System Administration Guide: Network Interfaces and Network Virtualization

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–6990–02
January 2008

090312@21990

Contents

Preface

Welcome to the System Administration Guide: Network Interfaces and Network Virtualization. This book is part of a fourteen-volume set that covers a significant part of the Solaris™ system administration information. This book assumes that you have already installed the Solaris operating system (Solaris OS). You should be ready to configure your network or ready to configure any networking software that is required on your network.

Note – This Solaris release supports systems that use the SPARC® and x86 families of processor architectures: UltraSPARC®, SPARC64, AMD64, Pentium, and Xeon EM64T. The supported systems appear in the *Solaris 10 Hardware Compatibility List* at `http://www.sun.com/bigadmin/hcl`. This document cites any implementation differences between the platform types.

In this document these x86 related terms mean the following:

- "x86" refers to the larger family of 64-bit and 32-bit x86 compatible products.
- "x64" points out specific 64-bit information about AMD64 or EM64T systems.
- "32-bit x86" points out specific 32-bit information about x86 based systems.

For supported systems, see the *Solaris 10 Hardware Compatibility List*.

Who Should Use This Book

This book is intended for anyone responsible for administering systems that run the Solaris OS release, 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.

How the System Administration Guides Are Organized

Here is a list of the topics that are covered by the System Administration Guides.

Book Title	Topics
System Administration Guide: Basic Administration	User accounts and groups, server and client support, shutting down and booting a system, managing services, and managing software (packages and patches)
System Administration Guide: Advanced Administration	Terminals and modems, system resources (disk quotas, accounting, and crontabs), system processes, and troubleshooting Solaris software problems
System Administration Guide: Devices and File Systems	Removable media, disks and devices, file systems, and backing up and restoring data
System Administration Guide: IP Services	TCP/IP network administration, IPv4 and IPv6 address administration, DHCP, IPsec, IKE, Solaris IP filter, Mobile IP, and IPQoS
System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)	DNS, NIS, and LDAP naming and directory services, including transitioning from NIS to LDAP and transitioning from NIS+ to LDAP
System Administration Guide: Naming and Directory Services (NIS+)	NIS+ naming and directory services
System Administration Guide: Network Interfaces and Network Virtualization	Networking stack, NIC driver property configuration, network interface configuration, administration of VLANs and link aggregations, IP networking multipathing (IPMP), WiFi wireless networking configuration, and virtual NICs (VNICs).
System Administration Guide: Network Services	Web cache servers, time-related services, network file systems (NFS and Autofs), mail, SLP, and PPP
System Administration Guide: Security Services	Auditing, device management, file security, BART, Kerberos services, PAM, Solaris Cryptographic Framework, privileges, RBAC, SASL, and Solaris Secure Shell
System Administration Guide: Virtualization Using the Solaris Operating System	Resource management features, which enable you to control how applications use available system resources; zones software partitioning technology, which virtualizes operating system services to create an isolated environment for running applications; and virtualization using Sun™ xVM hypervisor technology, which supports multiple operating system instances simultaneously

Book Title	Topics
Solaris CIFS Administration Guide	Solaris CIFS service, which enables you to configure a Solaris system to make CIFS shares available to CIFS clients; and native identity mapping services, which enables you to map user and group identities between Solaris systems and Windows systems
Solaris ZFS Administration Guide	ZFS storage pool and file system creation and management, snapshots, clones, backups, using access control lists (ACLs) to protect ZFS files, using ZFS on a Solaris system with zones installed, emulated volumes, and troubleshooting and data recovery
Solaris Trusted Extensions Administrator's Procedures	System installation, configuration, and administration that is specific to Solaris Trusted Extensions
System Administration Guide: Solaris Printing	Solaris printing topics and tasks, using services, tools, protocols, and technologies to set up and administer printing services and printers

Related Third-Party Web Site References

Third party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party Web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Typographic Conventions

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

TABLE P–1 Typographic Conventions

Typeface	Meaning	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 the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

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

Network Interface Administration in the Solaris Operating System

This chapter introduces network administration in the Solaris OS. It describes interrelationships that underlie interfaces, data links over which the interfaces are configured, and network devices. Support for flexible names for data links is also discussed at length.

Overview of the Networking Stack

Network interfaces provide the connection between the system and the network. These interfaces are configured over data links, which in turn correspond to instances of hardware devices in the system. Network hardware devices are also called *network interface cards (NICs)* or *network adapters*. NICs can be built in and already present in the system when the system is purchased. However, you can also purchase separate NICs to add to the system. Certain NICs have only a single interface that resides on the card. Many other brands of NICs have multiple interfaces that you can configure to perform network operations.

In the current model of the network stack, interfaces and links on the software layer build on the devices in the hardware layer. More specifically, a hardware device instance in the hardware layer has a corresponding link on the data-link layer and a configured interface on the interface layer. This one-to-one relationship among the network device, its data link, and the IP interface is illustrated in the figure that follows.

Note – For a fuller explanation of the TCP/IP stack, see Chapter 1, "Solaris TCP/IP Protocol Suite (Overview)," in *System Administration Guide: IP Services*.

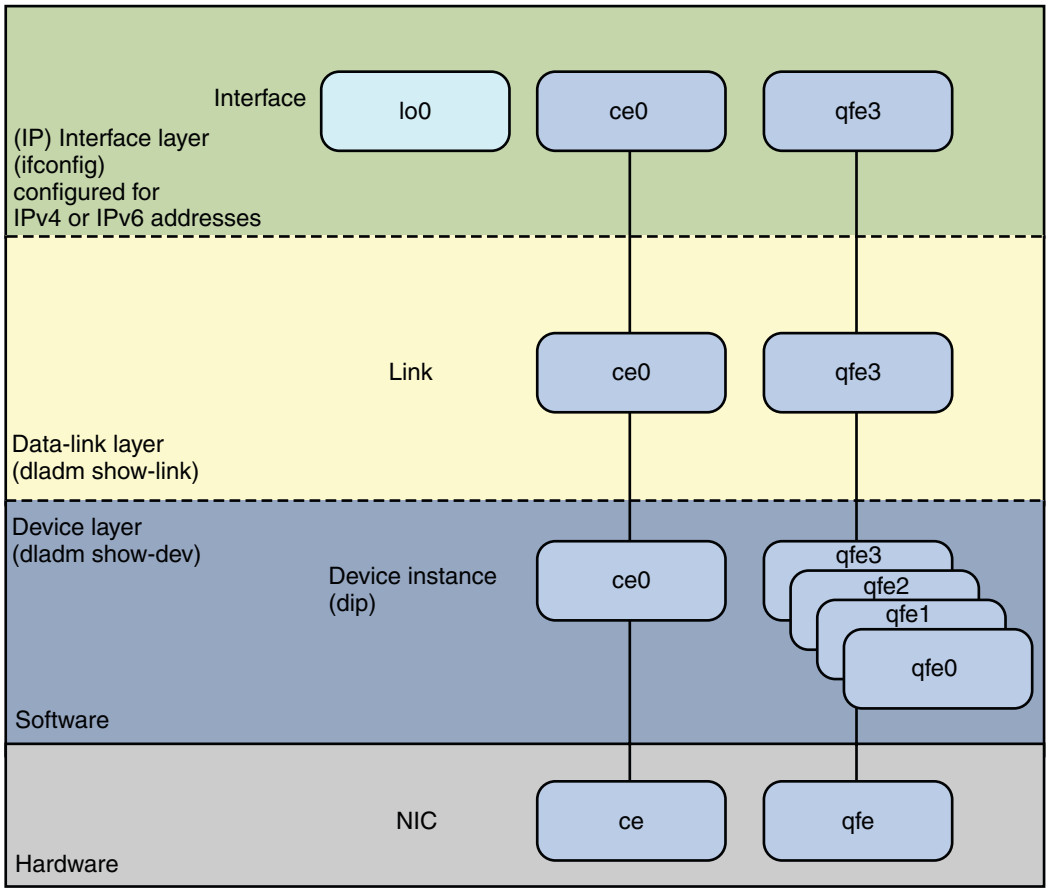

FIGURE P–1 Network Stack Showing Network Devices, Links, and Interfaces

The figure shows two NICs on the hardware layer: `ce` with a single device instance `ce0`, and `qfe` with multiple device instances, `qfe0` to `qfe3`. The devices `qfe0` through `qfe2` are not used. Devices `ce0` and `qfe3` are used and have corresponding links `ce0` and `qfe3` on the data-link layer. In the figure, the IP interfaces are likewise named after their respective underlying hardware, `ce0` and `qfe3`. These interfaces can be configured with IPv4 or IPv6 addresses to host both types of network traffic. Note also the presence of the loopback interface `lo0` on the interface layer. This interface is used to test, for example, that the IP stack is functioning properly.

Different administrative commands are used at each layer of the stack. For example, hardware devices that are installed on the system are listed by the `dladm show-dev` command. Information about links on the data-link layer is displayed by the `dladm show-link` command. The `ifconfig` command shows the IP interface configuration on the interface layer.

In this model, a one-to-one relationship exists that binds the device, the data link, and the interface. This relationship means that network configuration is dependent on hardware configuration and network topology. Interfaces must be reconfigured if changes are implemented in the hardware layer, such as replacing the NIC or changing the network topology.

The Solaris OS introduces a new implementation of the network stack in which the basic relationship between the hardware, data link, and interface layers remains. However, the software layer is decoupled from the hardware layer. With this separation, network configuration on the software level is no longer bound to the chipset or the network topology in the hardware layer. The new implementation makes network administration more flexible in the following two ways:

- The network configuration is insulated from any changes that might occur in the hardware layer. Link and interface configurations are preserved even if the underlying hardware is removed. These same configurations can then be reapplied to any replacement NIC, provided that the two NICs are of the same type.
- The separation of the network configuration from the network hardware configuration also allows the use of customized link names in the data-link layer. This feature is further explained in the following section.

Assigning Names to Data Links

From an administrative perspective, a network interface has a *link name*. The data link represents a data-link object in the second layer of the Open Systems Interconnection (OSI) model. The *physical link* is directly associated with a device and possesses a device name. The device name is essentially the device instance name, and is composed of the driver name and the device instance number.

Driver names can be `ce`, `hme`, `bge`, `e1000g`, among many other driver names. The variable *instance-number* can have a value from zero to *n*, depending on how many interfaces of that driver type are installed on the system.

For example, consider a 100BASE-TX Fast Ethernet card, which is often used as the primary NIC on both host systems and server systems. Some typical driver names for this NIC are `eri`, `qfe`, and `hme`. When used as the primary NIC, the Fast Ethernet interface has a device name such as `eri0` or `qfe0`.

Only one interface can be configured on NICs such as `eri` and `hme`. However, many brands of NICs can have multiple interfaces. For example, the Sun Quad FastEthernet™ (`qfe`) card has four interfaces, `qfe0` through `qfe3`. See Figure P–1.

With the separation of the network configuration between the software layer and the hardware layer, you can now use flexible names for data links . The device instance name continues to be based on the underlying hardware and cannot be changed. However, the data link name is no

longer similarly bound. Thus, you can change the device instance's link name to a name that is more meaningful in your network setup. You assign a customized name to the link, and then perform network configuration and maintenance tasks by referring to the assigned link name instead of the hardware-based name.

Using the information in Figure P–2, the following table illustrates the new correspondence between the hardware (NIC), the device instance, the link name, and the interface over the link.

Hardware (NIC)	Device Instance	Link's Assigned Name	IP Interface
`ce`	`ce0`	`subitops0`	`subitops0`
`qfe`	`qfe3`	`subitops1`	`subitops1`

As the table indicates, the `ce0` device instance's link is assigned the name `subitops0`, while the link for the `qfe3` instance is assigned the name `subitops1`. Such names allow you to readily identify links and their functions on the system. In this example, the links have been designated to service IT Operations.

Administration of Other Link Types

The separation between network configuration and network hardware configuration introduces the same flexibility to other types of link configurations. For example, virtual local area networks (VLANs), link aggregations, and IP tunnels can be assigned administratively-chosen names and then configured by referring to those names. Other related tasks, such as performing dynamic reconfiguration (DR) to replace hardware devices, are also easier to perform because no further network reconfiguration is required, provided that the network configuration was not deleted.

The following figure shows the interrelationship among devices, link types, and their corresponding interfaces.

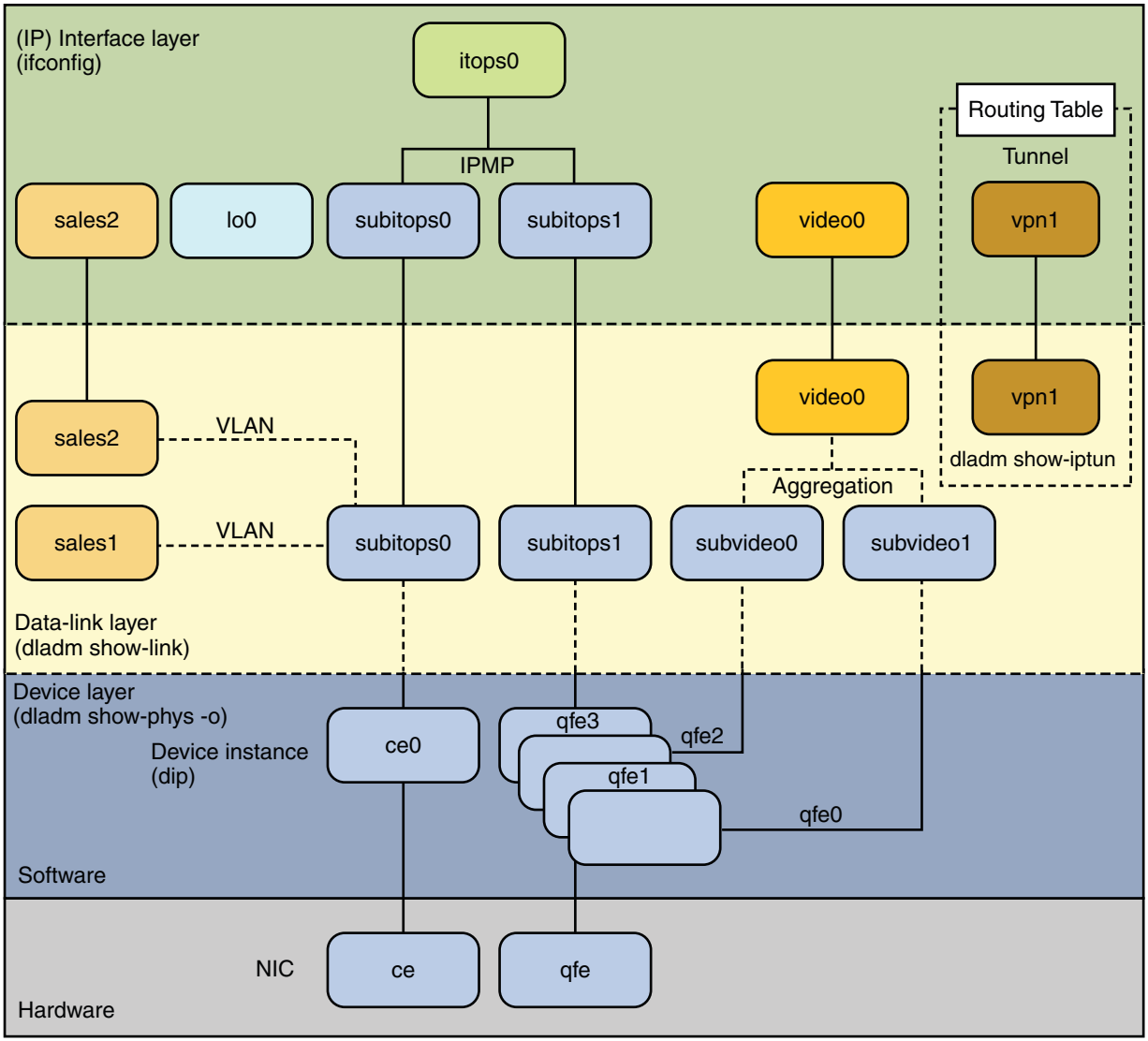

FIGURE P–2 Types of Link Configurations in the Network Stack

The figure also provides a sample of how administratively chosen names can be used in the network setup;

- The device instances `ce0` and `qfe3` are designated to service traffic for IT Operations, and hence given the link names `subitops0` and `subitops1`. The customized names facilitate the identification of the roles of these links.

- VLANs are configured on the `subitops0` link. These VLANs, in turn, are also assigned customized names, such as `sales1` and `sales2`. The VLAN `sales2`'s IP interface is plumbed and operational.
- The device instances `qfe0` and `qfe2` are used to service video traffic. Accordingly, the corresponding links in the data-link layer are assigned the names `subvideo0` and `subvideo1`. These two links are aggregated to host video feed. The link aggregation possesses its own customized name as well, `video0`.
- Two interfaces (`subitops0` and `subitops1`) with different underlying hardware (`ce` and `qfe`) are grouped together as an IPMP group (`itops0`) to host email traffic.

Note – Although IPMP interfaces are not links on the data-link layer, these interfaces, like the links, can also be assigned customized names. For more information about IPMP groups, see Chapter 7, "Introducing IPMP."

- Two interfaces have no underlying devices: the tunnel `vpn1`, which is configured for VPN connections and `lo0` for IP loopback operations.

All of the link and interface configurations in this figure are independent of the configurations in the underlying hardware. For example, if the `qfe` card is replaced, the `video0` interface configuration for video traffic remains and can later be applied to a replacement NIC.

Considerations for Working With Link Names

After you install the Solaris OS, your system's network links retain their original hardware-based names, such as `bge0` or `ce0`. However, in the new network implementation, these link names are no longer bound to their associated hardware. You can replace the link names with names that are more meaningful within the context of your network environment. Interface configurations are then performed by using the link names.

Before you change link names, note the following important considerations.

Replacing Hardware-Based Link Names

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

For example, you retain the link name `bge0` that is associated with the device `bge0`. All link configurations are performed by referring to the link name. Later, you might replace the NIC `bge` with the NIC `ce`. To reapply the former device's link configuration to the new NIC `ce0`, you would need to reassign the link name `bge0` to `ce0`. The combination of a hardware-based link

name bge0 with a different associated NIC ce0 can cause confusion. By using names that are not hardware-based, you can better distinguish the links from the associated devices.

Caution About Changing Link Names

Replacing hardware-based link names is recommended. However, you must plan carefully before you rename links. Prior to the installation of the Solaris release, your system might already have other configurations that are associated with the NIC's hardware-based name. Changing the device's link name does not automatically propagate the new name to all associated configurations. The following examples illustrate the risks when you change link names:

- You create a VLAN link, bge1000, over the link bge0. If you rename bge0 to net0, this step applies only to the link bge0. If you attempt to plumb net1000, the operation fails because the VLAN link continues to exist as bge1000. If you then unplumb and replumb bge1000, replumbing fails because bge0 no longer exists after being renamed net0.
- Some rules in a Solaris IP Filter configuration apply to specific links. When you change a link's name, the filter rules continue to refer to the link's original name. Consequently, these rules no longer behave as expected after you rename the link. You need to adjust the filter rules to apply to the link by using the new link name.

Thus, as a general rule, do not rename data links randomly. When renaming data links, ensure that all of the link's associated configurations continue to apply after the link name is changed. Some of the configurations that might be affected by renaming links are as follows:

- Solaris IP Filter rules
- All IP configurations that are specified in configuration files such as /etc/dhcp.* or /etc/hostname[6].*
- Zones
- autopush configuration

Note – No changes are required in the autopush configuration when you rename links. However, you must be aware of how the configuration would work with the per-link autopush property after the link has been renamed. For more information, see "How to Set STREAMS Modules on Data Links" on page 45.

When to Rename Links

The following describe circumstances when renaming links can be usefully applied:

- To identify an available link by another name. For example, a link bge0 is renamed to net0. In this situation, all link configuration of bge0 becomes based on net0. Note that renaming the link is allowed only if the original link, bge0 in this example, is not in use.

- To transfer a configuration of a removed physical link to a nonexistent link. The second link is nonexistent because the replacement hardware is not yet connected to the system. For example, `support0` is the link name of the device instance `bge0`. The NIC `bge` is removed to be replaced by the NIC `ce`. Even before you install the NIC `ce` on the system, you can already issue the command to rename the data link `ce0` to `support0`. After you connect the NIC, `ce0` inherits all the configuration that was formerly based on `bge0`.
- To transfer an inactive link configuration of a removed NIC to a connected replacement link. This case is similar to the preceding case except that the replacement NIC is connected first before the inactive link is renamed. Thus, the replacement device `ce` is first connected, and then renamed `support0`.

Rules for Valid Link Names

When you assign link names, observe the following rules:

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

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

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

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

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

Link Names and the `dladm` Command

Subcommands of the `dladm` command have either been created or modified to work with link names. For more detailed information about `dladm` subcommands, refer to the `dladm(1M)` man page.

`show-phys`	Displays the device names and the physical attributes of each device. This subcommand displays the equivalent information as the `show-dev` subcommand. However, to leverage the use of link names, use the `show-phys` subcommand instead of the `show-dev` subcommand.
`rename-link`	Assigns a new link name to replace an existing link name.
`show-link`	Displays information about available data links in the system.
`create-vlan`	Configures a VLAN in the network.
`show-vlan`	Lists existing VLANs in the network.
`delete-vlan`	Removes an unused VLAN. A VLAN that is being used cannot be deleted.
`delete-phys`	Removes all link configurations that are associated with a removed NIC. This operation allows the link name to be used with another data-link with new link configurations.

