ipv4    forwarding   rw     on        on           off       on,off
ipv4    ttl          rw     255       --           255       1-255
```

```
ipv6    forwarding   rw     on        on           off       on,off
ipv6    hoplimit     rw     255       --           255       1-255#
```

Note – The `forwarding` property of either IP interfaces or protocols is not 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.

Setting Up a Privileged Port

On transport protocols such as TCP, UDP, and SCTP, ports `1`–`1023` are default privileged ports where only processes that run with `root` permissions can bind to these ports. By using the `ipadm` command, you can reserve a port beyond this given default range such that it becomes a privileged port. Thus, only `root` processes can bind to that port. To set up a privileged port, you customize the following transport protocol properties:

- `smallest_nonpriv_port` – The property whose value indicates the range of port numbers to which regular users can bind. If your designated port is within this range, then you can set it as a privileged port. Use the `ipadm show-prop` command to display the property's values.
- `extra_priv_ports` – The property that specifies which ports are privileged. Use the `ipadm set-prop` subcommand to specify ports you want to restrict. This property can be assigned multiple values.

As an example, suppose you want to set TCP ports `3001` and `3050` as privileged ports with access restricted only to the `root` user. The `smallest_nonpriv_port` property indicates that `1024` is the lowest port number for a non privileged port. Therefore, the designated ports `3001` and `3050` can be changed to become privileged ports. You would proceed by issuing commands similar to the following:

```
# ipadm show-prop -p smallest_nonpriv_port tcp
PROTO PROPERTY                PERM   CURRENT   PERSISTENT   DEFAULT   POSSIBLE
tcp   smallest_nonpriv_port   rw     1024      --           1024      1024-32768

# ipadm show-prop -p extra_priv_ports tcp
PROTO    PROPERTY           PERM   CURRENT     PERSISTENT   DEFAULT     POSSIBLE
tcp      extra_priv_ports   rw     2049,4045   --           2049,4045   1-65535

# ipadm set-prop -p extra_priv_ports+=3001 tcp
# ipadm set-prop -p extra_priv_ports+=3050 tcp
# ipadm show-prop -p extra_priv_ports tcp
PROTO    PROPERTY           PERM   CURRENT     PERSISTENT   DEFAULT     POSSIBLE
tcp      extra_priv_ports   rw     2049,4045   3001,3050    2049,4045   1-65535
                                   3001,3050
```

To remove one of the ports, for example, `4045`, from being a privileged port, you would type the following commands:

```
# ipadm set-prop -p extra_priv_ports-=4045 tcp
# ipadm show-prop -p extra_priv_ports tcp
PROTO   PROPERTY           PERM   CURRENT     PERSISTENT   DEFAULT     POSSIBLE
tcp     extra_priv_ports   rw     2049,3001   3001,3050    2049,4045   1-65535
                                  3050
```

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 ideal in certain cases. 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 a response to a certain incoming request, such as an ICMP echo request, the request and the response might not traverse the same interface. Such a traffic routing configuration is called *asymmetric routing*. If your Internet service provider is implementing ingress filtering as described in RFC 3704 (`http://rfc-editor.org/rfc/bcp/bcp84.txt`), an asymmetric routing configuration might cause an outgoing packet to be dropped by the provider.

RFC 3704 intends to limit denial-of-service attacks across the Internet. To comply with this intent, your network must be configured for symmetric routing. In Oracle Solaris, 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.

The following example shows how to implement symmetric routing of IP packets on a multihomed host.

```
# ipadm set-prop -p hostmodel=strong ip
# 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
```

Implementing Traffic Congestion Control

Network congestion typically occurs in the form of router buffer overflows, when nodes send more packets than the network can accommodate. Various algorithms prevent traffic congestion through establishing controls on the sending systems. These algorithms are supported in Oracle Solaris and can be easily added or directly plugged in to the operating system.

The following table lists and describes the supported algorithms.

Algorithm	Oracle Solaris Name	Description
NewReno	`newreno`	Default algorithm in Oracle Solaris. Control mechanism includes sender's congestion window, slow start, and congestion avoidance.
HighSpeed	`highspeed`	One of the best known and simplest modifications of NewReno for high-speed networks.
CUBIC	`cubic`	Currently the default algorithm in Linux 2.6. Changes the congestion avoidance phase from linear window increase to a cubic function.
Vegas	`vegas`	A classic delay-based algorithm that attempts to predict congestion without triggering actual packet loss.

Congestion control is enabled by setting the following control-related TCP properties. Although these properties are listed for TCP, the control mechanism that is enabled by these properties also applies to SCTP traffic.

- `cong_enabled` – contains a list of algorithms, separated by commas, that are currently operational in the system. You can add or remove algorithms to enable only those algorithms you want to use. This property can have multiple values. Therefore you must use either the `+=` qualifier or the `-=` qualifier, depending on the change you want to effect.
- `cong_default` – the algorithm that is used by default when applications do not specify the algorithms explicitly in socket options. Currently, the value of the `cong_default` property applies to both global and non-global zones.

To add an algorithm for congestion control to the protocol, issue the following command:

```
# ipadm set-prop -p cong_enabled+=algorithm tcp
```

To remove an algorithm, issue the following command:

```
# ipadm set-prop -p cong_enabled-=algorithm tcp
```

To replace the default algorithm, issue the following command:

```
# ipadm set-prop -p cong_default=algorithm tcp
```

Note – No sequence rules are followed when you add or remove algorithms. You can remove an algorithm before adding other algorithms to a property. However, the `cong_default` property must always have a defined algorithm.

The following example shows steps that you might take to implement congestion control. In the example, the default algorithm for the TCP protocol is changed from `newreno` to `cubic.` Then, the `vegas` algorithm is removed from the list of enabled algorithms.

```
# ipadm show-prop -p cong_default,cong_enabled tcp
PROTO  PROPERTY      PERM  CURRENT          PERSISTENT   DEFAULT   POSSIBLE
tcp    cong_default  rw    newreno          --           newreno   -
tcp    cong_enabled  rw    newreno,cubic,   --           newreno   newreno,cubic,
                           highspeed,                              highspeed,vegas
                           vegas

# ipadm set-prop -p cong_enabled-=vegas tcp
# ipadm set-prop -p cong_default=cubic tcp

# ipadm show-prop -p cong_default,confg_enabled tcp
PROTO  PROPERTY      PERM  CURRENT          PERSISTENT   DEFAULT   POSSIBLE
tcp    cong_default  rw    cubic            --           newreno   -
tcp    cong_enabled  rw    newreno,cubic,   --           newreno   newreno,cubic,
                           highspeed                               highspeed,vegas
```

Changing the TCP Receive Buffer Size

The size of the TCP receive buffer is set by using the TCP property `recv_buf` which, by default, is 128 KB. However, applications do not use available bandwidths uniformly. Thus connection latency might require you to change the default size. For example, using the Secure Shell feature of Oracle Solaris causes overhead on bandwidth use because of the additional checksum and encryption processes that are performed on the data stream. Thus, the buffer size might need to be increased. Likewise, for applications that perform bulk transfer holds to use bandwidth efficiently, the same buffer size adjustment is also required.

You can calculate the correct receive buffer size to use by estimating the *bandwidth delay product* (BDP) as follows:

*BDP = available_bandwidth * connection-latency*

Use `ping -s` *host* to obtain the value of connection latency. Use the `uperf` and `iperf` tools to estimate the use of bandwidth.

The appropriate receive buffer size approximates the value of the BDP. Note, however, that the use of bandwidth also depends on a variety of conditions. A shared infrastructure or the number of applications and users that compete for the use of bandwidth can change that estimate.

To change the value of the buffer size, use the following syntax:

```
# ipdadm set-prop -p recv_buf=value tcp
```

The following example shows how to increase the buffer size to 164 KB

```
# ipadm show-prop -p recv_buf tcp
PROTO PROPERTY   PERM CURRENT   PERSISTENT   DEFAULT  POSSIBLE
tcp   recv_buf   rw   128000       --        128000   2048-1048576

# ipadm set-prop -p recv_buf=164000 tcp

# ipadm show-prop -p recv_buf tcp
PROTO PROPERTY   PERM CURRENT   PERSISTENT   DEFAULT  POSSIBLE
tcp   recv_buf   rw   164000       --        164000   2048-1048576
```

No set value of the buffer size is preferred because the preferred size varies in different circumstances. Consider the following cases that show different values for the BDP for each network with its own specific conditions:

Typical 1 Gbps LAN where 128 KB is the default value of the buffer size:

```
BDP = 128 MBps * 0.001 s = 128 kB
```

Theoretical 1Gbps WAN with 100 ms latency:

```
BDP = 128 MBps * 0.1 s = 12.8 MB
```

Europe-to-U.S. link (bandwidth measured by `uperf`)

```
BDP = 2.6 MBps * 0.175 = 470 kB
```

If you cannot compute the BDP, use the following observations as guidelines:

- For bulk transfers over a LAN, the default value of the buffer size, 128 KB, is sufficient.
- For most WAN deployments, the receive buffer size should be in the 2 MB range.

Caution – Increasing the TCP receive buffer size increases the memory footprint of many network applications.

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 subcommands to restrict the information that you want to display by using the following syntax:

ipadm show-* [*other-arguments***] [***interface***]**

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

This section provides several examples of how to use the ipadm subcommands to obtain interface information. For an explanation of all the fields displayed by the ipadm show-* commands, refer to the ipadm(1M) man page.

Obtaining General Information About IP Interfaces

Using the ipadm command without accompanying subcommands provides default information about all 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 sample output provides the following information:

- The IP interfaces.
- The class of each interface.
- The state of each interface.

- The status of the interface as being either a "stand alone" IP interface or being 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.
- The address objects 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.
- The type of IP address, which is indented under the `CLASS/TYPE` heading and which can be `static`, `dhcp` and so on.
- The actual addresses listed under the `ADDRESS` column.

Thus, the `ipadm` command provides a comprehensive picture of the system's interfaces.

Obtaining Information About IP Interfaces

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

The fields in the command output refer to the following:

`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 provides:

```
# 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 for information about properties of IP interfaces. If you do not specify a property or an interface, then information about all the properties of all the IP interfaces on the system is provided.

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 is an example of the information that the `ipadm show-ifprop` subcommand provides:

```
# 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*`]` subcommand. If you do not specify an interface, then the information about all the IP addresses on the system is provided.

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 6, "Configuring IP Tunnels," in *Configuring and Administering Oracle Solaris 11.1 Networks*.

The following is an example of the information that the `ipadm 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 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 listed 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 IP Interface" on page 18.

Obtaining Information About IP Address Properties

For information about IP address properties, use the `ipadm show-addrprop [`*addrobj*`]` subcommand. To list all the properties, omit the *addrobj* option. To list a single property for all the IP addresses, specify only the property. To list all the properties of a specific address, specify only 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 information the `ipadm show-addrprop` subcommand provides:

```
# 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 5

Configuring Wireless Networking on Laptops Running Oracle Solaris

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.

The following topics are covered:

- "WiFi Communications Task Map" on page 55
- "Secure WiFi Communications" on page 61

WiFi Communications Task Map

Task	Description	For Instructions
Connect to a WiFi network	Set up and establish communications with a local WiFi network.	"How to Connect to a WiFi Network" on page 56
Monitor communications on the WiFi link.	Use standard Oracle Solaris networking tools to check the state of the WiFi link.	"How to Monitor the WiFi Link" on page 59
Establish secure WiFi communications.	Create a Wired Equivalent Privacy (WEP) key and use it establish connections with a secure WiFi network	"How to Set Up an Encrypted WiFi Network Connection" on page 61

How to Connect to a WiFi Network

Before You Begin Perform the following steps to connect your laptop to a WiFi network.

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Display the physical attributes of datalinks.**

```
# dladm show-phys
LINK            MEDIA                STATE    SPEED  DUPLEX    DEVICE
net0            Ethernet             up       1500   full      ath0
net1            Ethernet             up       1500   full      e1000g0
```

   In this example, the output indicates that two links are available. `Net0` over the device `ath0` link supports WiFi communications. The `e1000g0` link serves to connect the system to a wired network.

3. **Configure the WiFi interface.**

   Use the following steps to configure the interface:

   a. **Create the interface that supports WiFi:**

```
# 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 example output of the `scan-wifi` command displays information about the available WiFi networks at the current location. The information in the output includes:

   `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, the 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 needed to access the network. The values are `none` or `WEP`. For information about WEP, refer to "Secure WiFi Communications" on page 61.
`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, or g, or these modes in combination.
`SPEED`	Refers to the speed in megabits per second of the particular network.

5 Connect to a WiFi network.

Do either of the following:

- Connect to the unsecured WiFi network with the strongest signal.