Changes were also implemented on current `dladm` subcommands to enable the following operations to work with link names:

- Creating link aggregations.
- Administering `autopush` link properties.

PART I

Administering Single Interfaces

This part describes procedures to administer single interfaces in your system.

◆ ◆ ◆ CHAPTER 1

Network Driver Configuration

This chapter discusses how to customize properties of an Ethernet network driver to fulfill specific performance requirements.

What's New With Configuring Network Interface Card Drivers

In this release, configuration of properties of the NIC driver is performed by using the `dladm` command. This command allows you to configure the properties dynamically without causing any network disruption on other NICs of similar types. The values that you set are stored into a dladm repository and therefore persist even after you reboot the system or unplumb the interface.

A Driver Configuration Framework (GLDv3) is implemented in this release. If used when you configure drivers, this framework provides the following benefits:

- Only single command interface, `dladm`, is needed to configure network driver properties.
- A uniform syntax is used regardless of the properties: `dladm` *subcommand properties data-link*.
- 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 other network connections.

Overview of NIC Driver Properties

NIC driver 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 MTU size that can be applied to all data-link drivers.

- *Private properties* that are particular to a certain subset of 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.

Properties of a NIC driver are typically set with default values. However, certain networking scenarios might require you to change specific property settings of a NIC. These property settings can be either public or private properties. 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 given driver. In this Solaris release, all of these settings can now be reset by a single administrative tool, `dladm`.

`dladm` Subcommands to Administer NIC Properties

For NIC drivers that have been converted to the GLDv3 framework, properties are configured by using the `dladm` command. This command enables you to configure the properties dynamically without causing any network disruption on other NICs of similar types. The values that you set are stored in a `dladm` repository and persist even after you reboot the system or unplumb the interface. Therefore, use `dladm` as the preferred command to configure NICs, instead of the `ndd` command.

To administer NIC drivers, you use the following `dladm` subcommands:

- `dladm show-linkprop` displays the properties that are associated with the data link.
- `dladm set-linkprop` sets values for specified data-link properties.
- `dladm reset-linkprop` restores property settings to the default values.
- `dladm show-ether` displays Ethernet parameter settings of a data link.

For more information about these commands, see the `dladm(1M)` man page.

Note – Customizing NIC properties by using the `dladm` command is supported only in network drivers that have been converted to the GLDv3 framework, such as `bge`, `nge`, `e1000g`, and `nxge`.

Work continues to make other drivers become supported in the GLDv3 framework. To confirm whether your specific driver supports this feature, refer to the driver's man page.

The following section provides procedures to set certain NIC driver properties. The selected properties are public and common to all NIC drivers. A separate section describes driver specific properties as well as procedures to configure selected private properties of the `e1000g` driver.

Administering NIC Driver Properties

The following section provides procedures with examples that show how to configure public and private properties of NIC drivers by using the `dladm` command.

Task	Description	For Instructions
Modify the MTU size.	Increases the MTU size of packet transmission to handle Jumbo frames.	"How to Enable Support for Jumbo Frames" on page 27
Modify the link speed.	Switches off higher link speed and advertises only the lower link speed to allow communications with an older system.	"How to Change Link Speed Parameters" on page 29
Display information about NIC properties.	Lists NIC properties and their current configuration; lists Ethernet parameter settings.	"How to Obtain Status Information About NIC Properties" on page 30
Configure driver to use DMA binding.	Sets threshold that causes the driver to switch from DMA binding or `bcopy` function during transmission.	"How to Set the `e1000g` Driver to Use Direct Memory Access Binding" on page 33
Set interrupt rates	Manually defines rates at which interrupts are delivered by the driver instead of the rate being defined automatically.	"How to Manually Set the Interrupt Rate" on page 33

▼ How to Enable Support for Jumbo Frames

Enabling support for Jumbo frames in a network setup is a common task for most network scenarios. Support for Jumbo frames requires increasing the size of a data link's maximum transmission unit (MTU). The following procedure includes the use of customized names to identify data links. For an overview of customized names and their use in network configuration, see "Overview of the Networking Stack" on page 13.

1 On the system that has the link whose MTU you want to modify, assume the System Administrator role.

The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2 **To identify the specific Ethernet device whose MTU size you need to reset, display the links in the system.**

```
# dladm show-phys
```

Perform this step especially if your network configuration uses customized names for data links. With customized names, data links are no longer necessarily identified by their hardware-based names. For example, the Ethernet device is `bge0`. However, the data link over the device is renamed `net0`. Therefore, you would need to configure the MTU size of `net0`. Refer to "Data Link and IP Interface Configuration (Tasks)" on page 37 for examples of configuration tasks on data links that use customized names.

3 **(Optional) Display the data link's current MTU size and other properties.**

- **To display a specific property of a data link, use the following syntax:**

  dladm show-linkprop -p *property data-link*

  This command displays the settings of the property that you specify.

- **To display several selected properties of the data link, use the following syntax:**

  # dladm show-link *data-link*

  This command displays data-link information, including MTU size.

Note – See "Link Administration and Monitoring" on page 47 for additional examples of the use of the `dladm show-link` syntax to display data-link information.

---

4 **Unplumb the interface that is configured over the data link.**

 # ifconfig *interface* **unplumb**

5 **Change the value of the link's MTU size to 9000, the value for Jumbo frames.**

 # dladm set-linkprop -p mtu=9000 *data-link*

6 **Plumb the IP interface over the link.**

 # ifconfig *interface* **plumb** *IP-address* **up**

For additional options that you can use with the `ifconfig` command, see the `ifconfig(1M)` man page.

7 **(Optional) Verify that the interface uses the new MTU size by using one of the command syntaxes in Step 3.**

 # dladm show-linkprop -p mtu *data-link*

8 **(Optional) Display the link's current Ethernet settings.**

 # dladm show-ether *data-link*

Example 1–1 Enabling Support for Jumbo Frames

The following example that enables support for Jumbo frames builds on the following scenario:

- The system has two `bge` NICS: `bge0` and `bge1`.
- The device `bge0` is used as a primary interface, while the device `bge1` is used for test purposes.
- You want to enable support for Jumbo frames on `bge1`, while you retain the default MTU size of the primary interface.
- The network configuration uses customized names for data links. The link name of `bge0` is `net0`. The link name of `bge1` is `web1`.

```
# dladm show-phys
LINK       MEDIA        STATE     SPEED     DUPLEX     DEVICE
net0       ether        up        100Mb     full       bge0
itops1     ether        up        100Mb     full       qfe3
web1       ether        up        100Mb     full       bge1

# dladm show-linkprop -p mtu web1
LINK     PROPERTY     VALUE     DEFAULT     POSSIBLE
web1     mtu          1500      1500        --

# ifconfig web1 unplumb
# dladm set-linkprop -p mtu=9000 web1
# ifconfig web1 plumb 10.10.1.2/24 up

# dladm show-link web1
LINK     CLASS     MTU      STATE     OVER
web1     phys      9000     up        --
```

Notice that the MTU value is now 9000. In this example, the `dladm` command enabled you to change `web1`'s MTU size directly. The previous method would have required you to unplumb `net0` as well, which would have unnecessarily disrupted the primary interface's operations.

▼ How to Change Link Speed Parameters

Most network setups consist of a combination of systems with varying speed capabilities. For example, the advertised speed between an older system and a newer system might need to be changed to a lower setting to allow communication. By default, all the speed and duplex capabilities of a NIC card are advertised. This procedure shows how to turn off the gigabit capabilities and advertise only the megabit capabilities.

1. **On the system that has the NIC whose properties you want to modify, assume the System Administrator role.**

   The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2. **(Optional) Display the current status of the property you want to modify.**

   ```
   # dladm show-linkprop -p property data-link
   ```

3. **To advertise lower speed capabilities, turn off the higher speed capabilities to prevent them from being advertised.**

   ```
   # dladm set-linkprop -p property=value1 data-link
   ```

Example 1–2 Disabling Advertisement of a NIC's Gigabit Capabilities

This example shows how you can prevent the link `web1` from advertising gigabit capabilities.

```
# dladm show-linkprop -p adv_1000fdx_cap web1
LINK     PROPERTY             VALUE     DEFAULT     POSSIBLE
web1     adv_1000fdx_cap      1         --          1,0

# dladm show-linkprop -p adv_1000hdx_cap web1
LINK     PROPERTY             VALUE     DEFAULT     POSSIBLE
web1     adv_1000hdx_cap      1         --          1,0
```

The properties that advertise the link's gigabit capabilities are `adv_1000fdx_cap` and `adv_1000hdx_cap`. To disable these properties from being advertised, you would type the following commands:

```
# dladm set-linkprop -p adv_1000fdx_cap=0 web1
# dladm set-linkprop -p adv_1000hdx_cap=0 web1
```

Listing the Ethernet parameter settings would display the following output:

```
# dladm show-ether web1
LINK     PTYPE       STATE    AUTO  SPEED-DUPLEX            PAUSE
web1     current     up       yes   1G-f                    both
```

▼ How to Obtain Status Information About NIC Properties

You can obtain information about the NIC driver's properties by displaying either the Ethernet parameter settings or the link properties.

1. **On the system that has the NIC whose properties you want to modify, assume the System Administrator role.**

   The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2. **To obtain information about the Ethernet parameter settings, use the following command:**

   ```
   # dladm show-ether [-x] data-link
   ```

   where the `-x` option includes additional parameter information about the link. Without the `-x` option, only the current parameter settings are displayed.

3. **To obtain information about all the properties of the link, use the following command:**

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

Example 1–3 Displaying Ethernet Parameter Settings

This example displays an extended list of parameter information about a specified link.

```
# dladm show-ether -x web1
LINK     PTYPE       STATE    AUTO  SPEED-DUPLEX              PAUSE
web1     current     up       yes   1G-f                      both
--       capable     --       yes   1G-fh,100M-fh,10M-fh      both
--       adv         --       yes   100M-fh,10M-fh            both
--       peeradv     --       yes   100M-f,10M-f              both
```

With the `-x` option, the command also displays the built-in capabilities of the specified link, as well as the capabilities that are currently advertised between the host and the link partner. The following information is displayed:

- 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 host and link partner can send and receive pause frames.
- Regardless of the current setting, the capabilities of the Ethernet device are listed. The negotiation type can be set to automatic, 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 web1 are advertised as follows: autonegotiation, speed-duplex, and flow control of pause frames.
- Similarly, `web1`'s link or peer partner advertises the following capabilities: autonegotiation, speed-duplex, and flow control of pause frames.

Example 1–4 Displaying Link Properties

This example shows how to list all the properties of a link. If you want to display only a specific property, you use the `-p` option with the specific property that you want to monitor.

```
# dladm show-linkprop web1
LINK     PROPERTY             VALUE       DEFAULT     POSSIBLE
web1     speed                1000        --          --
web1     autopush             --          --          --
web1     zone                 --          --          --
web1     duplex               half        --          half,full
web1     state                unknown     up          up,down
web1     adv_autoneg_cap      1           1           1,0
web1     mtu                  1500        1500        --
web1     flowctrl             no          bi          no,tx,rx,bi
web1     adv_1000fdx_cap      1           1           1,0
web1     en_1000fdx_cap       1           1           1,0
web1     adv_1000hdx_cap      1           1           1,0
web1     en_1000hdx_cap       1           1           1,0
web1     adv_100fdx_cap       0           0           1,0
web1     en_100fdx_cap        0           0           1,0
web1     adv_100hdx_cap       0           0           1,0
web1     en_100hdx_cap        0           0           1,0
web1     adv_10fdx_cap        0           0           1,0
web1     en_10fdx_cap         0           0           1,0
web1     adv_10hdx_cap        0           0           1,0
web1     en_10hdx_cap         0           0           1,0
```

The settings for the speed and duplex capabilities of the link are manually configured on the enabled-speed properties which are labeled `en_*_cap`. For example, `en_1000fdx_cap` is the property for the gigabit full-duplex capability, and `en_100hdx_cap` is the property for the 100 megabits half-duplex capability. The settings of these enabled speed properties are advertised between the host and its link partner by corresponding advertised speed properties, which are labeled `adv_*_cap` such as `adv_1000fdx_cap` and `adv_100hdx_cap`.

Normally, the settings 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 values of the `adv_*_cap` properties might only be a subset of the values of the `en_*_cap` properties. For more details about the enabled and advertised speed properties, see the `dladm(1M)` man page.

How to Set the `e1000g` Driver to Use Direct Memory Access Binding

This procedure and the next procedure show how to configure private properties. Both procedures apply to properties specific to the `e1000g` driver. However, the general steps can be used to configure private properties of other NIC drivers as well.

Bulk traffic, such as file transfers, normally involves 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 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. If a fragment size is within the threshold, then `bcopy` mode is used, where the fragment data is copied to the preallocated transmit buffer.

To set the threshold, perform the following steps:

1. **On the system that has the NIC whose properties you want to modify, assume the System Administrator role.**

2. **Set the appropriate value for the `_tx_bcopy_threshold` property.**

   ```
   # dladm set-linkprop -p _tx_bcopy_threshold=value  e1000g-data-link
   ```

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

   Note – As with configuring public properties, the interface must also be unplumbed before private property settings can be modified.

3. **(Optional) Verify the new threshold value.**

   ```
   # dladm show-linkprop -p _tx_bcopy_threshold e1000g-data-link
   ```

How to Manually Set the Interrupt Rate

Parameters 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 parameters:

- `_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 parameter is enabled.

1. **On the system that has the NIC whose driver properties you want to modify, assume the System Administrator role.**

   The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2. **If necessary, identify the device whose driver property you want to modify.**

   ```
   # dladm show-phys
   ```

3. **Disable automatic tuning of the interrupt throttling rate.**

   ```
   # dladm set-linkprop -p _intr_adaptive=0 e1000g-data-link
   ```

Note – When automatic tuning of the interrupt throttling rate is enabled, then any value that is set for the parameter `_intr_throttling_rate` is ignored.

---

4. **Unplumb the network interface.**

5. **Set the value for the minimum inter interrupt level.**

   ```
   # dladm set-linkprop -p _intr_throttling_rate=value e1000g-data-link
   ```

Note – The default value of the `_intr_throttling_rate` parameter is 550 on SPARC® based systems and 260 on x86 based systems. Setting the minimum inter-interrupt level to 0 disables the interrupt throttling logic.

---

6. **Plumb the interface and configure an IP address for the interface.**

7. **(Optional) Display the threshold's new settings.**

Example 1–5 Configuring for DMA Binding and Setting the Interrupt Throttling Rate

This example uses an x86 based system with an `e1000g` NIC. The driver is configured with a threshold setting toggle between using DMA binding or the `bcopy` mode for transmitting packets. The setting for the interrupt throttling rate is also modified. Further, the `e1000g` data link has been renamed with a customized name. Therefore, the configuration is performed on the data link by referring to the customized name, `public0`.

```
# dladm show-phys
LINK       MEDIA        STATE     SPEED     DUPLEX     DEVICE
public0    ether        up        100Mb     full       e1000g0

# dladm show-linkprop -p _tx_bcopy_threshold public0
LINK        PROPERTY               VALUE     DEFAULT     POSSIBLE
public0     _tx_bcopy_threshold    512       512         --

# dladm show-linkprop -p _intr-throttling_rate
LINK        PROPERTY                 VALUE     DEFAULT     POSSIBLE
public0     _intr-throttling_rate    260       260         --

# ifconfig public0 unplumb
# dladm set-linkprop -p _tx_bcopy_threshold=1024 public0
# dladm set-linkprop -p _intr_adaptive=0 public0
# dladm set-linkprop -p _intr-throttling_rate=1024 public0
# ifconfig public0 plumb 10.10.1.2/24 up

# dladm show-linkprop -p _tx_bocopy_threshold=1024 public0
LINK        PROPERTY               VALUE     DEFAULT     POSSIBLE
public0     _tx_bcopy_threshold    1024      512         --

# dladm show-linkprop -p _intr_adaptive public0
LINK        PROPERTY          VALUE     DEFAULT     POSSIBLE
public0     _intr-adaptive    0         1           --

# dladm show-linkprop -p _intr-throttling_rate
LINK        PROPERTY                 VALUE     DEFAULT     POSSIBLE
public0     _intr-throttling_rate    1024      260         --
```

◆ ◆ ◆ CHAPTER 2

Configuring an IP Interface

This chapter provides the procedures that are used to configure a single data link and then an IP interface over that link.

About IP Interface Configuration

After you install the Solaris OS, you might perform the following tasks:

- Configure an IP interface over a data link for a basic interface configuration. This chapter describes the procedures.
- Configure links for a virtual local area network (VLAN). The procedures are described in Chapter 5, "Administering VLANs."
- Configure links into an aggregation. The procedures are described in Chapter 6, "Administering Link Aggregations."
- Configure IP interfaces as members of an IPMP groups. The procedures are described in Chapter 8, "Administering IPMP."

You use the appropriate `dladm` subcommands to administer data links such as assigning customized names to a link. For a description of link names, see "Assigning Names to Data Links" on page 15. If the data link is also intended to be used for IP communication, you then configure IP interfaces over the data link by using the `ifconfig` command.

Data Link and IP Interface Configuration (Tasks)

This section describes basic configuration procedures on a data link.

TABLE 2–1 Configuring Network Links (Task Map)

Task	Description	For Instructions
Sets 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 an Interface Is Unique" on page 38
Configure an IP interface over a data link.	Performs basic IP interface configuration.	"How to Configure an IP Interface After System Installation" on page 40
Replace a network interface card (NIC).	Changes NICs in a system during dynamic reconfiguration.	"How to Replace a Network Interface Card With Dynamic Reconfiguration" on page 43
Set per-link `autopush` properties.	Configures STREAMS modules to be pushed on top of a data link.	"How to Set STREAMS Modules on Data Links" on page 45
Rename a data link.	Changes the name of a link to any chosen name for better identification.	"How to Rename a Data Link" on page 47
Display physical attributes of a data link.	Lists physical information that underly a data link, including type of media, associated device instance, and other information.	"How to Display Information About Physical Attributes of Data Links" on page 48
Display state of data links.	Lists information about the status of data links.	"How to Display Data Link Information" on page 49
Remove a data link.	Removes a link configuration that is associated with a NIC no longer in use.	"How to Delete a Data Link" on page 50

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

Some applications require every interface on a host to have a unique MAC addresses. However, every SPARC based system has a system-wide MAC address, which by default is used by all interfaces. Here are two situations where you might want to configure the factory-installed MAC addresses for the interfaces on a SPARC system.

- For link aggregations, you should use the factory-set MAC addresses of the interfaces in the aggregation configuration.
- For IPMP groups, each interface in the group must have a unique MAC address. These interfaces must use their factory-installed MAC addresses.

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

1 On the system with the interfaces to be configured, assume the Primary Administrator role or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

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 change `local-mac-address?=false` to `local-mac-address?=true` for aggregations.

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

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

When you reboot the system, the interfaces with factory-installed MAC addresses now use these factory settings, rather than the system-wide MAC address. Interfaces without factory-set MAC addresses 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, all interfaces use the system-wide MAC address `8:0:20:0:0:1`.

```
ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
      inet 127.0.0.1 netmask ff000000
hme0: flags=1004843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
      inet 10.0.0.112 netmask ffffff80 broadcast 10.0.0.127
      ether 8:0:20:0:0:1
ce0: flags=1004843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
      inet 10.0.0.114 netmask ffffff80 broadcast 10.0.0.127
      ether 8:0:20:0:0:1
ce1: flags=1004843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
      inet 10.0.0.118 netmask ffffff80 broadcast 10.0.0.127
      ether 8:0:20:0:0:1
```

Note – Continue to the next step only if more than one network interface still has the same MAC address. Otherwise, go on to the final step.

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

Specify a unique MAC address in the `/etc/hostname.`*interface* file for the particular interface.

In the example in Step 4, you would need to configure `ce0` and `ce1` with locally administered MAC addresses. For example, to reconfigure `ce1` with the locally administered MAC address `06:05:04:03:02`, you would add the following line to `/etc/hostname.ce1`:

```
ether 06:05:04:03:02
```

Note – To prevent any risk of manually configured MAC addresses conflicting with other MAC addresses on your network, you must always configure *locally administered* MAC addresses, as defined by the IEEE 802.3 standard.

You also can use the `ifconfig ether` command to configure an interface's MAC address for the current session. However, any changes made directly with `ifconfig` are not preserved across reboots. Refer to the `ifconfig(1M)` man page for details.

6 **Reboot the system.**

▼ How to Configure an IP Interface After System Installation

Link configuration and IP interface configuration are two separate tasks. However, the following procedure combines the two tasks together to illustrate how an IP interface is created that has the same name as the data link. Thus, after plumbing an IP interface, for example, the link name is propagated up to the IP administrative and programmatic interfaces.

1 **On the system with the interface to be configured, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Display information about physical attributes of data links currently on the system.**

```
# dladm show-phys
```

For more information about this command, see How to Display Information About Physical Attributes of Data Links.

3 If you intend to rename a data link, then make sure that the link is not opened by any application.

For example, if the IP interface over the link is plumbed, then unplumb the interface.

```
# ifconfig interface unplumb
```

where *interface* refers to the IP interface that is plumbed and using the link.

4 (Optional) Assign a meaningful name to the data link.

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

old-linkname Refers to the current name of the data link. When a NIC is installed for the first time, by default, the NIC's link name is hardware-based, such as `bge0`.

new-linkname Refers to any name that you want to assign to the data link. For rules for assigning link names, refer to "Rules for Valid Link Names" on page 20.

Note – Although this step is optional, assigning a customized name to a link is recommended. For more information, see "Considerations for Working With Link Names" on page 18.

---

5 Configure the IP interface over the link with a valid IP address..

```
# ifconfig interface plumb IP-address up
```

where *interface* refers to the IP interface you are configuring over the link and the *IP-address* uses the the CIDR notation. The IP interface is identified by the name of the link. Thus, if you renamed the link in the previous step, you use the same name when you configure the IP interface. See the example that follows this procedure for reference.

This step also brings that IP address up and enables the IP interface.

For arguments that you can use with the `ifconfig` command such as `broadcast`, refer to the `ifconfig`(1M) man page. See also "Monitoring the Interface Configuration With the `ifconfig` Command" on page 65 for examples of the different usages of the command.

Note – This step and all subsequent steps that describe the configuration of an IP interface apply to IPv4 interfaces. To configure IPv6 interfaces, refer to "Configuring an IPv6 Interface" in *System Administration Guide: IP Services*

---

6 (Optional) Display the network data-link information.

```
# dladm show-link
```

For more information about this command, see "How to Display Data Link Information" on page 49.

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

```
# ifconfig interface
```

8 (Optional) To make the interface configuration persist across reboots, perform the following steps:

a. Create an `/etc/hostname.`*interface* file for the IP interface.

b. Using a text editor, edit the `/etc/hostname.`*interface* file by adding the IP address.

At a minimum, add the IPv4 address of the interface to the file. The address can be in traditional IPv4 notation or CIDR notation. For example, you can also use the following syntax:

```
# echo IP-address > /etc/hostname.interface
```

c. Add entries for the IP addresses into the `/etc/inet/hosts` file.

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

d. Reboot the system.

```
# reboot
```

Example 2–1 Configuring the Network Interface

This example uses partial information from Figure P–2 to configure the data link `qfe3` on the host `campus01`. The example also shows a persistent configuration.

```
# dladm show-phys
LINK     MEDIA        STATE     SPEED     DUPLEX     DEVICE
qfe3     Ethernet     up        100Mb     full       qfe3

# ifconfig qfe3 unplumb
# dladm rename-link qfe3 subitops1
# ifconfig subitops1 plumb 192.168.84.3/24 up
# dladm show-link
LINK          CLASS     MTU     STATE     OVER
subitops1     phys      1500    up        --

# ifconfig subitops1
subitops1: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.84.3 netmask ffffff00 broadcast 192.168.84.255
        ether 8:0:20:c8:f4:1d

# echo 192.168.84.3/24 > /etc/hostname.subitops1

# vi /etc/inet/hosts
```

```
# Internet host table
#
127.0.0.1       localhost
10.0.0.14       myhost
192.168.84.3       campus01

# reboot
```

How to Replace a Network Interface Card With Dynamic Reconfiguration

Aside from servers, most laptops have PCMCIA slots that support dynamic reconfiguration (DR). This procedure shows how DR is now facilitated by the separation of the network link configuration from the network hardware configuration. 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 be inherited by the replacement NIC.

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, search for `dynamic reconfiguration` on `http://docs.sun.com`

Note – The following procedure refers only to aspects of DR that are specifically related to the use of flexible names for data links. The procedure does not contain the complete steps to perform DR. You must consult the appropriate DR documentation for your system.

---

1. **Assume the proper role that allows you to perform DR on the system, such as the Primary Administrator role or superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **(Optional) Display information about physical attributes of data links currently on the system.**

   ```
   # dladm show-phys
   ```

3. **Perform the DR procedures as detailed in your system's documentation to remove a NIC and then insert a replacement NIC.**

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

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

4. **Make sure that the replacement NIC is not being referenced by other configurations in the system.**

   For example, the replacement NIC you install is `ce0`. If a file `/etc/hostname.ce0` exists in the system, remove that file.

5. **Transfer the link configuration of the removed NIC to the replacement NIC.**

   `# dladm rename-link` *replacementNIC-linkname removedNIC-linkname*

replacementNIC-linkname	Refers to the default link name of the replacement NIC upon installation. When you insert a NIC into a system for the first time, the NIC's link name is hardware-based, such as `bge0` or `ce0`.
removedNIC-linkname	Refers to the customized link name of the NIC that you removed.

6. **Complete the DR process by enabling the new NIC's resources to become available for use by the Solaris release.**

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

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

   For example, you can use either `dladm show-phys` or `dladm show-link` to show information about the data links.

Example 2–2 Replacing a Network Card

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

```
# dladm show-phys
LINK          MEDIA        STATE     SPEED     DUPLEX     DEVICE
subitops1     Ethernet     up        100Mb     full       qfe3
net0          Ethernet     up        100Mb     full       bge0
```

You perform the DR-specific steps such as using `cfgadm` to disconnect `bge` and then install `ce`. Then the procedure continues.

```
# ls /etc/hostname.*
hostname.ce0

# rm /etc/hostname.ce0
# dladm rename-link ce0 net0

# dladm show-phys
LINK        MEDIA        STATE     SPEED     DUPLEX     DEVICE
subitops1   Ethernet     up        100Mb     full       qfe3
net0        Ethernet     up        100Mb     full       ce0
```

Configuring STREAMS Modules on Data Links

If necessary, you can set up to eight STREAMS modules to be pushed on top of a data link. 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 STREAMS modules to push on a specific data link is controlled by the `autopush` link property. In turn, the value of the `autopush` link property is set by using the `dladm set-linkprop` subcommand.

A separate `autopush` command can also be used to set the STREAMS `autopush` modules on a per-driver basis. However, the driver is always bound to the NIC. If the data link's underlying NIC is removed, then the link's autopush property information becomes lost as well.

To configure the STREAMS modules to be pushed on top of a data link, use the `dladm set-linkprop` command in preference over the `autopush` command. If both per-driver and per-link types of `autoputsh` configuration exist for a specific data link, the per-link information that is set with `dladm set-linkprop` is used and the per-driver information is ignored.

▼ How to Set STREAMS Modules on Data Links

The following procedure describes how to configure STREAMS modules with the `dladm set-linkprop` command.

1 Assume the System Administrator role.

The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2 **Push the modules to the stream when the link is opened.**

```
# dladm set-linkprop -p autopush=modulelist  link
```

modulelist Specifies the list of modules that you want to be automatically pushed on to the stream. A maximum of eight modules can be pushed over a link. These modules are pushed in the order that they are listed in *modulelist*. Separate the modules in the list by using dots as delimiters.

link Specifies the link on which the modules are pushed.

Example 2–3 Setting the `autopush` Link Property

In this example, you push the `vpnmod` and `bufmod` modules on top of the link `net0`. The link's underlying device is `bge0`.

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

If you later replace the `bge` card with `ce`, you can switch to the new data link without needing to reconfigure the `autopush` settings. You just assign the link name to the new data link, as follows:

```
# dladm rename-link ce0 net0
```

▼ How to Obtain `autopush` Link Property Settings

1 **Assume the System Administrator role.**

The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2 **Display `autopush` link property settings.**

```
# dladm show-linkprop -p autopush [link]
```

If you do not specify *link*, then the information for all configured links is displayed.

▼ How to Remove `autopush` Link Property Settings

1 **Assume the System Administrator role.**

The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2. **Remove the `autopush` link property settings of a specific data link.**

   ```
   # dladm reset-linkprop [-t] -p autopush link
   ```

   Use the `-t` option to remove the property settings temporarily. The settings are restored when you reboot the system.

Link Administration and Monitoring

New and revised `dladm` subcommands are now available to work with link names. This section specifically discusses the subcommands `show-phys` and `show-link`. Other commands that display information are discussed in their respective chapters that describe the specific network setups, such as link aggregations and VLAN configurations.

▼ How to Rename a Data Link

Use this procedure if you want to change a data link name, for example, to assign a meaningful name that would easily identify the link's role within your network setup.

Before You Begin Make sure that you have studied and prepared for other steps you need to perform on associated configurations that might be affected by the change of link names. For more information, see "Considerations for Working With Link Names" on page 18.

1. **On the system on which you want to rename a data link, assume the System Administrator role.**

   The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2. **Unplumb the IP interface.**

   ```
   # ifconfig interface unplumb
   ```

3. **Change the link's current link name.**

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

   old-linkname Refers to the current name of the data link. By default, the link name is hardware-based, such as `bge0`.

   new-linkname Refers to any name that you want to assign to the data link. For rules for assigning link names, refer to "Rules for Valid Link Names" on page 20. See also "Considerations for Working With Link Names" on page 18 for further information about renaming data links.

If you do not want the new link name to persist across a system reboot, then use the `-t` option immediately after the subcommand. The option renames a link temporarily. The original link name reverts when the system is rebooted.

Note – You can use `dladm rename-link` to transfer link configurations from one data link to another. For an example, see "How to Replace a Network Interface Card With Dynamic Reconfiguration" on page 43. When you rename a link for this purpose, make sure that the link that is inheriting the configuration does not have any prior existing configurations. Otherwise, the transfer fails.

Example 2–4 Changing a Link Name

The following example shows how a link name is changed from a hardware-based name to a customized name.

```
# dladm rename-link bge0 net0
```

▼ How to Display Information About Physical Attributes of Data Links

This procedure lists the steps to display information about the physical attributes of a system's data links.

1 On the system, assume the System Administrator role.

The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2 Display information about physical attributes of data links currently on the system.

```
# dladm show-phys -P
```

You can use the `-P` with this command to also display data links that are flagged as unavailable. A data link becomes unavailable if its associated hardware has been removed. Without the `-P` option, the command displays only available data links.

To view the `/devices` path of the data links, use the `-v` option.

Example 2–5 Displaying Available Data Links

In the following example, the `-P` option includes the `FLAGS` column where unavailable links are indicated. The `r` flag for the data link `net0` indicates the hardware that is associated with the link (`eri`) has been removed.

```
# dladm show-phys -P
LINK        MEDIA          STATE     SPEED     DUPLEX     DEVICE   FLAGS
subitops1   Ethernet       up        100Mb     full       qfe3     -----
ibd0        Infiniband     down      0Mb       --         ibd0     -----
subitops0   Ethernet       up        100Mb     full       ce0      -----
net0        Ethernet       --        0Mb       --         eri0     r----
```

The following example shows the output that is generated when you use the `-v` option.

```
# dladm show-phys -v
LINK     PATH
net2     /pci@1f,700000/network@2
ibd3     /pci@1d,700000/network@2
bge3     /pci@1f,700000/network@2,1
```

▼ How to Display Data Link Information

This procedure displays the status of available links.

1. **Assume the System Administrator role.**

   The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2. **Display link information.**

   ```
   # dladm show-link
   ```

Example 2–6 Displaying Available Links

The following example shows persistent and available links on the system.

```
# dladm show-link -P
LINK        CLASS     OVER
subitops1   phys      --
ibd0        phys      --
eri0        phys      --
```

The `-P` option also displays any existing persistent but unavailable links. A persistent link becomes unavailable if the link is temporarily deleted. A link also becomes unavailable if the associated hardware has been removed.

How to Delete a Data Link

This procedure deletes link configurations that are associated with NICs. If you detach a NIC without intending to replace it, then you can delete the link configuration that is associated with that NIC. After you complete this procedure, the link name can be reused.

1. **On the system with the interfaces to be configured, assume the System Administrator role.**

   The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2. **Display the data links on the system including those links whose hardware have been removed.**

   To include information about removed hardware, use the `-P` option.

   ```
   # dladm show-phys -P
   ```

3. **Remove the link configuration of the removed hardware that you do not intend to replace.**

   ```
   # dladm delete-phys link
   ```

Example 2–7 Deleting a Data Link

In the following example, the `r` flag for `net0` indicates that the link's associated hardware (`ce`) has been removed. Therefore, you can also remove the link `net0` and then reassign the name to a new data link.

```
# dladm show-phys -P
LINK        DEVUCE      MEDIA        FLAGS
mylink0     qfe0        Ethernet     -----
eri0        eri0        Ethernet     -----
net0        ce0         Ethernet     r----

# dladm delete-phys net0
```

◆ ◆ ◆ CHAPTER 3

Configuring Wireless Interface Communications on the Solaris OS

This chapter explains how to configure and use wireless interface communications on a laptop that runs the Solaris OS. The following topics are covered:

- Communicating over WiFi Interfaces
- Finding a WiFi Network
- Connecting and Using WiFi on Solaris OS Systems
- Secure WiFi Communications

WiFi Communications Task Map

Task	Description	For Instructions
Plan for WiFi communications on your system.	Set up your laptop or wireless network configuration, optionally including a router, in a location that supports WiFi	"How to Prepare a System for WiFi Communications" on page 53
Connect to a WiFi network	Set up and establish communications with a local WiFi network	"How to Connect to a WiFi Network" on page 54
Monitor communications on the WiFi link	Use standard Solaris networking tools to check the state of WiFi link	"How to Monitor the WiFi Link" on page 59
Establish secure WiFi communications	Create a WEP key and use it establish connections with a secure WiFi network	"How to Set Up an Encrypted WiFi Network Connection" on page 61

Communicating Over WiFi Interfaces

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.

The Solaris OS contains features that enable you to configure a system as a WiFi client. This section explains how to use the WiFi connectivity options of the `dladm` command to connect a laptop or home computer to a local WiFi network.

Note – The Solaris OS does not contain features for configuring WiFi servers or access points.

Finding a WiFi Network

WiFi networks typically come in three varieties:

- Commercially available WiFi networks
- Municipal WiFi networks
- Private WiFi networks

A location that is served by WiFi is referred to as a *hot spot*. Each hot spot includes an access point. The *access point* is a router with a "wired" connection to the Internet, for example, Ethernet or DSL. The Internet connection is usually through a wireless Internet service provider (WISP) or traditional ISP.

Commercial WiFi Networks

Many hotels and cafes offer wireless Internet connections as a service to their customers with laptop computers. These commercial hot spots have access points within their facilities. The access points are routers with wired connections to a WISP that serves commercial locations. Typical WISPs include independent providers and cellular phone companies.

You can use a laptop that runs the Solaris OS to connect to a WiFi network that is offered by a hotel or other commercial hot spot. Ask for instructions at the hot spot for connecting to the WiFi network. Typically, the connection process involves supplying a key to a browser that you launch upon login. You might have to pay a fee to the hotel or WISP in order to use the network.

Commercial locations that are Internet hot spots usually advertise this capability to their patrons. You can also find lists of wireless hot spots from various web sites, for example, Wi-FiHotSpotList.com (`http://www.wi-fihotspotlist.com`).

Municipal WiFi Networks

Cities throughout the world have constructed free municipal WiFi networks, which their citizens can access from systems in their homes. Municipal WiFi uses radio transmitters on telephone poles or other outdoor locations to form a "mesh" over the area that the network serves. These transmitters are the access points to the municipal WiFi network. If your area is served by a municipal WiFi network, your home might be included in the network's mesh.

Access to municipal WiFi is usually free. You can access the municipal network from a properly equipped laptop or personal computer that runs the Solaris OS. You do not need a home router to access the municipal network from your system. However, configuring a home router is recommended for areas where the signal from the municipal network is weak. Home routers are also recommended if you require secure connections over the WiFi network. For more information, see "Secure WiFi Communications" on page 60.

Private WiFi Networks

Because WiFi networks are relatively easy to configure, companies and universities use private WiFi networks with access limited to employees or students. Private WiFi networks typically require you to supply a key when you connect or run a secure VPN after you connect. You need a properly equipped laptop or PC that runs the Solaris OS and permission to use the security features in order to connect to the private network.

Planning for WiFi Communications

Before you can connect your system to a WiFi network, complete the following instructions.

▼ How to Prepare a System for WiFi Communications

Before You Begin The following preparations assumes that your system is a laptop or personal computer that runs the Solaris Express, Developer Edition 2/07 or later release.

1 Equip your system with a supported WiFi interface.

Your system must have a WiFi card that is supported by Solaris. For the Solaris Express, Developer Edition 2/07 and later releases, you can use WiFi cards that support most Atheros chip sets. For a list of currently supported drivers and chip sets, refer to Wireless Networking for OpenSolaris (`http://opensolaris.org/os/community/laptop/wireless`).

If the interface is not already present on the system, follow the manufacturer's instructions for installing the interface card. You configure the interface software during the procedure "How to Connect to a WiFi Network" on page 54.

2 **Locate your system in a place that is served by a WiFi network, either commercial, municipal, or private.**

Your system must be near the access point for the network, which is normally not a consideration for a commercial or private network hot spot. However, if you plan to use a free municipal network, your location must be near the transmitter access point.

3 **(Optional) Set up a wireless router to serve as an additional access point.**

Set up your own router if no WiFi network is available at your location. For example, if you have a DSL line, connect the wireless router to the DSL router. Then the wireless router becomes the access point for your wireless devices.

Connecting and Using WiFi on Solaris OS Systems

This section contains tasks for establishing and monitoring WiFi connections for a laptop or desktop computer that runs the Solaris OS.

▼ How to Connect to a WiFi Network

Before You Begin The following procedure assumes that you have followed the instructions in "How to Prepare a System for WiFi Communications" on page 53.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Check for available links.**

```
# dladm show-link
LINK        CLASS     MTU    STATE    OVER
ath0        phys      1500   up          --
e1000g0     phys      1500   up          --
```

In this example, the output indicates that two links are available. The `ath0` link supports WiFi communications beginning with the Solaris Express, Developer Edition 2/07release. The `e1000g` link is for attaching the system to a wired network.

3 **Configure the WiFi interface.**

Use the following steps to configure the interface:

- Plumb the link that supports WiFi:

```
# ifconfig ath0 plumb
```

- Verify that the link has been plumbed:

```
# ifconfig -a

lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL>
 mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g: flags=2001004802<BROADCAST,RUNNING,MULTICAST,DHCP,IPv4,CoS>
 mtu 1500 index 2
        inet 0.0.0.0 netmask 0
        ether 0:e:6:4:8:1
ath0: flags=201000803<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS>
 mtu 1500 index 3
        inet 0.0.0.0 netmask ff000000
        ether 0:b:6:e:f:18
```

4 Check for available networks.

```
# dladm scan-wifi
 LINK       ESSID         BSSID/IBSSID       SEC     STRENGTH   MODE   SPEED
 ath0       net1          00:0e:38:49:01:d0  none    good       g      54Mb
 ath0       net2          00:0e:38:49:02:f0  none    very weak  g      54Mb
 ath0       net3          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`	Link name to be used in the WiFi connection.
`ESSID`	Extended Service Set ID. The `ESSID` is the name of the WiFi network, such as `net1`, `net2`, and `net3` in the example output.
`BSSID/IBSSID`	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`	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 60.
`STRENGTH`	Strength of the radio signals from the WiFi networks that are available at your location.
`MODE`	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`	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.

```
# ifconfig interface dhcp start
```

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

```
ifconfig: 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.

```
# ifconfig interface IP-address/CIDR-mask | netmask
```

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

```
# dladm show-wifi
LINK       STATUS        ESSID         SEC     STRENGTH   MODE   SPEED
ath0       connected     net3          none    very good  g      36Mb
```

In this example, the output indicates that the system is now connected to the `net3` network. The earlier `scan-wifi` output indicated that `net3` had the strongest signal among the available networks. The `dladm show-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 hot spot 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 was 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 3–1 Connecting to a Specific WiFi Network

The following example shows a typical scenario that you might encounter when using a laptop that runs the Solaris Express, Developer Edition 2/07 or later Developer releases in an Internet coffee house.

Learn whether a WiFi link is available.

```
# dladm show-wifi
ath0            type: non-vlan    mtu: 1500        device: ath0
```

The `ath0` link is installed on the laptop. Configure the `ath0` interface, and verify that it is up.

```
# ifconfig ath0 plumb
# ifconfig -a
lo0: flags=2001000849<LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
ath0: flags=201000803<BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 3
        inet 0.0.0.0 netmask ff000000
        ether 0:b:6b:4e:8f:18
```

Display the available WiFi links at your location.

```
# dladm scan-wifi
 LINK       ESSID         BSSID/IBSSID       SEC     STRENGTH   MODE   SPEED
 ath0       net1          00:0e:38:49:01:d0  none    weak       g      54Mb
 ath0       net2          00:0e:38:49:02:f0  none    very weak  g      54Mb
 ath0       net3          00:0d:ed:a5:47:e0  wep     very good  g      54Mb
 ath0       citinet       00:40:96:2a:56:b5  none    good       b      11Mb
```

The output indicates that `net3` has the best signal. `net3` requires a key, for which the provider for the coffee house charges a fee. `citinet` is a free network provided by the local town.

Connect to the `citinet` network.

```
# dladm connect-wifi -e citinet
```

The `-e` option of `connect-wifi` takes the `ESSID` of the preferred WiFi network as its argument. The argument in this command is `citinet`, the `ESSID` of the free local network. The `dladm connect-wifi` command offers several options for connecting to the WiFi network. For more information, refer to the `dladm(1M)` man page.

Configure an IP address for the WiFi interface.

```
# ifconfig ath0 10.192.16.3/24 up
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL>
 mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0: flags=201004843<UP,,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4,CoS>
 mtu 1500 index 3
        inet 129.146.69.34 netmask fffffe00 broadcast 129.146.69.255
        ether 0:e:7b:b5:64:a4
ath0: flags=201004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4,CoS>
 mtu 1500 index 4
        inet 10.192.16.3 netmask ffffff00 broadcast 10.255.255.255
        ether 0:b:6b:4e:8f:18
```

This example assumes that you have the static IP address `10.192.16.3/24` configured on your laptop.

```
# dladm show-wifi
LINK       STATUS        ESSID         SEC     STRENGTH   MODE   SPEED
ath0       connected     citinet       none    good       g      11Mb
```

The output indicates that the laptop is now connected to network `citinet`.

```
# firefox
```

The home page for the Firefox browser displays.