  ```
  # dladm connect-wifi
  ```

- Connect to an unsecured network by specifying its `ESSID`.

  ```
  # dladm connect-wifi -e ESSID
  ```

  The `connect-wifi` subcommand of `dladm` has several more options for connecting to a WiFi network. For complete details, refer to the `dladm(1M)` man page.

6 Configure an IP address for the interface.

Do either of the following:

- Obtain an IP address from a DHCP server.

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

  If the WiFi network does not support DHCP, you receive the following message:

  ```
  ipadm: interface: interface does not exist or cannot be managed using DHCP
  ```

- Configure a static IP address:

  Use this option if you have a dedicated IP address for the system.

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

7 Check the status of the WiFi network to which the system is connected.

```
# dladm show-wifi
LINK       STATUS        ESSID       SEC     STRENGTH    MODE    SPEED
net0       connected     ofc         none    very good    g       36Mb
```

In this example, the output indicates that the system is now connected to the `ofc` network. The earlier `scan-wifi` output from Step 4 indicated that `ofc` has the strongest signal among the available networks. The `dladm connect-wifi` command automatically chooses the WiFi network with strongest signal, unless you directly specify a different network.

8 **Access the Internet through the WiFi network.**

Do either of the following, depending on the network to which the system is connected:

- If the access point offers free service, you can now run a browser or an application of your choice.
- If the access point is in a commercial WiFi network that requires a fee, follow the instructions provided at the current location. Typically, you run a browser, supply a key, and give credit card information to the network provider.

9 **Conclude the session.**

Do one of the following:

- 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 turning off the system through the `shutdown` command.

Example 5–1 Connecting to a Specific WiFi Network

The following example combines the different steps you would take to connect your Oracle Solaris laptop 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 example assume that you have the static IP address `10.192.16.3/24` configured on your laptop. The example begins with determining the availability of a WiFi link.

```
# 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
```

```
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

# 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

# dladm show-wifi
LINK       STATUS        ESSID         SEC     STRENGTH    MODE   SPEED
net0       connected     citinet       none    good        g      11Mb
```

Run a browser or other application to commence your work over the WiFi network.

```
# firefox
```

The home page for the Firefox browser appears.

Terminate the session but leave the laptop 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

This procedure explains how to monitor the status of a WiFi link through standard networking tools and change a selected link property through the `linkprop` subcommand.

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Connect to the WiFi network, as described in "How to Connect to a WiFi Network" on page 56.**

3 View the properties of the link.

Use the following syntax:

```
# dladm show-linkprop link
```

For example, you would use the following syntax to show the status of the connection established over the `net0` wireless link:

```
# dladm show-linkprop net0
...
PROPERTY        VALUE           DEFAULT         POSSIBLE
channel         5               --              --
powermode       off             off             off,fast,max
radio           ?               on              on,off
speed           36               --              1,2,5.5,6,9,11,12,18,24,36,48,54
...
```

4 Set a fixed speed for the link.

Caution – Oracle Solaris automatically chooses the optimal speed for the WiFi connection. Modifying the initial speed of the link might cause reduced performance or prevent the establishment of certain WiFi connections.

You can modify the link speed to one of the possible values for speed that is listed in the `show-linkprop` output.

```
# 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 5–2 Setting the Speed of a Link

This example shows how to set the speed of a link after you have connected to a WiFi network

```
# dladm show-linkprop -p speed net0
PROPERTY        VALUE           DEFAULT         POSSIBLE
speed           24               --              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
PROPERTY        VALUE           DEFAULT         POSSIBLE
speed           36               --              1,2,5,6,9,11,12,18,24,36,48,54
```

Secure WiFi Communications

Radio wave technology makes WiFi networks readily available and often freely accessible to users in many locations. As a result, connecting to a WiFi network can be an insecure undertaking. However, certain types of WiFi connections are more secure:

- Connecting to a private, restricted-access WiFi network

  Private networks, such as internal networks established by corporations or universities, restrict access to their networks 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 VPN.

- Encrypting your connection to the 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 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) key for encrypting connections through the access point. The WEP protocol is defined in IEEE 802.11 specifications for wireless connections. For complete details on the WEP-related options of the `dladm` command, refer to the `dladm(1M)` man page.