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

```
# dladm disconnect-wifi
# dladm show-wifi
LINK        STATUS        ESSID         SEC     STRENGTH   MODE   SPEED
ath0        disconnected   --            --       --        --      --
```

The output of `show-wifi` verifies that you have disconnected the `ath0` link from the WiFi network.

How to Monitor the WiFi Link

This procedure shows how to monitor the status of a WiFi link through standard networking tools, and change link properties through the `linkprop` subcommand.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Connect to the WiFi network, as described in "How to Connect to a WiFi Network" on page 54.**

3. **View the properties of the link.**

   Use the following syntax:

   `# dladm show-linkprop` *interface*

   For example, you would use the following syntax to show the status of the connection established over the `ath0` link:

```
# dladm show-linkprop ath0
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 – The Solaris OS 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 ath0 -i 5
   input   ath0      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 3–2 Set 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 ath0
PROPERTY        VALUE           DEFAULT         POSSIBLE
speed           24               --              1,2,5,6,9,11,12,18,24,36,48,54
# dladm set-linkprop -p speed=36 ath0

# dladm show-linkprop -p speed ath0
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 shows 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. This procedure assumes that you use such a router and have its documentation available. The procedure also assumes that your system is already plugged into the router.

1 Start the software for configuring the home router.

Refer to the manufacturer's documentation for instructions. Router manufacturers typically offer an internal web site or a graphical user interface for router configuration.

2 Generate the value for the WEP key.

Follow the manufacturer's instructions for creating a secure key for the router. The router configuration GUI might ask you to supply a passphrase of your choice for the key. The software then uses the passphrase to generate a hexadecimal string, typically 5 bytes or 13 bytes in length. This string becomes the value to be used for the WEP key.

3 Apply and save the key configuration.

Refer to the manufacturer's documentation for instructions.

4 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

5 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.

6 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 five byte or thirteen byte string, in ASCII or in hexadecimal for the key value.

7 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.

8 Make an encrypted connection to the WiFi network.

```
# dladm connect-wifi -e network -k keyname interface
```

9 Verify that the connection is secure.

```
# dladm show-wifi
LINK       STATUS        ESSID         SEC     STRENGTH   MODE   SPEED
ath0       connected     net1          wep     good       g      11Mb
```

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

Example 3–3 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

```
# dladm create-secobj -c wep mykey
provide value for mykey: *****
confirm value for mkey: *****
```

When you supply the WEP key generated that is by the router, asterisks mask the value that you type.

```
# dladm show-secobj
OBJECT              CLASS
 mykey              wep
```

```
# dladm connect-wifi -e citinet -k mykey ath0
```

This command establishes an encrypted connection to the WiFi network `citinet`, using the secure object `mykey`.

```
# dladm show-wifi
LINK       STATUS        ESSID         SEC     STRENGTH   MODE   SPEED
ath0       connected     citinet       wep     good       g      36Mb
```

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

◆ ◆ ◆ CHAPTER 4

Troubleshooting Common Problems with Interfaces and Links

This chapter discusses different commands to monitor the status of data links and IP interfaces in your systems.

Monitoring the Interface Configuration With the `ifconfig` Command

You use the `ifconfig` command to manually assign IP addresses to interfaces and to manually configure interface parameters. In addition, the Solaris startup scripts run `ifconfig` to configure pseudo interfaces, such as 6to4 tunnel endpoints.

This book contains many tasks that use the various options of the versatile `ifconfig` command. For a complete description of this command, its options, and its variables, refer to the `ifconfig(1M)` man page. The basic syntax of `ifconfig` follows:

`ifconfig` *interface [protocol-family]*

▼ How to Get Information About a Specific Interface

Use the `ifconfig` command to determine basic information about the interfaces of a particular system. For example, a simple `ifconfig` query can tell you the following:

- Device names of all interfaces on a system
- All IPv4 and, if applicable, all IPv6 addresses that are assigned to the interfaces
- Whether these interfaces are currently configured

The following procedure shows how to use the `ifconfig` command to obtain basic configuration information about a system's interfaces.

1 **On the local host, assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Obtain information about a particular interface.**

```
# ifconfig interface
```

The output from the ifconfig command has the following format:

- Status line

  The first line in the `ifconfig` command output includes the interface name and status flags currently associated with the interface. Also, the status line includes the maximum transmission unit (MTU) that is configured for the particular interface and an index number. Use the status line to determine the current state of the interface.

- IP address information line

  The second line of the `ifconfig` output includes the IPv4 address or IPv6 address that is configured for the interface. For an IPv4 address, the configured netmask and broadcast address are also displayed.

- MAC address line

  When you run the `ifconfig` command as superuser or with a similar role, the `ifconfig` output contains a third line. For an IPv4 address, the third line shows the MAC address (Ethernet layer address) that is assigned to the interface. For an IPv6 address, the third line in the output shows the link-local address that the IPv6 `in.ndpd` daemon generates from the MAC address.

Example 4–1 Basic Interface Information From the `ifconfig` Command

The following example shows how to obtain information about the `eri` interface on a particular host by using the `ifconfig` command.

```
# ifconfig eri
eri0: flags=863<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 1
      inet 10.0.0.112 netmask ffffff80 broadcast 10.8.48.127
      ether 8:0:20:b9:4c:54
```

The next table describes the variable information in an `ifconfig` query. The preceding output is used as an example.

Variable	Screen Output	Description
Interface name	`eri0`	Indicates the device name of the interface whose status was requested in the `ifconfig` command.
Interface status	`flags=863<UP`	Displays the status of the interface, including any flags that are currently associated with the interface. Here you can determine whether the interface is currently initialized (`UP`) or not initialized (`DOWN`).
Broadcast status	`BROADCAST`	Indicates that the interface supports IPv4 broadcasts.
Transmission status	`RUNNING`	Indicates that the system is transmitting packets through the interface.
Multicast status	`MULTICAST, IPv4`	Shows that the interface supports multicast transmissions. The example interface supports IPv4 multicast transmissions.
Maximum transmission unit	`mtu 1500`	Shows that this interface has a maximum transfer size of 1500 octets.
IP address	`inet 10.0.0.112`	Displays the IPv4 or IPv6 address that is assigned to the interface. Example interface `eri0` has the IPv4 address `10.0.0.112`.
Netmask	`netmask ffffff80`	Displays the IPv4 netmask of the particular interface. Note that IPv6 addresses do not use netmasks.
MAC address	`ether 8:0:20:b9:4c:54`	Shows the interface's Ethernet layer address.

How to Display Interface Address Assignments

Routers and multihomed hosts have more than one interface and, often, more than one IP address assigned to each interface. You can use the `ifconfig` command to display all addresses that are assigned to the interfaces of a system. You can also use the `ifconfig` command to display only IPv4 or IPv6 address assignments. To additionally display the MAC addresses of the interfaces, you must first log in as superuser or assume the appropriate role.

For more information on the `ifconfig` command, see the `ifconfig(1M)` man page.

1 On the local system, assume the Network Management role or become superuser.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 Obtain information about all interfaces.

You can use variations of the `ifconfig -a` command to do the following:

- View all addresses of all interfaces on the system.

```
# ifconfig -a
```

- View all IPv4 addresses that are assigned to a system's interfaces.

```
# ifconfig -a4
```

- If the local system is IPv6-enabled, display all IPv6 addresses that are assigned to a system's interfaces.

```
ifconfig -a6
```

Example 4–2 Displaying Addressing Information for All Interfaces

This example shows entries for a host with solely a primary network interface, `qfe0`. Nevertheless, the `ifconfig` output shows that three forms of addresses are currently assigned to `qfe0`: loopback (`lo0`), IPv4 (`inet`), and IPv6 (`inet6`). In the IPv6 section of the output, note that the line for interface `qfe0` displays the link-local IPv6 address. The second address for `qfe0` is displayed on the `qfe0:1` line.

```
% ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1004843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.0.0.112 netmask ffffff80 broadcast 10.0.0.127
        ether 8:0:20:b9:4c:54
lo0: flags=2000849 <UP,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841 <UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b9:4c:54
        inet6 fe80::a00:20ff:feb9:4c54/10
qfe0:1: flags=2080841 <UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 2001:db8:3c4d:48:a00:20ff:feb9:4c54/64
```

Example 4–3 Displaying Addressing Information for All IPv4 Interfaces

This example shows the IPv4 address that is configured for a multihomed host. You do not need to be logged in as superuser to run this form of the `ifconfig` command.

```
% ifconfig -a4
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1004843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.0.0.112 netmask ffffff80 broadcast 10.0.0.127
        ether 8:0:20:b9:4c:54
qfe1: flags=1004843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.0.0.118 netmask ffffff80 broadcast 10.0.0.127
        ether 8:0:20:6f:5e:17
```

Example 4–4 Displaying Addressing Information for All IPv6 Interfaces

This example shows only the IPv6 addresses that are configured for a particular host. You do not need to be logged in as superuser to run this form of the `ifconfig` command.

```
% ifconfig -a6
lo0: flags=2000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841 <UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b9:4c:54
        inet6 fe80::a00:20ff:feb9:4c54/10
qfe0:1: flags=2080841 <UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 2001:db8:3c4d:48:a00:20ff:feb9:4c54/64
```

This output from `ifconfig` shows the following three types of IPv6 address forms that are assigned to the single interface of a host:

`lo0`
: IPv6 loopback address.

`inet6 fe80::a00:20ff:feb9:4c54/10`
: Link-local address that is assigned to the primary network interface.

`inet6 2001:db8:3c4d:48:a00:20ff:feb9:4c54/64`
: IPv6 address, including subnet prefix. The term `ADDRCONF` in the output indicates that this address was autoconfigured by the host.

Monitoring IPMP Interfaces

In this Solaris release, IP multipathing (IPMP) groups are represented as IPMP interfaces. The IPMP interface is treated just like any other IP interface on the IP layer. Thus, the `ifconfig` command can still be used to provide information about the IPMP group.

However, a new command, `ipmpstat`, that is more IPMP-specific has been introduced in this release. This command combined with different options can provide information about the IPMP group, the group's underlying interfaces, data addresses, test addresses, and the state of group as a whole. Thus, to obtain information about IPMP groups, use the `ipmpstat` command instead of `ifconfig`.

Specific ways of using the `ipmpstat` command are provided in "Monitoring IPMP Information" on page 143.

Monitoring Link Configurations

The `dladm` *subcommand* command is the administrative tool to use to configure data links on the link layer of the networking stack. You also use `dladm` to monitor links and obtain information such as the link states, associated physical devices, and other information specific

to the type of link configuration. For more information about monitoring links, see "Link Administration and Monitoring" on page 47. You can also refer to the specific chapters that discuss types of link configurations, such as "Performing Other Administrative Tasks on VLANs" on page 79 and Chapter 6, "Administering Link Aggregations."

PART II

Administering Interface Groups

This part discusses administration of other types of configurations such as virtual local area networks (VLANs), link aggregations, and IP multipathing (IPMP) groups.

◆ ◆ ◆ CHAPTER 5

Administering VLANs

This chapter describes procedures to configure and maintain virtual local area networks (VLANs). The procedures include steps that avail of features such as support for flexible link names.

Administering Virtual Local Area Networks

A *virtual local area network (VLAN)* is a subdivision of a local area network at the data link layer of the TCP/IP protocol stack. You can create VLANs for local area networks that use switch technology. By assigning groups of users to VLANs, you can improve network administration and security for the entire local network. You can also assign interfaces on the same system to different VLANs.

Consider dividing your local network into VLANs if you need to do the following:

- Create a logical division of workgroups.

  For example, suppose all hosts on a floor of a building are connected on one switched-based local network. You could create a separate VLAN for each workgroup on the floor.
- Enforce differing security policies for the workgroups.

  For example, the security needs of a Finance department and an Information Technologies department are quite different. If systems for both departments share the same local network, you could create a separate VLAN for each department. Then, you could enforce the appropriate security policy on a per-VLAN basis.
- Split workgroups into manageable broadcast domains.

  The use of VLANs reduces the size of broadcast domains and improves network efficiency.

Overview of VLAN Topology

Switched LAN technology enables you to organize the systems on a local network into VLANs. Before you can divide a local network into VLANs, you must obtain switches that support VLAN technology. You can configure all ports on a switch to serve a single VLAN or multiple VLANs, depending on the VLAN topology design. Each switch manufacturer has different procedures for configuring the ports of a switch.

The following figure shows a local area network that has the subnet address `192.168.84.0`. This LAN is subdivided into three VLANs, Red, Yellow, and Blue.

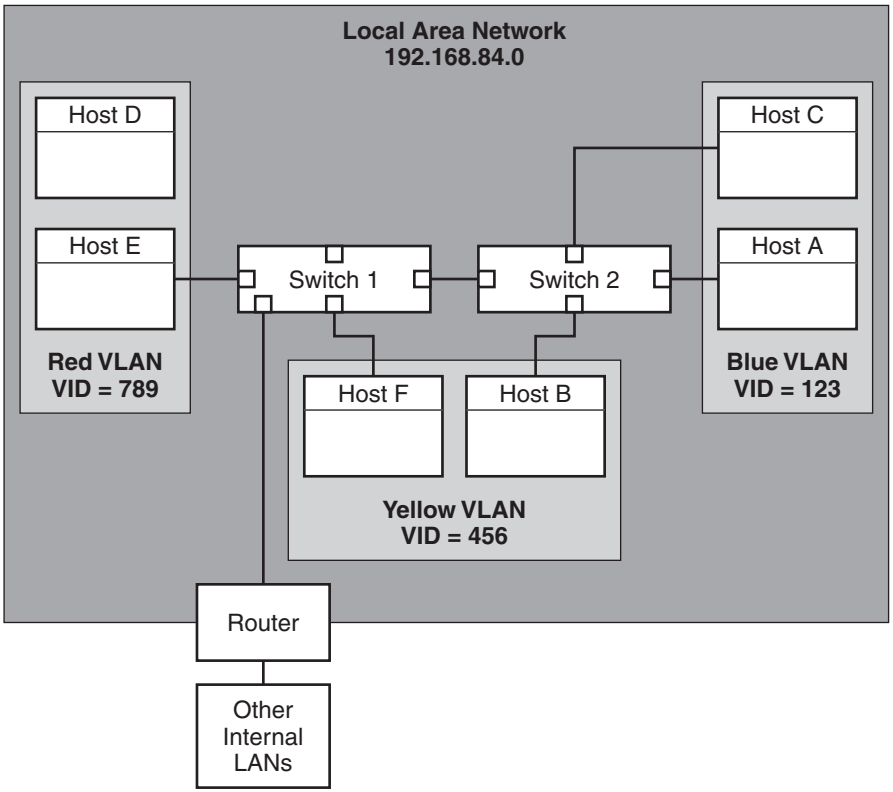

FIGURE 5–1 Local Area Network With Three VLANs

Connectivity on LAN `192.168.84.0` is handled by Switches 1 and 2. The Red VLAN contains systems in the Accounting workgroup. The Human Resources workgroup's systems are on the Yellow VLAN. Systems of the Information Technologies workgroup are assigned to the Blue VLAN.

VLAN Tags and Physical Points of Attachment

Each VLAN in a local area network is identified by a VLAN tag, or *VLAN ID (VID)*. The VID is assigned during VLAN configuration. The VID is a 12-bit identifier between 1 and 4094 that provides a unique identity for each VLAN. In Figure 5–1, the Red VLAN has the VID 789, the Yellow VLAN has the VID 456, and the Blue VLAN has the VID 123.

When you configure switches to support VLANs, you need to assign a VID to each port. The VID on the port must be the same as the VID assigned to the interface that connects to the port, as shown in the following figure.

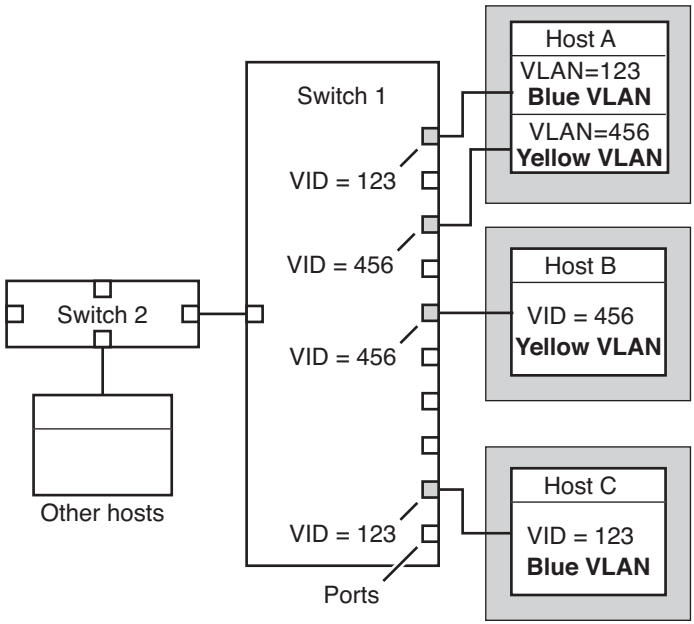

FIGURE 5–2 Switch Configuration for a Network with VLANs

Figure 5–2 shows multiple hosts that are connected to different VLANs. Two hosts belong to the same VLAN. In this figure, the primary network interfaces of the three hosts connect to Switch 1. Host A is a member of the Blue VLAN. Therefore, Host A's interface is configured with the VID 123. This interface connects to Port 1 on Switch 1, which is then configured with the VID 123. Host B is a member of the Yellow VLAN with the VID 456. Host B's interface connects to Port 5 on Switch 1, which is configured with the VID 456. Finally, Host C's interface connects to Port 9 on Switch 1. The Blue VLAN is configured with the VID 123.

The figure also shows that a single host can also belong to more than one VLAN. For example, Host A has two interfaces. The second interface is configured with the VID 456 and is connected to Port 3 which is also configured with the VID 456. Thus, Host A is a member of both the Blue VLAN and the Yellow VLAN.

Meaningful Names for VLANs

In this Solaris release, you can assign meaningful names to VLAN interfaces. VLAN names consist of a link name and the VLAN ID number (VID), such as `sales0` You should assign customized names when you create VLANs. For more information about customized names, see "Assigning Names to Data Links" on page 15. For more information about valid customized names, see "Rules for Valid Link Names" on page 20.

Planning for VLANs on a Network

Use the following procedure to plan for VLANs on your network.

▼ How to Plan a VLAN Configuration

1. **Examine the local network topology and determine where subdivision into VLANs is appropriate.**

   For a basic example of such a topology, refer to Figure 5–1.

2. **Create a numbering scheme for the VIDs, and assign a VID to each VLAN.**

   Note – A VLAN numbering scheme might already exist on the network. If so, you must create VIDs within the existing VLAN numbering scheme.

3. **On each system, determine which interfaces will be members of a particular VLAN.**

   a. **Determine which interfaces are configured on a system.**

      ```
      # dladm show-link
      ```

   b. **Identify which VID will be associated with each data link on the system.**

   c. **Create the VLAN by using the `dladm create-vlan` command.**

4. **Check the connections of the interfaces to the network's switches.**

   Note the VID of each interface and the switch port where each interface is connected.

5. **Configure each port of the switch with the same VID as the interface to which it is connected.**

   Refer to the switch manufacturer's documentation for configuration instructions.

Configuring VLANs

The following procedure shows how to create and configure a VLAN. In this Solaris release, all Ethernet devices can support VLANs. However, some restrictions exist with certain devices. For these exceptions, refer to "VLANs on Legacy Devices" on page 78.

▼ How to Configure a VLAN

Before You Begin Data links must already be configured on your system before you can create VLANs. See How to Configure an IP Interface After System Installation.

1 On the system in which you configure VLANs, assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 Determine the types of links that are in use in your system.

```
# dladm show-link
```

3 Create a VLAN link over a data-link.

```
# dladm create-vlan -l link -v VID vlan-link
```

link	Specifies the link on which the VLAN interface is being created.
VID	Indicates the VLAN ID number
vlan-link	Specifies the name of the VLAN, which can also be an administratively-chosen name.

4 Verify the VLAN configuration.

```
# dladm show-vlan
```

5 Configure an IP interface over the VLAN.

```
# ifconfig interface plumb IP-address up
```

where *interface* takes the same name as the VLAN name.

Note – You can assign IPv4 or IPv6 addresses to the VLAN's IP interface.

---

6 (Optional) To make the IP configuration for the VLAN persist across reboots, create an `/etc/hostname.`*interface* file to contain the interface's IP address.

The *interface* takes the name that you assign to the VLAN.

Example 5–1 Configuring a VLAN

This example configures the VLAN `sales` over the link subitops0. The VLAN is configured to persist across reboots.

```
# dladm show-link
LINK        CLASS     MTU     STATE     OVER
subitops0   phys      1500    up        --
ce1         phys      1500    up        --

# dladm create-vlan -l subitops0 -v 7 sales
# dladm show-vlan
LINK       VID     OVER        FLAGS
sales      7       subitops0   ----
```

When link information is displayed, the VLAN link is included in the list.

```
# dladm show-link
LINK          CLASS    MTU      STATE     OVER
subitops0     phys     1500     up        --
ce1           phys     1500     up        --
sales         vlan     1500     up        subitops0

# ifconfig sales plumb 10.0.0.3/24 up
# echo 10.0.0.3/24 > /etc/hostname.sales
```

VLANs on Legacy Devices

Certain legacy devices handle only packets whose maximum frame size is 1514 bytes. Packets whose frame sizes exceed the maximum limit are dropped. For such cases, follow the same procedure listed in "How to Configure a VLAN" on page 77. However, when creating the VLAN, use the `-f` option to force the creation of the VLAN.

The general steps to perform are as follows:

1. Create the VLAN with the `-f` option.

   ```
   # dladm create-vlan -f -l link -v VID [vlan-link]
   ```

2. Set a lower size for the maximum transmission unit (`MTU`), such as 1496 bytes.

   ```
   # dladm set-linkprop -p default_mtu=1496 vlan-link
   ```

   The lower `MTU` value allows space for the link layer to insert the VLAN header prior to transmission.

3. Perform the same step to set the same lower value for the `MTU` size of each node in the VLAN.

For more information about changing link property values, refer to "Administering NIC Driver Properties" on page 27.

Performing Other Administrative Tasks on VLANs

This section describes the usage of new `dladm` subcommands for other VLAN tasks. These `dladm` commands also work with link names.

▼ How to Display VLAN Information

1 Assume the System Administrator role or become superuser.

The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2 Display VLAN information.

```
# dladm show-vlan [vlan-link]
```

If you do not specify a VLAN link, the command displays information about all configured VLANs.

Example 5–2 Displaying VLAN Information

The following example shows the available VLANs in a system.

```
# dladm show-vlan
LINK          VID     OVER        FLAGS
sales         7       subitops0   ----
managers      5       net0        ----
```

Configured VLANs also appear when you issue the `dladm show-link` command. In the command output, the VLANs are appropriately identified in the `CLASS` column.

```
# dladm show-link
LINK           CLASS     MTU     STATE     OVER
subitops0      phys      1500    up        --
sales          vlan      1500    up        subitops0
net0           phys      1500    up        --
managers       vlan      1500    up        net0
```

How to Remove a VLAN

1. **Assume the System Administrator role or become superuser.**

   The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2. **Determine which VLAN you want to remove.**

   ```
   # dladm show-vlan
   ```

3. **Unplumb the VLAN's IP interface.**

   ```
   # ifconfig vlan-interface unplumb
   ```

   where *vlan-interface* is the IP interface that is configured over the VLAN.

   Note – You cannot remove a VLAN that is currently in use.

4. **Remove the VLAN by performing one of the following steps:**

   - To delete the VLAN temporarily, use the `-t` option as follows:

     ```
     # dladm delete-vlan -t vlan
     ```

   - To make the deletion persist, perform the following:

     a. Remove the VLAN.

        ```
        # dladm delete-vlan vlan
        ```

     b. Remove the `/etc/hostname.`*vlan-interface* file.

Example 5–3 Removing a VLAN

```
# dladm show-vlan
LINK        VID     OVER          FLAGS
sales       5       subitops0     ----
managers    7       net0          ----

# ifconfig managers unplumb
# dladm delete-vlan managers
# rm /etc/hostname.managers
```

◆ ◆ ◆ CHAPTER 6

Administering Link Aggregations

This chapter describes procedures to configure and maintain link aggregations. The procedures include steps that avail of new features such as support for flexible link names.

Overview of Link Aggregations

The Solaris OS supports the organization of network interfaces into link aggregations. A *link aggregation* consists of several interfaces on a system that are configured together as a single, logical unit. Link aggregation, also referred to as *trunking*, is defined in the IEEE 802.3ad Link Aggregation Standard (`http://www.ieee802.org/3/index.html`).

The IEEE 802.3ad Link Aggregation Standard provides a method to combine the capacity of multiple full-duplex Ethernet links into a single logical link. This link aggregation group is then treated as though it were, in fact, a single link.

The following are features of link aggregations:

- **Increased bandwidth** – The capacity of multiple links is combined into one logical link.
- **Automatic failover/failback** – Traffic from a failed link is failed over to working links in the aggregation.
- **Load balancing** – Both inbound and outbound traffic is distributed according to user selected load-balancing policies, such as source and destination MAC or IP addresses.
- **Support for redundancy** – Two systems can be configured with parallel aggregations.
- **Improved administration** – All interfaces are administered as a single unit.
- **Less drain on the network address pool** – The entire aggregation can be assigned one IP address.

Link Aggregation Basics

The basic link aggregation topology involves a single aggregation that contains a set of physical interfaces. You might use the basic link aggregation in the following situations:

- For systems that run an application with distributed heavy traffic, you can dedicate an aggregation to that application's traffic.
- For sites with limited IP address space that nevertheless require large amounts of bandwidth, you need only one IP address for a large aggregation of interfaces.
- For sites that need to hide the existence of internal interfaces, the IP address of the aggregation hides its interfaces from external applications.

Figure 6–1 shows an aggregation for a server that hosts a popular web site. The site requires increased bandwidth for query traffic between Internet customers and the site's database server. For security purposes, the existence of the individual interfaces on the server must be hidden from external applications. The solution is the aggregation `aggr1` with the IP address `192.168.50.32`. This aggregation consists of three interfaces,`bge0` through `bge2`. These interfaces are dedicated to sending out traffic in response to customer queries. The outgoing address on packet traffic from all the interfaces is the IP address of `aggr1`, `192.168.50.32`.

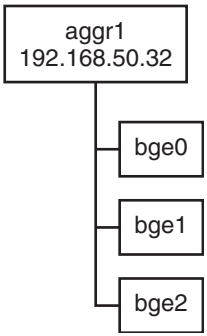

FIGURE 6–1 Basic Link Aggregation Topology

Figure 6–2 depicts a local network with two systems, and each system has an aggregation configured. The two systems are connected by a switch. If you need to run an aggregation through a switch, that switch must support aggregation technology. This type of configuration is particularly useful for high availability and redundant systems.

In the figure, System A has an aggregation that consists of two interfaces, `bge0` and `bge1`. These interfaces are connected to the switch through aggregated ports. System B has an aggregation of four interfaces, `e1000g0` through `e1000g3`. These interfaces are also connected to aggregated ports on the switch.

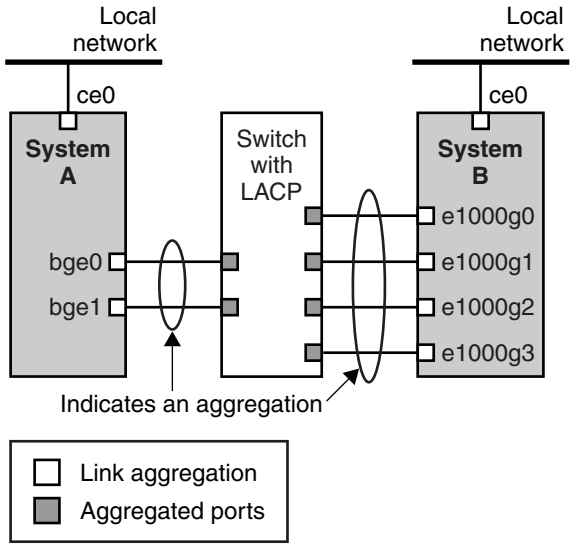

FIGURE 6–2 Link Aggregation Topology With a Switch

Back-to-Back Link Aggregations

The back-to-back link aggregation topology involves two separate systems that are cabled directly to each other, as shown in the following figure. The systems run parallel aggregations.

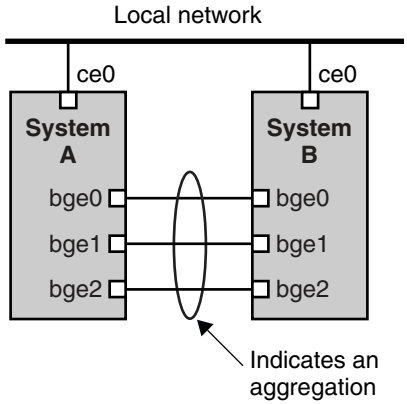

FIGURE 6–3 Basic Back-to-Back Aggregation Topology

In this figure, device `bge0` on System A is directly linked to `bge0` on System B, and so on. In this way, Systems A and B can support redundancy and high availability, as well as high-speed communications between both systems. Each system also has interface `ce0` configured for traffic flow within the local network.

The most common application for back-to-back link aggregations is mirrored database servers. Both servers need to be updated together and therefore require significant bandwidth, high-speed traffic flow, and reliability. The most common use of back-to-back link aggregations is in data centers.

Policies and Load Balancing

If you plan to use a link aggregation, consider defining a policy for outgoing traffic. This policy can specify how you want packets to be distributed across the available links of an aggregation, thus establishing load balancing. The following are the possible layer specifiers and their significance for the aggregation policy:

- **L2** – Determines the outgoing link by hashing the MAC (L2) header of each packet
- **L3** – Determines the outgoing link by hashing the IP (L3) header of each packet
- **L4** – Determines the outgoing link by hashing the TCP, UDP, or other ULP (L4) header of each packet

Any combination of these policies is also valid. The default policy is L4. For more information, refer to the `dladm`(1M) man page.

Aggregation Mode and Switches

If your aggregation topology involves connection through a switch, you must note whether the switch supports the *link aggregation control protocol (LACP)*. If the switch supports LACP, you must configure LACP for the switch and the aggregation. However, you can define one of the following *modes* in which LACP is to operate:

- **Off mode** – The default mode for aggregations. LACP packets, which are called *LACPDUs* are not generated.
- **Active mode** – The system generates LACPDUs at regular intervals, which you can specify.
- **Passive mode** – The system generates an LACPDU only when it receives an LACPDU from the switch. When both the aggregation and the switch are configured in passive mode, they cannot exchange LACPDUs.

See the `dladm`(1M) man page and the switch manufacturer's documentation for syntax information.

Requirements for Link Aggregations

Your link aggregation configuration is bound by the following requirements:

- You must use the `dladm` command to configure aggregations.
- An interface that has been plumbed cannot become a member of an aggregation.
- All interfaces in the aggregation must run at the same speed and in full-duplex mode.
- You must set the value for MAC addresses to "true" in the EEPROM parameter `local-mac-address?` For instructions, refer to How to Ensure That the MAC Address of an Interface Is Unique.

Certain devices do not fulfill the requirement of the IEEE 802.3ad Link Aggregation Standard to support link state notification. This support must exist in order for a port to attach to an aggregation or to detach from an aggregation. Devices that do not support link state notification can be aggregated only by using the `-f` option of the `dladm create-aggr` command. For such devices, the link state is always reported as `UP`. For information about the use of the `-f` option, see "How to Create a Link Aggregation" on page 85.

Flexible Names for Link Aggregations

Flexible names can be assigned to link aggregations. Any meaningful name can be assigned to a link aggregation. For more information about flexible or customized names, see "Assigning Names to Data Links" on page 15. Previous Solaris releases identify a link aggregation by the value of a *key* that you assign to the aggregation. For an explanation of this method, see "Overview of Link Aggregations" on page 81. Although that method continues to be valid, preferably, you should use customized names to identify link aggregations.

Similar to all other data-link configurations, link aggregations are administered with the `dladm` command.

▼ How to Create a Link Aggregation

Before You Begin

Note – Link aggregation only works on full-duplex, point-to-point links that operate at identical speeds. Make sure that the interfaces in your aggregation conform to this requirement.

If you are using a switch in your aggregation topology, make sure that you have done the following on the switch:

- Configured the ports to be used as an aggregation
- If the switch supports LACP, configured LACP in either active mode or passive mode

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Display the network data-link information.**

   ```
   # dladm show-link
   ```

3. **Make sure that the link over which you are creating the aggregation is not opened by any application.**

   For example, if the IP interface over the link is plumbed, then unplumb the interface.

   ```
   # ifconfig interface unplumb
   ```

   where *interface* refers to the IP interface that is plumbed and using the link.

4. **Create a link aggregation.**

   ```
   # dladm create-aggr [-f] -l link1 -l link2 [...] aggr
   ```

   `-f` Forces the creation of the aggregation. Use this option when you are attempting to aggregate devices that do not support link state notification.

   linkn Specifies the data links that you want to aggregate.

   aggr Specifies the name that you want to assign to the aggregation.

5. **Plumb and configure an IP interface over the newly created aggregation.**

   ```
   # ifconfig interface plumb IP-address up
   ```

   where *interface* takes the name of the aggregation.

6. **Check the status of the aggregation you just created.**

   The aggregation's state should be `UP`.

   ```
   # dladm show-aggr
   ```

7. **(Optional) Make the IP configuration of the link aggregation persist across reboots.**

   a. **Create the `/etc/hostname` file for the aggregation's interface.**

      If the aggregation contains IPv4 addresses, the corresponding hostname file is `/etc/hostname.`*aggr*. For IPv6–based link aggregations, the corresponding hostname file is `/etc/hostname6.`*aggr*.

   b. **Type the IPv4 or IPv6 address of the link aggregation into the file.**

c. Perform a reconfiguration boot.

```
# reboot -- -r
```

Example 6–1 Creating a Link Aggregation

This example shows the commands that are used to create a link aggregation with two data links, subvideo0 and subvideo1. The configuration is persistent across system reboots.

```
# dladm show-link
LINK          CLASS     MTU     STATE     OVER
subvideo0     phys      1500    up        ----
subvideo1     phys      1500    up        ----

# dladm create-aggr -l subvideo0 -l subvideo1 video0
# ifconfig video0 plumb 10.8.57.50/24 up
# dladm show-aggr
LINK      POLICY  ADDRPOLICY      LACPACTIVITY   LACPTIMER  FLAGS
video0    L4      auto            off            short      -----

# echo 10.8.57.50/24 > /etc/hostname.video0

# reboot -- -r
```

When you display link information, the link aggregation is included in the list.

```
# dladm show-link
LINK          CLASS     MTU     STATE     OVER
subvideo0     phys      1500    up        ----
subvideo1     phys      1500    up        ----
video0        aggr      1500    up        subvideo0, subvideo1
```

▼ How to Modify an Aggregation

This procedure shows how to make the following changes to an aggregation definition:

- Modifying the policy for the aggregation
- Changing the mode for the aggregation

1 Assume the System Administrator role.

The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2 Modify the policy of the aggregation.

```
# dladm modify-aggr -P policy-key aggr
```

policy-key Represents one or more of the policies L2, L3, and L4, as explained in "Policies and Load Balancing" on page 84.

aggr Specifies the aggregation whose policy you want to modify.

3 Modify the LACP mode of the aggregation.

```
# dladm modify-aggr -L LACP-mode -T timer-value aggr
```

`-L` *LACP-mode* Indicates the LACP mode in which the aggregation is to run. The values are `active`, `passive`, and `off`. If the switch runs LACP in passive mode, be sure to configure active mode for your aggregation.

`-T` *timer-value* Indicates the LACP timer value, either `short` or `long`.

Example 6–2 Modifying a Link Aggregation

This example shows how to modify the policy of aggregation `video0` to L2 and then turn on active LACP mode.

```
# dladm modify-aggr -P L2 video0
# dladm modify-aggr -L active -T short video0
# dladm show-aggr
LINK      POLICY  ADDRPOLICY      LACPACTIVITY   LACPTIMER  FLAGS
video0    L2      auto            active         short      -----
```

▼ How to Add a Link to an Aggregation

1 Assume the System Administrator role or become superuser.

The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2 Ensure that the link you want to add has no IP interface that is plumbed over the link.

```
# ifconfig interface unplumb
```

3 Add the link to the aggregation.

```
# dladm add-aggr -l link [-l link] [...] aggr
```

where *link* represents a data link that you are adding to the aggregation.

4. **Perform other tasks to modify the entire link aggregation configuration after more data links are added.**

   For example, in the case of a configuration that is illustrated in Figure 6–3, you might need to add or modify cable connections and reconfigure switches to accommodate the additional data links. Refer to the switch documentation to perform any reconfiguration tasks on the switch.

Example 6–3 Adding a Link to an Aggregation

This example shows how to add a link to the aggregation `video0`.

```
# dladm show-link
LINK          CLASS     MTU     STATE     OVER
subvideo0     phys      1500    up        ----
subvideo1     phys      1500    up        ----
video0        aggr      1500    up        subvideo0, subvideo1
net3          phys      1500    unknown   ----

# dladm add-aggr -l net3 video0
# dladm show-link
LINK          CLASS     MTU     STATE     OVER
subvideo0     phys      1500    up        ----
subvideo1     phys      1500    up        ----
video0        aggr      1500    up        subvideo0, subvideo1, net3
net3          phys      1500    up        ----
```

▼ How to Remove a Link From an Aggregation

1. **Assume the System Administrator role.**

   The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2. **Remove a link from the aggregation.**

   `# dladm remove-aggr -l` *link aggr-link*

Example 6–4 Removing a Link From an Aggregation

This example shows how to remove a link from the aggregation `video0`.

```
dladm show-link
LINK          CLASS     MTU     STATE     OVER
subvideo0     phys      1500    up        ----
```

```
subvideo1     phys      1500    up        ----
video0        aggr      1500    up        subvideo0, subvideo1, net3
net3          phys      1500    up        ----

# dladm remove-aggr -l net3 video0
# dladm show-link
LINK          CLASS     MTU     STATE     OVER
subvideo0     phys      1500    up        ----
subvideo1     phys      1500    up        ----
video0        aggr      1500    up        subvideo0, subvideo1
net3          phys      1500    unknown   ----
```

How to Delete an Aggregation

1. **Assume the System Administrator role.**

   The System Administrator role includes the Network Management profile. To create the role and assign the role to a user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*.

2. **Unplumb the aggregation.**

   ```
   # ifconfig aggr unplumb
   ```

3. **Delete the aggregation.**

   ```
   # dladm delete-aggr aggr
   ```

4. **To make the deletion persistent, remove the IP configuration for the link aggregation in /etc/hostname.*interface* file.**

   ```
   # rm /etc/hostname.interface
   ```

Example 6–5 Deleting an Aggregation

This example deletes the aggregation `video0`. The deletion is persistent.

```
# ifconfig video0 unplumb
# dladm delete-aggr video0
# rm /etc/hostname.video0
```

▼ How to Configure VLANs Over a Link Aggregation

In the same manner as configuring VLANs over an interface, you can also create VLANs on a link aggregation. VLANs are described in Chapter 5, "Administering VLANs." This section combines configuring VLANs and link aggregations.

Before You Begin Create the link aggregation first and configure it with a valid IP address. To create link aggregations, refer to "How to Create a Link Aggregation" on page 85.

1. **List the aggregations that are configured in the system.**

   ```
   # dladm show-link
   ```

2. **Create a VLAN over the link aggregation.**

   ```
   # dladm create-vlan -l link -v VID vlan-link
   ```

   where

   link Specifies the link on which the VLAN interface is being created. In this specific case, the link refers to the link aggregation.

   VID Indicates the VLAN ID number

   vlan-link Specifies the name of the VLAN, which can also be an administratively-chosen name.

3. **Repeat Step 2 to create other VLANs over the aggregation.**

4. **Configure IP interfaces over the VLANs with valid IP addresses.**

5. **To create persistent VLAN configurations, add the IP address information to the corresponding `/etc/hostname.`*interface* configuration files.**

   The *interface* takes the name of the VLAN that you assigned.

Example 6–6 Configuring Multiple VLANs Over a Link Aggregation

In this example, two VLANs are configured on a link aggregation. The VLANs are assigned VIDs 193 and 194, respectively.

```
# dladm show-link
LINK            CLASS     MTU     STATE     OVER
subvideo0       phys      1500    up        ----
subvideo1       phys      1500    up        ----
video0          aggr      1500    up        subvideo0, subvideo1

# dladm create-vlan -l video0 -v 193 salesregion1
```

```
# dladm create-vlan -l video0 -v 194 salesregion2

# ifconfig salesregion1 192.168.10.5/24 plumb up
# ifconfig salesregion2 192.168.10.25/24 plumb up

# vi /etc/hostname.salesregion1
192.168.10.5/24

# vi /etc/hostname.salesregion2
192.168.10.25/24
```

Combining Network Configuration Tasks While Using Customized Names

This section provides an example that combines all the procedures in the previous chapters about configuring links, VLANs, and link aggregations while using customized names. For a description of other networking scenarios that use customized names, see the article in http://www.sun.com/bigadmin/sundocs/articles/vnamingsol.jsp.

EXAMPLE 6–7 Configuring Links, Aggregations, and VLANs

In this example, a system that consists of 4 NICs needs to be configured to be a router for 8 separate subnets. To attain this objective, 8 links will be configured, one for each subnet. First, a link aggregation is created on all 4 NICs. This untagged link becomes the default untagged subnet for the network to which the default route points.

Then VLAN interfaces are configured over the link aggregation for the other subnets. The subnets are named by basing on a color-coded scheme. Accordingly, the VLAN names are likewise named to correspond to their respective subnets. The final configuration consists of 8 links for the eight subnets: 1 untagged link, and 7 tagged VLAN links.

To make the configurations persist across reboots, the same procedures apply as in previous Solaris releases. For example, IP addresses need to be added to configuration files like `/etc/inet/ndpd.conf` or `/etc/hostname.`*interface*. Or, filter rules for the interfaces need to be included in a rules file. These final steps are not included in the example. For these steps, refer to the appropriate chapters in *System Administration Guide: IP Services*, particularly *TCP/IP Administration* and *DHCP*.