▼ How to Set Up an Encrypted WiFi Network Connection

The next procedure explains 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 can generate a secure key.

Before You Begin If you are connecting to your own home's wireless network, make sure that you have configured your router and have generated the WEP key. Follow the router manufacturer's documentation for generating and saving the key configuration.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Create a secure object that contains the WEP key.

Open a terminal window on the system and type the following:

```
# 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 in hexadecimal 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 for 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
```

The `wep` value under the `SEC` heading indicates that WEP encryption is in place for the connection.

Example 5–3 Setting Up Encrypted WiFi Communications

This example assumes that you have already done the following:

- Connected your system to a home router that can create a WEP key
- 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

Create a secure object.

```
# 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.

```
# dladm show-secobj
OBJECT               CLASS
 mykey               wep
# dladm connect-wifi -e citinet -k mykey net0
```

The preceding command establishes an encrypted connection to the WiFi network `citinet` by using the secure object `mykey`.

```
# dladm show-wifi
LINK       STATUS        ESSID         SEC     STRENGTH   MODE   SPEED
net0       connected     citinet       wep     good       g      36Mb
```

This output verifies that you are connected to `citinet` through WEP encryption.

APPENDIX A

Comparison Map: ifconfig and ipadm Commands

The ipadm command has replaced the ifconfig command for the purpose of configuring network interfaces. Although the ifconfig command is still functional in Oracle Solaris 11, the ipadm command is the preferred tool for networking configuration. However, some ifconfig options do not have equivalent in ipadm subcommands. The following table lists selected command options of the ifconfig command and their equivalents in the ipadm command.

Note – The table does not provide a comprehensive list of ipadm options. For a full list, see the ipadm(1M) man page.

TABLE A–1 Syntax Mapping Between the ifconfig and ipadm Commands

ifconfig Command	ipadm Command
plumb/unplumb	ipadm create-ip ipadm create-vni ipadm create-ipmp ipadm enable-addr ipadm delete-ip ipadm delete-vni ipadm delete-ipmp ipadm disable-addr

TABLE A–1 Syntax Mapping Between the ifconfig and ipadm Commands *(Continued)*

ifconfig Command	ipadm Command
[*address*[/*prefix-length*] [*dest-address*]] [addif *address*[*prefix-length*]] [removeif *address*[*prefix-length*]][netmask *mask*][destination *dest-address*]{auto-dhcp|dhcp}[primary][wait *seconds*]extend | release | start	ipadm create-addr ipadm create-addr -T dhcp ipadm create-addr -T addrconf ipadm delete-addr ipadm refresh-addr
[deprecated | -deprecated] [preferred | -preferred] [private | -private] [zone *zonename* | -zones | -all-zones][xmit | -xmit]	ipadm set-addprop ipadm reset-addprop ipadm show-addprop
up	ipadm up-addr
down	ipadm down-addr
[metric *n*] [mtu *n*] [nud | -nud] [arp | -arp] [usesrc [*name* | none] [router | -router]	ipadm set-ifprop ipadm show-ifprop ipadm reset-ifprop
[ipmp] [group [*name* | ""]] standby | -standby] [failover | -failover]	ipadm create-ipmp ipadm delete-ipmp ipadm add-ipmp ipadm remove-ipmp ipadm set-ifprop -p [standby] [group]
[*interface*] [-a]	ipadm ipadm show-if ipadm show-addr
[tdst *tunnel-dest-addr*] [tsrc *tunnel-srcs-addr*] [encaplimit *n*| -encaplimit] [thoplimit *n*]	dladm *-iptun set of commands. For more details, see the dladm(1M) man page and “Tunnel Configuration and Administration With the dladm Command” in *Configuring and Administering Oracle Solaris 11.1 Networks*.
[auth_algs *authentication algorithm*] [encr_algs *encryption algorithm*] [encr_auth_algs *encryption authentication algorithm*]	ipsecconf For details, see the ipsecconf(1M) man page and Chapter 7, “Configuring IPsec (Tasks),” in *Securing the Network in Oracle Solaris 11.1*.

TABLE A–1 Syntax Mapping Between the `ifconfig` and `ipadm` Commands *(Continued)*

`ifconfig` Command	`ipadm` Command
`[auth_revarp] [ether [`*address*`]] [index` *if-index*`] [subnet` *subnet-address*`] [broadcast` *broadcast-address*`] [token` *address/prefix-length*`]` DHCP options – `inform, ping, release, status, drop`	Equivalent subcommands currently unavailable.
`modlist] [modinsert` *mod_name@pos*`] [modremove` *mod_name@pos*`]`	Equivalent subcommands currently unavailable.

APPENDIX B

Comparison Map: ndd and ipadm Commands

The ipadm command has replaced the ndd command for the purpose of customizing network parameters or tunables. Although the ndd command is still functional in Oracle Solaris 11, the ipadm command is the preferred tool for customizing network parameters. However, some ndd options do not have equivalent ipadm subcommands. The following table lists selected command options of the ndd command and their equivalents in the ipadm command.

Note – The table does not provide a comprehensive list of ipadm options. For a full list, see theipadm(1M) man page.

TABLE B–1 Syntax Mapping Between the ndd and ipadm Commands: Retrieving Properties

ndd Command	ipadm Command
bash-3.2# **ndd -get /dev/ip ?** ip_def_ttl (read and write) ip6_def_hops (read and write) ip_forward_directed_broadcasts (read and write) ip_forwarding (read and write)	bash-3.2# **ipadm show-prop ip** PROTO PROPERTY PERM CURRENT PERSISTENT DEFAULT POSSIBLE ipv4 forwarding rw off -- off on,off ipv4 ttl rw 255 -- 255 1-255 ipv6 forwarding rw off -- off on,off ipv6 hoplimit rw 255 -- 255 1-255 ...
bash-3.2# **ndd -get /dev/ip ** **ip_def_ttl** 100 bash-3.2# **ndd -get /dev/ip ** **ip6_def_hops** 255	bash-3.2# **ipadm show-prop -p ttl,hoplimit ip** PROTO PROPERTY PERM CURRENT PERSISTENT DEFAULT POSSIBLE ipv4 ttl rw 255 -- 255 1-255 ipv6 hoplimit rw 255 -- 255 1-255
bash-3.2# **ndd -get /dev/tcp ?** tcp_cwnd_max (read and write) tcp_strong_iss (read and write) tcp_time_wait_interval (read and write) tcp_tstamp_always (read and write) tcp_tstamp_if_wscale (read and write)	bash-3.2# **ipadm show-prop tcp** PROTO PROPERTY PERM CURRENT PERSISTENT DEFAULT POSSIBLE tcp ecn rw passive -- passive never,passive, active tcp extra_ priv_ports rw 2049 2049,4045 2049,4045 1-65535 tcp largest_ anon_port rw 65535 -- 65535 1024-65535 tcp recv_ maxbuf rw 128000 -- 128000 2048-1073741824 tcp sack rw active -- active never,passive, active tcp send_ maxbuf rw 49152 -- 49152 4096-1073741824 tcp smallest_ anon_port rw 32768 -- 32768 1024-65535 tcp smallest_ nonpriv_port rw 1024 -- 1024 1024-32768 ... ...
bash-3.2# **ndd -get /dev/tcp ecn** 1 bash-3.2# **ndd -get /dev/tcp sack** 2	bash-3.2# **ipadm show-prop -p ecn,sack tcp** PROTO PROPERTY PERM CURRENT PERSISTENT DEFAULT POSSIBLE tcp ecn rw passive -- passive never,passive,active tcp sack rw active -- active never,passive,active

TABLE B–2 Syntax Mapping Between the ndd and ipadm Commands: Setting Properties

ndd Command	ipadm Command
bash-3.2# **ndd -set /dev/ip ** **ip_def_ttl** 64 bash-3.2# **ndd -get /dev/ip ** **ip_def_ttl** 64	bash-3.2# **ipadm set-prop -p ttl=64 ipv4** bash-3.2# **ipadm show-prop -p ttl ip** PROTO PROPERTY FAMILY PERM VALUE DEFAULT POSSIBLE ip ttl inet rw 64 255 1-255 PROTO PROPERTY PERM CURRENT PERSISTENT DEFAULT POSSIBLE ipv4 ttl rw 64 64 255 1-255 bash-3.2# **ipadm reset-prop -p ttl ip** bash-3.2# **ipadm show-prop -p ttl ip** PROTO PROPERTY PERM CURRENT PERSISTENT DEFAULT POSSIBLE ipv4 ttl rw 255 255 255 1-255

Index

T

U

V

W