```
# dladm show-link
LINK        CLASS      MTU  STATE    OVER
nge0        phys      1500  up       --
nge1        phys      1500  up       --
e1000g0     phys      1500  up       --
e1000g1     phys      1500  up       --
```

EXAMPLE 6–7 Configuring Links, Aggregations, and VLANs *(Continued)*

```
# dladm show-phys
LINK        MEDIA               STATE      SPEED  DUPLEX   DEVICE
nge0        Ethernet            up        1000Mb  full     nge0
nge1        Ethernet            up        1000Mb  full     nge1
e1000g0     Ethernet            up        1000Mb  full     e1000g0
e1000g1     Ethernet            up        1000Mb  full     e1000g1

# ifconfig nge0 unplumb
# ifconfig nge1 unplumb
# ifconfig e1000g0 unplumb
# ifconfig e1000g1 unplumb

# dladm rename-link nge0 net0
# dladm rename-link nge1 net1
# dladm rename-link e1000g0 net2
# dladm rename-link e1000g1 net3

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

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

# dladm create-aggr -P L2,L3 -l net0 -l net1 -l net2 -l net3 default0

# dladm show-link
LINK        CLASS      MTU  STATE    OVER
net0        phys      1500  up       --
net1        phys      1500  up       --
net2        phys      1500  up       --
net3        phys      1500  up       --
default0    aggr      1500  up       net0 net1 net2 net3

# dladm create-vlan -v 2 -l default0 orange0
# dladm create-vlan -v 3 -l default0 green0
# dladm create-vlan -v 4 -l default0 blue0
```

EXAMPLE 6–7 Configuring Links, Aggregations, and VLANs *(Continued)*

```
# dladm create-vlan -v 5 -l default0 white0
# dladm create-vlan -v 6 -l default0 yellow0
# dladm create-vlan -v 7 -l default0 red0
# dladm create-vlan -v 8 -l default0 cyan0

# dladm show-link
LINK        CLASS      MTU  STATE    OVER
net0        phys      1500  up       --
net1        phys      1500  up       --
net2        phys      1500  up       --
net3        phys      1500  up       --
default0    aggr      1500  up       net0 net1 net2 net3
orange0     vlan      1500  up       default0
green0      vlan      1500  up       default0
blue0       vlan      1500  up       default0
white0      vlan      1500  up       default0
yellow0     vlan      1500  up       default0
red0        vlan      1500  up       default0
cyan0       vlan      1500  up       default0

# dladm show-vlan
LINK          VID   OVER        FLAGS
orange0         2   default0    -----
green0          3   default0    -----
blue0           4   default0    -----
white0          5   default0    -----
yellow0         6   default0    -----
red0            7   default0    -----
cyan0           8   default0    -----

# ifconfig orange0 plumb ...
# ifconfig green0 plumb ...
# ifconfig blue0 plumb ...
# ifconfig white0 plumb ...
# ifconfig yellow0 plumb ...
# ifconfig red0 plumb ...
# ifconfig cyan0 plumb ...
```

◆ ◆ ◆ CHAPTER 7

Introducing IPMP

IP network multipathing (IPMP) provides physical interface failure detection, transparent network access failover, and packet load spreading for systems with multiple interfaces that are connected to a particular local area network or LAN.

This chapter contains the following information:

Note – Throughout the description of IPMP in this chapter and in Chapter 8, "Administering IPMP," all references to the term *interface* specifically mean *IP interface*. Unless a qualification explicitly indicates a different use of the term, such as a network interface card (NIC), the term always refers to the interface that is configured on the IP layer.

What's New With IPMP

The following features differentiate the current IPMP implementation from the previous implementation:

- An IPMP group is represented as an IPMP IP interface. This interface is treated just like any other interface on the IP layer of the networking stack. All IP administrative tasks, routing tables, Address Resolution Protocol (ARP) tables, firewall rules, and other IP-related procedures work with an IPMP group by referring to the IPMP interface.

- The system becomes responsible for the distribution of data addresses among underlying active interfaces. In the previous IPMP implementation, the administrator initially determines the binding of data addresses to corresponding interfaces when the IPMP group is created. In the current implementation, when the IPMP group is created, data addresses belong to the IPMP interface as an address pool. The kernel then automatically and randomly binds the data addresses to the underlying active interfaces of the group.
- The `ipmpstat` tool is introduced as the principal tool to obtain information about IPMP groups. This command provides information about all aspects of the IPMP configuration, such as the underlying IP interfaces of the group, test and data addresses, types of failure detection being used, and which interfaces have failed. The `ipmpstat` functions, the options you can use, and the output each option generates are all described in "Monitoring IPMP Information" on page 143.
- The IPMP interface can be assigned a customized name to identify the IPMP group more easily within your network setup. For the procedures to configure IPMP groups with customized names, see any procedure that describes the creation of an IPMP group in "Configuring IPMP Groups" on page 124.

Deploying IPMP

This section describes various topics about the use of IPMP groups.

Why You Should Use IPMP

Different factors can cause an interface to become unusable. Commonly, an IP interface can fail. Or, an interface might be switched offline for hardware maintenance. In such cases, without an IPMP group, the system can no longer be contacted by using any of the IP addresses that are associated with that unusable interface. Additionally, existing connections that use those IP addresses are disrupted.

With IPMP, one or more IP interfaces can be configured into an *IPMP group*. The group functions like an IP interface with data addresses to send or receive network traffic. If an underlying interface in the group fails, the data addresses are redistributed among the remaining underlying active interfaces in the group. Thus, the group maintains network connectivity despite an interface failure. With IPMP, network connectivity is always available, provided that a minimum of one interface is usable for the group.

Additionally, IPMP improves overall network performance by automatically spreading out outbound network traffic across the set of interfaces in the IPMP group. This process is called outbound *load spreading*. The system also indirectly controls inbound load spreading by performing source address selection for packets whose IP source address was not specified by the application. However, if an application has explicitly chosen an IP source address, then the system does not vary that source address.

When You Must Use IPMP

The configuration of an IPMP group is determined by your system configurations. Observe the following rules:

- Multiple IP interfaces on the same local area network or LAN must be configured into an IPMP group. LAN broadly refers to a variety of local network configurations including VLANs and both wired and wireless local networks whose nodes belong to *the same link-layer broadcast domain*.
- Underlying IP interfaces of an IPMP group must not span different LANs.

For example, suppose that a system with three interfaces is connected to two separate LANs. Two IP interfaces link to one LAN while a single IP interface connects to the other. In this case, the two IP interfaces connecting to the first LAN must be configured as an IPMP group, as required by the first rule. In compliance with the second rule, the single IP interface that connects to the second LAN cannot become a member of that IPMP group. No IPMP configuration is required of the single IP interface. However, you can configure the single interface into an IPMP group to monitor the interface's availability. The single-interface IPMP configuration is discussed further in "Types of IPMP Interface Configurations" on page 106.

Consider another case where the link to the first LAN consists of three IP interfaces while the other link consists of two interfaces. This setup requires the configuration of two IPMP groups: a three-interface group that links to the first LAN, and a two-interface group to connect to the second.

Comparing IPMP and Link Aggregation

IPMP and link aggregation are different technologies to achieve improved network performance as well as maintain network availability. In general, you deploy link aggregation to obtain better network performance, while you use IPMP to ensure high availability.

The following table presents a general comparison between link aggregation and IPMP.

	IPMP	Link Aggregation
Network technology type	Layer 3 (IP layer)	Layer 2 (link layer)
Configuration tool	`ifconfig`	`dladm`
Link-based failure detection	Supported.	Supported.

	IPMP	Link Aggregation
Probe-based failure detection	ICMP-based, targeting any defined system in the same IP subnet as test addresses, across multiple levels of intervening layer-2 switches.	Based on Link Aggregation Control Protocol (LACP), targeting immediate peer host or switch.
Use of standby interfaces	Supported	Not supported
Span multiple switches	Supported	Generally not supported; some vendors provide proprietary and non-interoperable solutions to span multiple switches.
Hardware support	Not required	Required. For example, a link aggregation in the system that is running the Solaris OS requires that corresponding ports on the switches be also aggregated.
Link layer requirements	Broadcast-capable	Ethernet-specific
Driver framework requirements	None	Must use GLDv3 framework
Load spreading support	Present, controlled by kernel. Inbound load spreading is indirectly affected by source address selection.	Finer grain control of the administrator over load spreading of outbound traffic by using `dladm` command. Inbound load spreading supported.

In link aggregations, incoming traffic is spread over the multiple links that comprise the aggregation. Thus, networking performance is enhanced as more NICs are installed to add links to the aggregation. IPMP's traffic uses the IPMP interface's data addresses as they are bound to the available active interfaces. Thus, for example, if all the data traffic is flowing between only two IP addresses but not necessarily over the same connection, then adding more NICs will not improve performance with IPMP because only two IP addresses remain usable.

The two technologies complement each other and can be deployed together to provide the combined benefits of network performance and availability. For example, except where proprietary solutions are provided by certain vendors, link aggregations currently cannot span multiple switches. Thus, a switch becomes a single point of failure for a link aggregation between the switch and a host. If the switch fails, the link aggregation is likewise lost, and network performance declines. IPMP groups do not face this switch limitation. Thus, in the scenario of a LAN using multiple switches, link aggregations that connect to their respective switches can be combined into an IPMP group on the host. With this configuration, both enhanced network performance as well as high availability are obtained. If a switch fails, the data addresses of the link aggregation to that failed switch are redistributed among the remaining link aggregations in the group.

For other information about link aggregations, see Chapter 6, "Administering Link Aggregations."

Using Flexible Link Names on IPMP Configuration

With support for customized link names, link configuration is no longer bound to the physical NIC to which the link is associated. Using customized link names allows you to have greater flexibility in administering IP interfaces. This flexibility extends to IPMP administration as well. In certain cases of failure of an underlying interface of an IPMP group, the resolution would require the replacement of the physical hardware or NIC. The replacement NIC, provided it is the same type as the failed NIC, can be renamed to inherit the configuration of the failed NIC. You do not have to create new configurations for the new NIC before you can add it to the IPMP group. After you rename the new NIC's link with the link name of the replaced NIC, the new NIC automatically becomes a member of the IPMP group when you bring that NIC online. The multipathing daemon then deploys the interface according to the IPMP configuration of active and standby interfaces.

Therefore, to optimize your networking configuration and facilitate IPMP administration, you must employ flexible link names for your interfaces by assigning them generic names. In the following section "How IPMP Works" on page 99, all the examples use flexible link names for the IPMP group and its underlying interfaces. For details about the processes behind NIC replacements in a networking environment that uses customized link names, refer to "IPMP and Dynamic Reconfiguration" on page 112. For an overview of the networking stack and the use of customized link names, refer to "Overview of the Networking Stack" on page 13.

How IPMP Works

IPMP maintains network availability by attempting to preserve the original number of active and standby interfaces when the group was created.

IPMP failure detection can be link-based or probe-based or both to determine the availability of a specific underlying IP interface in the group. If IPMP determines that an underlying interface has failed, then that interface is flagged as failed and is no longer usable. The data IP address that was associated with the failed interface is then redistributed to another functioning interface in the group. If available, a standby interface is also deployed to maintain the original number of active interfaces.

Consider a three-interface IPMP group `itops0` with an active-standby configuration, as illustrated in Figure 7–1.

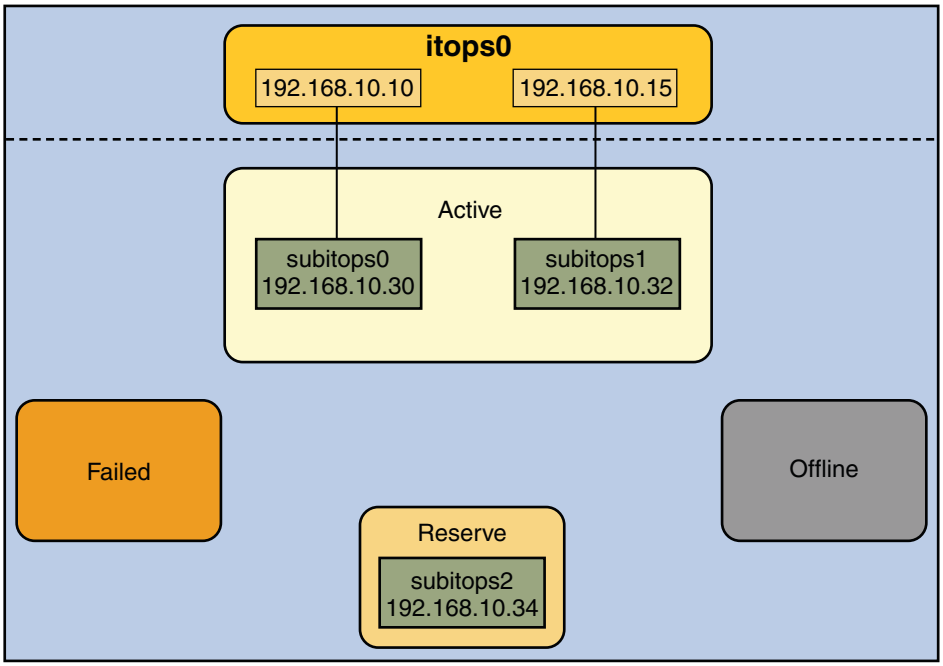

FIGURE 7–1 IPMP Active–Standby Configuration

The group `itops0` is configured as follows:

- Two data addresses are assigned to the group: `192.168.10.10` and `192.168.10.15`.
- Two underlying interfaces are configured as active interfaces and are assigned flexible link names: `subitops0` and `subitops1`.
- The group has one standby interface, also with a flexible link name: `subitops2`.
- Probe–based failure detection is used, and thus the active and standby interfaces are configured with test addresses, as follows:
  - `subitops0`: `192.168.10.30`
  - `subitops1`: `192.168.10.32`
  - `subitops2`: `192.168.10.34`

Note – The `Active`, `Offline`, `Reserve`, and `Failed` areas in the figures indicate only the status of underlying interfaces, and not physical locations. No physical movement of interfaces or addresses nor transfer of IP interfaces occur within this IPMP implementation. The areas only serve to show how an underlying interface changes status as a result of either failure or repair.

You can use the `ipmpstat` command with different options to display specific types of information about existing IPMP groups. For additional examples, see "Monitoring IPMP Information" on page 143.

The IPMP configuration in Figure 7–1 can be displayed by using the following `ipmpstat` command:

```
# ipmpstat -g
GROUP     GROUPNAME     STATE     FDT       INTERFACES
itops0    itops0        ok        10.00s    subitops1 subitops0 (subitops2)
```

To display information about the group's underlying interfaces, you would type the following:

```
# ipmpstat -i
INTERFACE       ACTIVE     GROUP     FLAGS      LINK       PROBE     STATE
subitops0       yes        itops0    -------    up         ok        ok
subitops1       yes        itops0    --mb---    up         ok        ok
subitops2       no         itops0    is-----    up         ok        ok
```

IPMP maintains network availability by managing the underlying interfaces to preserve the original number of active interfaces. Thus, if `subitops0` fails, then `subitops2` is deployed to ensure that the group continues to have two active interfaces. The activation of the `subitops2` is shown in Figure 7–2.

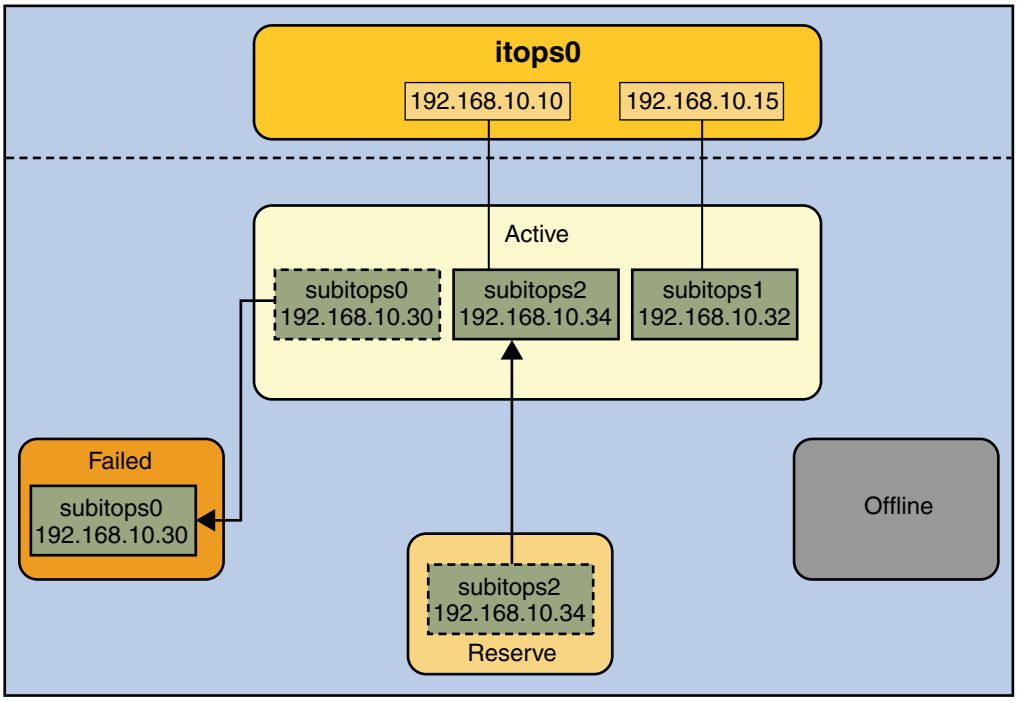

FIGURE 7–2 Interface Failure in IPMP

Note – The one–to–one mapping of data addresses to active interfaces in Figure 7–2 serves only to simplify the illustration. The IP kernel module can assign data addresses randomly without necessarily adhering to a one–to–one relationship between data addresses and interfaces.

The `ipmpstat` utility displays the information in Figure 7–2 as follows:

```
# ipmpstat -i
INTERFACE       ACTIVE     GROUP     FLAGS      LINK       PROBE     STATE
subitops0       no         itops0    -------    up         failed    failed
subitops1       yes        itops0    --mb---    up         ok        ok
subitops2       yes        itops0    -s-----    up         ok        ok
```

After `subitops0` is repaired, then it reverts to its status as an active interface. In turn, `subitops2` is returned to its original standby status.

A different failure scenario is shown in Figure 7–3, where the standby interface `subitops2` fails (1), and later, one active interface, `subitops1`, is switched offline by the administrator (2). The result is that the IPMP group is left with a single functioning interface, `subitops0`.

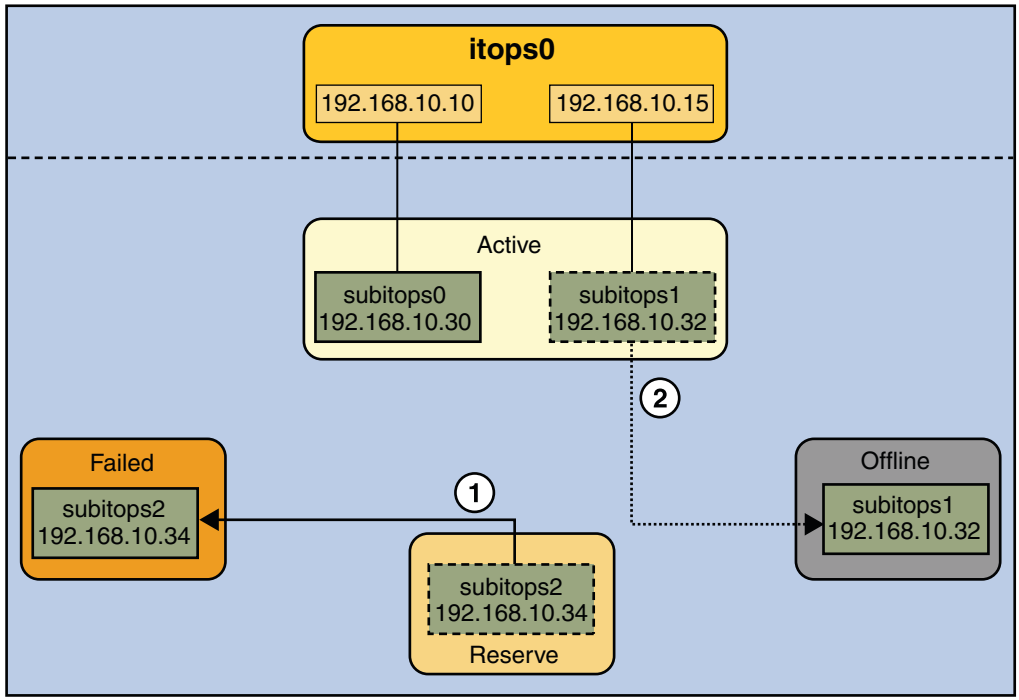

FIGURE 7–3 Standby Interface Failure in IPMP

The `ipmpstat` utility would display the information illustrated by Figure 7–3 as follows:

```
# ipmpstat -i
INTERFACE       ACTIVE     GROUP     FLAGS       LINK       PROBE      STATE
subitops0       yes        itops0    -------     up         ok         ok
subitops1       no         itops0    --mb-d-     up         ok         offline
subitops2       no         itops0    is-----     up         failed     failed
```

For this particular failure, the recovery after an interface is repaired behaves differently. The restoration depends on the IPMP group's original number of active interfaces compared with the configuration after the repair. The recovery process is represented graphically in Figure 7–4.

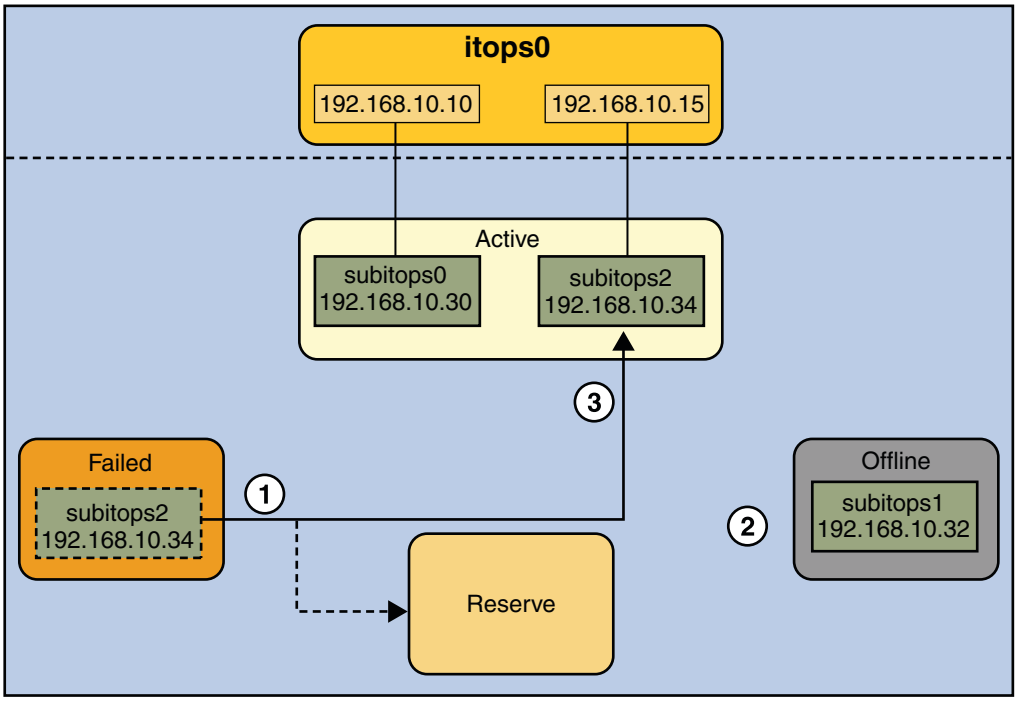

FIGURE 7–4 IPMP Recovery Process

In Figure 7–4, when `subitops2` is repaired, it would normally revert to its original status as a standby interface (1). However, the IPMP group still would not reflect the original number of two active interfaces, because `subitops1` continues to remain offline (2). Thus, IPMP deploys `subitops2` as an active interface instead (3).

The `ipmpstat` utility would display the post-repair IPMP scenario as follows:

```
# ipmpstat -i
INTERFACE       ACTIVE     GROUP     FLAGS       LINK        PROBE     STATE
subitops0       yes        itops0    -------     up          ok        ok
subitops1       no         itops0    --mb-d-     up          ok        offline
subitops2       yes        itops0    -s-----     up          ok        ok
```

A similar restore sequence occurs if the failure involves an active interface that is also configured in `FAILBACK=no` mode, where a failed active interface does not automatically revert to active status upon repair. Suppose `subitops0` in Figure 7–2 is configured in `FAILBACK=no` mode. In that mode, a repaired `subitops0` is switched to a reserve status as a standby interface, even though it was originally an active interface. The interface `subitops2` would remain active to maintain the IPMP group's original number of two active interfaces. The `ipmpstat` utility would display the recovery information as follows:

```
# ipmpstat -i
INTERFACE       ACTIVE     GROUP     FLAGS      LINK       PROBE     STATE
subitops0       no         itops0    i------    up         ok        ok
subitops1       yes        itops0    --mb---    up         ok        ok
subitops2       yes        itops0    -s-----    up         ok        ok
```

For more information about this type of configuration, see "The FAILBACK=no Mode" on page 111.

Solaris IPMP Components

Solaris IPMP involves the following software:

The *multipathing daemon* `in.mpathd` detects interface failures and repairs. The daemon performs both link-based failure detection and probe-based failure detection if test addresses are configured for the underlying interfaces. Depending on the type of failure detection method that is employed, the daemon sets or clears the appropriate flags on the interface to indicate whether the interface failed or has been repaired. As an option, the daemon can also be configured to monitor the availability of all interfaces, including those that are not configured to belong to an IPMP group. For a description of failure detection, see "Failure and Repair Detection in IPMP" on page 108.

The `in.mpathd` daemon also controls the designation of active interfaces in the IPMP group. The daemon attempts to maintain the same number of active interfaces that was originally configured when the IPMP group was created. Thus `in.mpathd` activates or deactivates underlying interfaces as needed to be consistent with the administrator's configured policy. For more information about the manner by which the `in.mpathd` daemon manages activation of underlying interfaces, refer to "How IPMP Works" on page 99. For more information about the daemon, refer to the `in.mpathd(1M)` man page.

The *IP kernel module* manages outbound load-spreading by distributing the set of available IP data addresses in the group across the set of available underlying IP interfaces in the group. The module also performs source address selection to manage inbound load-spreading. Both roles of the IP module improve network traffic performance.

The *IPMP configuration file* `/etc/default/mpathd` is used to configure the daemon's behavior. For example, you can specify how the daemon performs probe-based failure detection by setting the time duration to probe a target to detect failure, or which interfaces to probe. You can also specify what the status of a failed interface should be after that interface is repaired. You also set the parameters in this file to specify whether the daemon should monitor all IP interfaces in the system, not only those that are configured to belong to IPMP groups. For procedures to modify the configuration file, refer to "How to Configure the Behavior of the IPMP Daemon" on page 139.

The *ipmpstat utility* provides different types of information about the status of IPMP as a whole. The tool also displays other specific information about the underlying IP interfaces for each

group, as well as data and test addresses that have been configured for the group. For more information about the use of this command, see "Monitoring IPMP Information" on page 143 and the `ipmpstat(1M)` man page.

Types of IPMP Interface Configurations

An IPMP configuration typically consists of two or more physical interfaces on the same system that are attached to the same LAN. These interfaces can belong to an IPMP group in either of the following configurations:

- active-active configuration – an IPMP group in which all underlying interfaces are active. An *active interface* is an IP interface that is currently available for use by the IPMP group. By default, an underlying interface becomes active when you configure the interface to become part of an IPMP group. For additional information about active interfaces and other IPMP terms, see also "IPMP Terminology and Concepts" on page 114.
- active-standby configuration – an IPMP group in which at least one interface is administratively configured as a reserve. The reserve interface is called the *standby interface*. Although idle, the standby IP interface is monitored by the multipathing daemon to track the interface's availability, depending on how the interface is configured. If link-failure notification is supported by the interface, link-based failure detection is used. If the interface is configured with a test address, probe-based failure detection is also used. If an active interface fails, the standby interface is automatically deployed as needed. You can configure as many standby interfaces as you want for an IPMP group.

A single interface can also be configured in its own IPMP group. The single interface IPMP group has the same behavior as an IPMP group with multiple interfaces. However, this IPMP configuration does not provide high availability for network traffic. If the underlying interface fails, then the system loses all capability to send or receive traffic. The purpose of configuring a single-interfaced IPMP group is to monitor the availability of the interface by using failure detection. By configuring a test address on the interface, you can set the daemon to track the interface by using probe-based failure detection. Typically, a single-interfaced IPMP group configuration is used in conjunction with other technologies that have broader failover capabilities, such as Sun Cluster software. The system can continue to monitor the status of the underlying interface. But the Sun Cluster software provides the functionalities to ensure availability of the network when failure occurs. For more information about the Sun Cluster software, see *Sun Cluster Overview for Solaris OS*.

An IPMP group without underlying interfaces can also exist, such as a group whose underlying interfaces have been removed. The IPMP group is not destroyed, but the group cannot be used to send and receive traffic. As underlying IP interfaces are brought online for the group, then the data addresses of the IPMP interface are allocated to these interfaces and the system resumes hosting network traffic.

IPMP Addressing

You can configure IPMP failure detection on both IPv4 networks and dual-stack, IPv4 and IPv6 networks. Interfaces that are configured with IPMP support two types of addresses:

- *Data Addresses* are the conventional IPv4 and IPv6 addresses that are assigned to an IP interface dynamically at boot time by the DHCP server, or manually by using the `ifconfig` command. Data addresses are assigned to the IPMP interface. The standard IPv4 packet traffic and, if applicable, IPv6 packet traffic are considered *data traffic*. Data traffic flow use the data addresses that are hosted on the IPMP interface and flow through the active interfaces of that group.
- *Test Addresses* are IPMP-specific addresses that are used by the `in.mpathd` daemon to perform probe-based failure and repair detection. Test addresses can also be assigned dynamically by the DHCP server, or manually by using the `ifconfig` command. These addresses are configured with the `NOFAILOVER` flag that identifies them as test addresses. While data addresses are assigned to the IPMP interface, only test addresses are assigned to the underlying interfaces of the group. For an underlying interface on a dual-stack network, you can configure an IPv4 test address or an IPv6 test address or both. When an underlying interface fails, the interface's test address continues to used by the `in.mpathd` daemon for probe-based failure detection to check for the interface's subsequent repair.

Note – You need to configure test addresses only if you specifically want to use probe-based failure detection. For more information about probe-based failure detection and the use of test addresses, refer to "Probe-Based Failure Detection" on page 109.

In previous IPMP implementations, test addresses needed to be marked as `DEPRECATED` to avoid being used by applications especially during interface failures. In the current implementation, test addresses reside in the underlying interfaces. Thus, these addresses can no longer be accidentally used by applications that are unaware of IPMP. Consequently, marking test addresses as `DEPRECATED` is no longer required.

IPv4 Test Addresses

In general, you can use any IPv4 address on your subnet as a test address. IPv4 test addresses do not need to be routeable. Because IPv4 addresses are a limited resource for many sites, you might want to use non-routeable RFC 1918 private addresses as test addresses. Note that the `in.mpathd` daemon exchanges only ICMP probes with other hosts on the same subnet as the test address. If you do use RFC 1918-style test addresses, be sure to configure other systems, preferably routers, on the network with addresses on the appropriate RFC 1918 subnet. The `in.mpathd` daemon can then successfully exchange probes with target systems. For more information about RFC 1918 private addresses, refer to RFC 1918, Address Allocation for Private Internets (`http://www.ietf.org/rfc/rfc1918.txt?number=1918`).

IPv6 Test Addresses

The only valid IPv6 test address is the link-local address of a physical interface. You do not need a separate IPv6 address to serve as an IPMP test address. The IPv6 link-local address is based on the Media Access Control (MAC) address of the interface. Link-local addresses are automatically configured when the interface becomes IPv6-enabled at boot time or when the interface is manually configured through `ifconfig`. Just like IPv4 test addresses, IPv6 test addresses must be configured with the `NOFAILOVER` flag.

For more information on link-local addresses, refer to "Link-Local Unicast Address" in *System Administration Guide: IP Services*.

When an IPMP group has both IPv4 and IPv6 plumbed on all the group's interfaces, you do not need to configure separate IPv4 test addresses. The `in.mpathd` daemon can use the IPv6 link-local addresses with the `NOFAILOVER` flag as test addresses.

Failure and Repair Detection in IPMP

To ensure continuous availability of the network to send or receive traffic, IPMP performs failure detection on the IPMP group's underlying IP interfaces. Failed interfaces remain unusable until these are repaired. Remaining active interfaces continue to function while any existing standby interfaces are deployed as needed.

A *group failure* occurs when all interfaces in an IPMP group appear to fail at the same time. In this case, no underlying interface is usable. Also, when all the target systems fail at the same time and probe-based failure detection is enabled, the `in.mpathd` daemon flushes all of its current target systems and probes for new target systems.

Types of Failure Detection in IPMP

The `in.mpathd` daemon handles the following types of failure detection:

- Link-based failure detection, if supported by the NIC driver
- Probe-based failure detection, when test addresses are configured
- Detection of interfaces that were missing at boot time

Link-Based Failure Detection

Link-based failure detection is always enabled, provided that the interface supports this type of failure detection.

To determine whether a third-party interface supports link-based failure detection, use the `ipmpstat -i` command. If the output for a given interface includes an `unknown` status for its `LINK` column, then that interface does not support link-based failure detection. Refer to the manufacturer's documentation for more specific information about the device.

These network drivers that support link-based failure detection monitor the interface's link state and notify the networking subsystem when that link state changes. When notified of a change, the networking subsystem either sets or clears the `RUNNING` flag for that interface, as appropriate. If the `in.mpathd` daemon detects that the interface's `RUNNING` flag has been cleared, the daemon immediately fails the interface.

Probe-Based Failure Detection

The multipathing daemon performs probe-based failure detection on each interface in the IPMP group that has a test address. Probe-based failure detection involves sending and receiving ICMP probe messages that use test addresses. These messages, also called *probe traffic* or test traffic, go out over the interface to one or more target systems on the same local network. The daemon probes all the targets separately through all the interfaces that have been configured for probe-based failure detection. If no replies are made in response to five consecutive probes on a given interface, `in.mpathd` considers the interface to have failed. The probing rate depends on the *failure detection time* (FDT). The default value for failure detection time is 10 seconds. However, you can tune the failure detection time in the IPMP configuration file. For instructions, go to "How to Configure the Behavior of the IPMP Daemon" on page 139. To optimize probe-based failure detection, you must set multiple target systems to receive the probes from the multipathing daemon. By having multiple target systems, you can better determine the nature of a reported failure. For example, the absence of a response from the only defined target system can indicate a failure either in the target system or in one of the IPMP group's interfaces. By contrast, if only one system among several target systems does not respond to a probe, then the failure is likely in the target system rather than in the IPMP group itself.

Repair detection time is twice the failure detection time. The default time for failure detection is 10 seconds. Accordingly, the default time for repair detection is 20 seconds. After a failed interface has been repaired and the interface's `RUNNING` flag is once more detected, `in.mpathd` clears the interface's `FAILED` flag. The repaired interface is redeployed depending on the number of active interfaces that the administrator has originally set.

The `in.mpathd` daemon determines which target systems to probe dynamically. First the daemon searches the routing table for target systems that are on the same subnet as the test addresses that are associated with the IPMP group's interfaces. If such targets are found, then the daemon uses them as targets for probing. If no target systems are found on the same subnet, then `in.mpathd` sends multicast packets to probe neighbor hosts on the link. The multicast packet is sent to the all hosts multicast address, `224.0.0.1` in IPv4 and `ff02::1` in IPv6, to determine which hosts to use as target systems. The first five hosts that respond to the echo packets are chosen as targets for probing. If `in.mpathd` cannot find routers or hosts that responded to the multicast probes, then ICMP echo packets, `in.mpathd` cannot detect probe-based failures. In this case, the `ipmpstat -i` utility will report the probe state as `unknown`.

You can use host routes to explicitly configure a list of target systems to be used by `in.mpathd`. For instructions, refer to "Configuring for Probe-Based Failure Detection" on page 138.

NICs That Are Missing at Boot

NICs that are not present at system boot represent a special instance of failure detection. At boot time, the startup scripts track any interfaces with `/etc/hostname.`*interface* files. Any data addresses in such an interface's `/etc/hostname.`*interface* file are automatically configured on the corresponding IPMP interface for the group. However, if the interfaces themselves cannot be plumbed because they are missing, then error messages similar to the following are displayed:

```
moving addresses from missing IPv4 interfaces: hme0 (moved to ipmp0)
moving addresses from missing IPv6 interfaces: hme0 (moved to ipmp0)
```

Note – In this instance of failure detection, only data addresses that are explicitly specified in the missing interface's `/etc/hostname.`*interface* file are moved to the IPMP interface.

If an interface with the same name as another interface that was missing at system boot is reattached using DR, the Reconfiguration Coordination Manager (RCM) automatically plumbs the interface. Then, RCM configures the interface according to the contents of the interface's `/etc/hostname.`*interface* file. However, data addresses, which are addresses without the `NOFAILOVER` flag, that are in the `/etc/hostname.`*interface* file are ignored. This mechanism adheres to the rule that data addresses should be in the `/etc/hostname.`*ipmp-interface* file, and only test addresses should be in the underlying interface's `/etc/hostname.`*interface* file. Issuing the `ifconfig group` command causes that interface to again become part of the group. Thus, the final network configuration is identical to the configuration that would have been made if the system had been booted with the interface present.

For more information about missing interfaces, see "About Missing Interfaces at System Boot" on page 143.

Failure Detection and the Anonymous Group Feature

IPMP supports failure detection in an anonymous group. By default, IPMP monitors the status only of interfaces that belong to IPMP groups. However, the IPMP daemon can be configured to also track the status of interfaces that do not belong to any IPMP group. Thus, these interfaces are considered to be part of an "anonymous group." When you issue the command `ipmpstat -g`, the anonymous group will be displayed as double-dashes (`--`). In anonymous groups, the interfaces would have their data addresses function also as test addresses. Because these interfaces do not belong to a named IPMP group, then these addresses are visible to applications. To enable tracking of interfaces that are not part of an IPMP group, see "How to Configure the Behavior of the IPMP Daemon" on page 139.

Detecting Physical Interface Repairs

When an underlying interface fails and probe-based failure detection is used, the `in.mpathd` daemon continues to use the interface's test address to continue probing target systems. During an interface repair, the restoration proceeds depending on the original configuration of the failed interface:

- Failed interface was originally an active interface – the repaired interface reverts to its original active status. The standby interface that functioned as a replacement during the failure is switched back to standby status if enough interfaces are active for the group as defined by the system administrator.

  Note – An exception to this step are cases when the repaired active interface is also configured with the `FAILBACK=no` mode. For more information, see "The `FAILBACK=no` Mode" on page 111

- Failed interface was originally a standby interface – the repaired interface reverts to its original standby status, provided that the IPMP group reflects the original number of active interfaces. Otherwise, the standby interface is switched to become an active interface.

To see a graphical presentation of how IPMP behaves during interface failure and repair, see "How IPMP Works" on page 99.

The `FAILBACK=no` Mode

By default, active interfaces that have failed and then repaired automatically return to become active interfaces in the group. This behavior is controlled by the setting of the `FAILBACK` parameter in the daemon's configuration file. However, even the insignificant disruption that occurs as data addresses are remapped to repaired interfaces might not be acceptable to some administrators. The administrators might prefer to allow an activated standby interface to continue as an active interface. IPMP allows administrators to override the default behavior to prevent an interface to automatically become active upon repair. These interfaces must be configured in the `FAILBACK=no` mode. For related procedures, see "How to Configure the Behavior of the IPMP Daemon" on page 139.

When an active interface in `FAILBACK=no` mode fails and is subsequently repaired, the IPMP daemon restores the IPMP configuration as follows:

- The daemon retains the interface's `INACTIVE` status, provided that the IPMP group reflects the original configuration of active interfaces.
- If the IPMP configuration at the moment of repair does not reflect the group's original configuration of active interfaces, then the repaired interface is redeployed as an active interface, notwithstanding the `FAILBACK=no` status.

Note – The `FAILBACK=NO` mode is set for the whole IPMP group. It is not a per-interface tunable parameter.

IPMP and Dynamic Reconfiguration

Dynamic reconfiguration (DR) feature allows you to reconfigure system hardware, such as interfaces, while the system is running. DR can be used only on systems that support this feature.

You typically use the `cfgadm` command to perform DR operations. However, some platforms provide other methods. Make sure to consult your platform's documentation for details to perform DR. For systems that use the Solaris OS, you can find specific documentation about DR in the resources that are listed in Table 7–1. Current information about DR is also available at `http://docs.sun.com` and can be obtained by searching for the topic "dynamic reconfiguration."

TABLE 7–1 Documentation Resources for Dynamic Reconfiguration

Description	For Information
Detailed information on the `cfgadm` command	`cfgadm`(1M) man page
Specific information about DR in the Sun Cluster environment	*Sun Cluster 3.1 System Administration Guide*
Specific information about DR in the Sun Fire environment	*Sun Fire 880 Dynamic Reconfiguration Guide*
Introductory information about DR and the `cfgadm` command	Chapter 6, "Dynamically Configuring Devices (Tasks)," in *System Administration Guide: Devices and File Systems*
Tasks for administering IPMP groups on a system that supports DR	"Recovering an IPMP Configuration With Dynamic Reconfiguration" on page 141

The sections that follow explain how DR interoperates with IPMP.

On a system that supports DR of NICs, IPMP can be used to preserve connectivity and prevent disruption of existing connections. IPMP is integrated into the Reconfiguration Coordination Manager (RCM) framework. Thus, you can safely attach, detach, or reattach NICs and RCM manages the dynamic reconfiguration of system components.

Attaching New NICs

With DR support, you can attach, plumb, and then add new interfaces to existing IPMP groups. Or, if appropriate, you can configure the newly added interfaces into their own IPMP group. For procedures to configure IPMP groups, refer to "Configuring IPMP Groups" on page 124. After these interfaces have been configured, they are immediately available for use by IPMP. However, to benefit from the advantages of using customized link names, you must assign generic link names to replace the interface's hardware-based link names. Then you create corresponding configuration files by using the generic name that you just assigned. For procedures to configure a singe interface by using customized link names, refer to "How to Configure an IP Interface After System Installation" on page 40. After you assign a generic link name to interface, make sure that you always refer to the generic name when you perform any additional configuration on the interface such as using the interface for IPMP.

Detaching NICs

All requests to detach system components that contain NICs are first checked to ensure that connectivity can be preserved. For instance, by default you cannot detach a NIC that is not in an IPMP group. You also cannot detach a NIC that contains the only functioning interfaces in an IPMP group. However, if you must remove the system component, you can override this behavior by using the `-f` option of `cfgadm`, as explained in the `cfgadm(1M)` man page.

If the checks are successful, the daemon sets the `OFFLINE` flag for the interface. All test addresses on the interfaces are unconfigured. Then, the NIC is unplumbed from the system. If any of these steps fail, or if the DR of other hardware on the same system component fails, then the previous configuration is restored to its original state. A status message about this event will be displayed. Otherwise, the detach request completes successfully. You can remove the component from the system. No existing connections are disrupted.

Replacing NICs

When an underlying interface of an IPMP group fails, a typical solution would be to replace the failed interface by attaching a new NIC. RCM records the configuration information associated with any NIC that is detached from a running system. If you replace a failed NIC with an *identical* NIC, then RCM automatically configures the interface according to the contents of the existing `/etc/hostname.`*interface* file.

For example, suppose you replace a failed `bge0` interface with another `bge0` interface. The failed `bge0` already has a corresponding `/etc/hostname.bge0` file. After you attach the replacement `bge` NIC, RCM plumbs and then configures the `bge0` interface by using the information in the `/etc/hostname.bge0` file. Thus the interface is properly configured with the test address and is added to the IPMP group according to the contents of the configuration file.

You can replace a failed NIC with a different NIC, provided that both are the same type, such as ethernet. In this case, RCM plumbs the new interface after it is attached. If you did not use customized link names when you first configured your interfaces, and no corresponding configuration file for the new interface exists, then you will have to perform additional configuration steps. You will need to create a new corresponding configuration file for the new NIC. Additionally, you will need to add correct information to the file before you can add the interface to the IPMP group.

However, if you used customized link names, the additional configuration steps are unnecessary. By reassigning the failed interface's link name to the new interface, then the new interface acquires the configuration specified in the removed interface's configuration file. RCM then configures the interface by using the information in that file. For procedures to recover your IPMP configuration by using DR when an interface fails, refer to "Recovering an IPMP Configuration With Dynamic Reconfiguration" on page 141.

IPMP Terminology and Concepts

This section introduces terms and concepts that are used throughout the IPMP chapters in this book.

active interface

Refers to an underlying interface that can be used by the system to send or receive data traffic. An interface is active if the following conditions are met:

- At least one IP address is `UP` in the interface. See `UP` address.
- The `FAILED`, `INACTIVE`, or `OFFLINE` flag is not set on the interface.
- The interface has not been flagged as having a duplicate hardware address.

Compare to unusable interface, `INACTIVE` interface.

data address

Refers to an IP address that can be used as the source or destination address for data. Data addresses are part of an IPMP group and can be used to send and receive traffic on any interface in the group. Moreover, the set of data addresses in an IPMP group can be used continuously, provided that one interface in the group is functioning. In previous IPMP implementations, data addresses were hosted on the underlying interfaces of an IPMP group. In the current implementation, data addresses are hosted on the IPMP interface.

`DEPRECATED` address

Refers to an IP address that cannot be used as the source address for data. Typically, IPMP test addresses are

	DEPRECATED. However, any address can be marked DEPRECATED to prevent the address from being used as a source address.
dynamic reconfiguration	Refers to a feature that allows you to reconfigure a system while the system is running, with little or no impact on ongoing operations. Not all Sun platforms support DR. Some Sun platforms might only support DR of certain types of hardware. On platforms that support DR of NICs, IPMP can be used for uninterrupted network access to the system during DR. For more information about how IPMP supports DR, refer to "IPMP and Dynamic Reconfiguration" on page 112.
explicit IPMP interface creation	Applies only to the current IPMP implementation. The term refers to the method of creating an IPMP interface by using the `ifconfig ipmp` command. Explicit IPMP interface creation is the preferred method for creating IPMP groups. This method allows the IPMP interface name and IPMP group name to be set by the administrator. Compare to implicit IPMP interface creation.
`FAILBACK=no` mode	Refers to a setting of an underlying interface that minimizes rebinding of incoming addresses to interfaces by avoiding redistribution during interface repair. Specifically, when an interface repair is detected, the interface's `FAILED` flag is cleared. However, if the mode of the repaired interface is `FAILBACK=no`, then the `INACTIVE` flag is also set to prevent use of the interface, provided that a second functioning interface also exists. If the second interface in the IPMP group fails, then the `INACTIVE` interface is eligible to take over. While the concept of failback no longer applies in the current IPMP implementation, the name of this mode is preserved for administrative compatibility.
`FAILED` interface	Indicates an interface that the `in.mpathd` daemon has determined to be malfunctioning. The determination is achieved by either link-based or probe-based failure detection. The `FAILED` flag is set on any failed interface.
failure detection	Refers to the process of detecting when a physical interface or the path from an interface to an Internet layer device no

	longer works. Two forms of failure detection are implemented: link-based failure detection, and probe-based failure detection.
implicit IPMP interface creation	Refers to the method of creating an IPMP interface by using the `ifconfig` command to place an underlying interface in a nonexistent IPMP group. Implicit IPMP interface creation is supported for backward compatibility with the previous IPMP implementation. Thus, this method does not provide the ability to set the IPMP interface name or IPMP group name. Compare to explicit IPMP interface creation.
`INACTIVE` interface	Refers to an interface that is functioning but is not being used according to administrative policy. The `INACTIVE` flag is set on any `INACTIVE` interface. Compare to active interface, unusable interface.
IPMP anonymous group support	Indicates an IPMP feature in which the IPMP daemon tracks the status of all network interfaces in the system, regardless of whether they belong to an IPMP group. However, if the interfaces are not actually in an IPMP group, then the addresses on these interfaces are not available in case of interface failure.
IPMP group	Refers to a set of network interfaces that are treated as interchangeable by the system in order to improve network availability and utilization. Each IPMP group has a set of data addresses that the system can associate with any set of active interfaces in the group. Use of this set of data addresses maintains network availability and improves network utilization. The administrator can select which interfaces to place into an IPMP group. However, all interfaces in the same group must share a common set of properties, such as being attached to the same link and configured with the same set of protocols (for example, IPv4 and IPv6).
IPMP group interface	See IPMP interface.
IPMP group name	Refers to the name of an IPMP group, which can be assigned with the `ifconfig group` subcommand. All underlying interfaces with the same IPMP group name are defined as part of the same IPMP group. In the current implementation, IPMP group names are de-emphasized in

	favor of IPMP interface names. Administrators are encouraged to use the same name for both the IPMP interface and the group.
IPMP interface	Applies only to the current IPMP implementation. The term refers to the IP interface that represents a given IPMP group, any or all of the interface's underlying interfaces, and all of the data addresses. In the current IPMP implementation, the IPMP interface is the core component for administering an IPMP group, and is used in routing tables, ARP tables, firewall rules, and so forth.
IPMP interface name	Indicates the name of an IPMP interface. This document uses the naming convention of `ipmp`*N*. The system also uses the same naming convention in implicit IPMP interface creation. However, the administrator can choose any name by using explicit IPMP interface creation.
IPMP singleton	Refers to an IPMP configuration that is used by Sun Cluster software that allows a data address to also act as a test address. This configuration applies, for instance, when only one interface belongs to an IPMP group.
link-based failure detection	Specifies a passive form of failure detection, in which the link status of the network card is monitored to determine an interface's status. Link-based failure detection only tests whether the link is up. This type of failure detection is not supported by all network card drivers. Link-based failure detection requires no explicit configuration and provides instantaneous detection of link failures. Compare to probe-based failure detection.
load spreading	Refers to the process of distributing inbound or outbound traffic over a set of interfaces. Unlike load balancing, load spreading does not guarantee that the load is evenly distributed. With load spreading, higher throughput is achieved. Load spreading occurs only when the network traffic is flowing to multiple destinations that use multiple connections. Inbound load spreading indicates the process of distributing inbound traffic across the set of interfaces in an IPMP group. Inbound load spreading cannot be controlled directly with IPMP. The process is indirectly manipulated by the source address selection algorithm.

	Outbound load spreading refers to the process of distributing outbound traffic across the set of interfaces in an IPMP group. Outbound load spreading is performed on a per-destination basis by the IP module, and is adjusted as necessary depending on the status and members of the interfaces in the IPMP group.
`NOFAILOVER` address	Applies only to the previous IPMP implementation. Refers to an address that is associated with an underlying interface and thus remains unavailable if the underlying interface fails. All `NOFAILOVER` addresses have the `NOFAILOVER` flag set. IPMP test addresses must be designated as `NOFAILOVER`, while IPMP data addresses must never be designated as `NOFAILOVER`. The concept of failover does not exist in the IPMP implementation. However, the term `NOFAILOVER` remains for administrative compatibility.
`OFFLINE` interface	Indicates an interface that has been administratively disabled from system use, usually in preparation for being removed from the system. Such interfaces have the `OFFLINE` flag set. The `if_mpadm` command can be used to switch an interface to an offline status.
physical interface	See: underlying interface
probe	Refers to an ICMP packet, similar to the packets that are used by the `ping` command. This probe is used to test the send and receive paths of a given interface. Probe packets are sent by the `in.mpathd` daemon, if probe-based failure detection is enabled. A probe packet uses an IPMP test address as its source address.
probe-based failure detection	Indicates an active form of failure detection, in which probes are exchanged with probe targets to determine an interface's status. When enabled, probe-based failure detection tests the entire send and receive path of each interface. However, this type of detection requires the administrator to explicitly configure each interface with a test address. Compare to link-based failure detection.
probe target	Refers to a system on the same link as an interface in an IPMP group. The target is selected by the `in.mpathd` daemon to help check the status of a given interface by using probe-based failure detection. The probe target can

	be any host on the link that is capable of sending and receiving ICMP probes. Probe targets are usually routers. Several probe targets are usually used to insulate the failure detection logic from failures of the probe targets themselves.
source address selection	Refers to the process of selecting a data address in the IPMP group as the source address for a particular packet. Source address selection is performed by the system whenever an application has not specifically selected a source address to use. Because each data address is associated with only one hardware address, source address selection indirectly controls inbound load spreading.
`STANDBY` interface	Indicates an interface that has been administratively configured to be used only when another interface in the group has failed. All `STANDBY` interfaces will have the `STANDBY` flag set.
test address	Refers to an IP address that must be used as the source or destination address for probes, and must not be used as a source or destination address for data traffic. Test addresses are associated with an underlying interface. These addresses are designated as `NOFAILOVER` so that they remain on the underlying interface even if the interface fails to facilitate repair detection. Because test addresses are not available upon interface failure, all test addresses must be designated as `DEPRECATED` to keep the system from using them as a source addresses for data packets.
underlying interface	Specifies an IP interface that is part of an IPMP group and is directly associated with an actual network device. For example, if `ce0` and `ce1` are placed into IPMP group `ipmp0`, then `ce0` and `ce1` comprise the underlying interfaces of `ipmp0`. In the previous implementation, IPMP groups consist solely of underlying interfaces. However, in the current implementation, these interfaces underlie the IPMP interface (for example, `ipmp0`) that represents the group, hence the name.
undo-offline operation	Refers to the act of administratively enabling a previously offlined interface to be used by the system. The `if_mpadm` command can be used to perform an undo-offline operation.

unusable interface	Refers to an underlying interface that cannot be used to send or receive data traffic at all in its current configuration. An unusable interface differs from an `INACTIVE` interface, that is not currently being used but can be used if an active interface in the group becomes unusable. An interface is unusable if one of the following conditions exists: ■ The interface has no `UP` address. ■ The `FAILED` or `OFFLINE` flag has been set for the interface. ■ The interface has been flagged has having the same hardware address as another interface in the group.
target systems	See probe target.
`UP` address	Refers to an address that has been made administratively available to the system by setting the `UP` flag. An address that is not `UP` is treated as not belonging to the system, and thus is never considered during source address selection.

◆ ◆ ◆ CHAPTER 8

Administering IPMP

This chapter provides tasks for administering interface groups with IP network multipathing (IPMP). The following major topics are discussed:

IPMP Administration Task Maps

In this Solaris release, the `ipmpstat` command is the preferred tool to use to obtain information about IPMP group information. In this chapter, the `ipmpstat` command replaces certain functions of the `ifconfig` command that were used in previous Solaris releases to provide IPMP information.

For information about the different options for the `ipmpstat` command, see "Monitoring IPMP Information" on page 143.

This following sections provide links to the tasks in this chapter.

IPMP Group Creation and Configuration (Task Map)

Task	Description	For Instructions
Plan an IPMP group.	Lists all ancillary information and required tasks before you can configure an IPMP group.	"How to Plan an IPMP Group" on page 124
Configure an IPMP group by using DHCP.	Provides an alternative method to configure IPMP groups by using DHCP.	"How to Configure an IPMP Group by Using DHCP" on page 125
Configure an active-active IPMP group.	Configures an IPMP group in which all underlying interfaces are deployed to host network traffic.	"How to Manually Configure an Active-Active IPMP Group" on page 128
Configure an active-standby IPMP group.	Configures an IPMP group in which one underlying interface is kept inactive as a reserve.	"How to Manually Configure an Active-Standby IPMP Group" on page 131

IPMP Group Maintenance (Task Map)

Task	Description	For Instructions
Add an interface to an IPMP group.	Configures a new interface as a member of an existing IPMP group.	"How to Add an Interface to an IPMP Group" on page 134
Remove an interface from an IPMP group.	Removes an interface from an IPMP group.	"How to Remove an Interface From an IPMP Group" on page 135
Add IP addresses to or remove IP addresses from an IPMP group.	Adds or removes addresses for an IPMP group.	"How to Add or Remove IP Addresses" on page 135
Change an interface's IPMP membership.	Moves interfaces among IPMP groups.	"How to Move an Interface From One IPMP Group to Another Group" on page 136
Delete an IPMP group.	Deletes an IPMP group that is no longer needed.	"How to Delete an IPMP Group" on page 137
Replace cards that failed.	Removes or replaces failed NICs of an IPMP group.	"How to Replace a Physical Card That Has Failed" on page 141

Probe-Based Failure Detection Configuration (Task Map)

Task	Description	For Instructions
Manually specify target systems	Identifies and adds systems to be targeted for probe-based failure detection.	"How to Manually Specify Target Systems for Probe-Based Failure Detection" on page 139
Configure the behavior of probe-based failure detection.	Modifies parameters to determine the behavior of probe-based failure detection.	"How to Configure the Behavior of the IPMP Daemon" on page 139

IPMP Group Monitoring (Task Map)

Task	Description	For Instructions
Obtain group information.	Displays information about an IPMP group.	"How to Obtain IPMP Group Information" on page 144
Obtain data address information.	Displays information about the data addresses that are used by an IPMP group.	"How to Obtain IPMP Data Address Information" on page 145
Obtain IPMP interface information.	Displays information about the underlying interfaces of IPMP interfaces or groups.	"How to Obtain Information About Underlying IP Interfaces of a Group" on page 145
Obtain probe target information.	Displays information about targets of probe-based failure detection.	"How to Obtain IPMP Probe Target Information" on page 147
Obtain probe information.	Displays real-time information about ongoing probes in the system.	"How to Observe IPMP Probes" on page 148
Customize the information display for monitoring IPMP groups.	Determines the IPMP information that is displayed.	"How to Customize the Output of the `ipmpstat` Command in a Script" on page 149

Configuring IPMP Groups

This section provides procedures that are used to plan and configure IPMP groups.

▼ How to Plan an IPMP Group

The following procedure includes the required planning tasks and information to be gathered prior to configuring an IPMP group. The tasks do not have to be performed in sequence.

1 Determine the general IPMP configuration that would suit your needs.

Your IPMP configuration depends on what your network needs to handle the type of traffic that is hosted on your system. IPMP spreads outbound network packets across the IPMP group's interfaces, and thus improves network throughput. However, for a given TCP connection, inbound traffic normally follows only one physical path to minimize the risk of processing out-of-order packets.

Thus, if your network handles a huge volume of outbound traffic, configuring multiple interfaces into an IPMP group can improve network performance. If instead, the system hosts heavy inbound traffic, then the number of interfaces in the group does not necessarily improve performance by load spreading traffic. However, having multiple interfaces helps to guarantee network availability during interfaces failure.

2 For SPARC based systems, verify that each interface in the group has a unique MAC address.

To configure a unique MAC address for each interface in the system, see "SPARC: How to Ensure That the MAC Address of an Interface Is Unique" on page 38.

3 Ensure that the same set of STREAMS modules is pushed and configured on all interfaces in the IPMP group.

All interfaces in the same group must have the same STREAMS modules configured in the same order.

a. Check the order of STREAMS modules on all interfaces in the prospective IPMP group.

You can print a list of STREAMS modules by using the `ifconfig` *interface* `modlist` command. For example, here is the `ifconfig` output for an `hme0` interface:

```
# ifconfig hme0 modlist
    0 arp
    1 ip
    2 hme
```

Interfaces normally exist as network drivers directly below the IP module, as shown in the output from `ifconfig hme0 modlist`. They should not require additional configuration.

However, certain technologies insert themselves as a STREAMS module between the IP module and the network driver. If a STREAMS module is stateful, then unexpected behavior

can occur on failover, even if you push the same module onto all of the interfaces in a group. However, you can use stateless STREAMS modules, provided that you push them in the same order on all interfaces in the IPMP group.

b. Push the modules of an interface in the standard order for the IPMP group.

`ifconfig` *interface* **`modinsert`** *module-name*`@`*position*

```
ifconfig hme0 modinsert vpnmod@3
```

4 Use the same IP addressing format on all interfaces of the IPMP group.

If one interface is configured for IPv4, then all interfaces of the group must be configured for IPv4. For example, if you add IPv6 addressing to one interface, then all interfaces in the IPMP group must be configured for IPv6 support.

5 Determine the type of failure detection that you want to implement.

For example, if you want to implement probe-based failure detection, then you must configure test addresses on the underlying interfaces. For related information, see"Types of Failure Detection in IPMP" on page 108.

6 Ensure that all interfaces in the IPMP group are connected to the same local network.

For example, you can configure Ethernet switches on the same IP subnet into an IPMP group. You can configure any number of interfaces into an IPMP group.

Note – You can also configure a single interface IPMP group, for example, if your system has only one physical interface. For related information, see "Types of IPMP Interface Configurations" on page 106.

---

7 Ensure that the IPMP group does not contain interfaces with different network media types.

The interfaces that are grouped together should be of the same interface type, as defined in `/usr/include/net/if_types.h`. For example, you cannot combine Ethernet and Token ring interfaces in an IPMP group. As another example, you cannot combine a Token bus interface with asynchronous transfer mode (ATM) interfaces in the same IPMP group.

8 For IPMP with ATM interfaces, configure the ATM interfaces in LAN emulation mode.

IPMP is not supported for interfaces using Classical IP over ATM.

▼ How to Configure an IPMP Group by Using DHCP

In the current IPMP implementation, IPMP groups can be configured with Dynamic Host Configuration Protocol (DHCP) support.

A multiple-interfaced IPMP group can be configured with active-active interfaces or active-standby interfaces. For related information, see "Types of IPMP Interface Configurations" on page 106. The following procedure describes steps to configure an active-standby IPMP group by using DHCP.

Before You Begin Make sure that IP interfaces that will be in the prospective IPMP group have been correctly configured over the system's network data links. For procedures to configure links and IP interfaces, see "Data Link and IP Interface Configuration (Tasks)" on page 37. For information about configuring IPv6 interfaces, see "Configuring an IPv6 Interface" in *System Administration Guide: IP Services*.

Additionally, if you are using a SPARC system, configure a unique MAC address for each interface. For procedures, see "SPARC: How to Ensure That the MAC Address of an Interface Is Unique" on page 38.

Finally, if you are using DHCP, make sure that the underlying interfaces have infinite leases. Otherwise, in case of a group failure, the test addresses will expire and the IPMP daemon will then revert to link-based failure detection. Such circumstances would trigger errors in the manner the group's failure detection behaves during interface recovery. For more information about configuring DHCP, refer to Chapter 12, "Planning for DHCP Service (Tasks)," in *System Administration Guide: IP Services*.

1 On the system on which you want to configure the IPMP group, assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 Create an IPMP interface.

```
# ifconfig ipmp-interface ipmp [group group-name]
```

Note – To configure IPv6 IPMP interfaces, use the same command syntax for configuring IPv6 interfaces by specifying `inet6` in the `ifconfig` command, for example:

```
# ifconfig ipmp-interface inet6 ipmp [group group-name]
```

This note applies to all configuration procedures that involve IPv6 IPMP interfaces.

---

ipmp-interface Specifies the name of the IPMP interface. You can assign any meaningful name to the IPMP interface. As with any IP interface, the name consists of a string and a number, such as `ipmp0`.

group-name Specifies the name of the IPMP group. The name can be any name of your choice. Assigning a group name is optional. By default, the name of the IPMP interface also becomes the name of the IPMP group. Preferably, retain this default setting by not using the *group-name* option.

Note – The syntax in this step uses the preferred explicit method of creating an IPMP group by creating the IPMP interface.

An alternative method to create an IPMP group is implicit creation, in which you use the syntax `ifconfig` *interface* `group` *group-name*. In this case, the system creates the lowest available `ipmp`*N* to become the group's IPMP interface. For example, if `ipmp0` already exists for group `acctg`, then the syntax `ifconfig ce0 group fieldops` causes the system to create `ipmp1` for group `fieldops`. All `UP` data addresses of `ce0` are then assigned to `ipmp1`.

However, implicit creation of IPMP groups is not encouraged. Support for implicit creation is provided only to have compatible implementation with previous Solaris releases. Explicit creation provides optimal control over the configuration of IPMP interfaces.

---

3 Add underlying IP interfaces that will contain test addresses to the IPMP group, including the standby interface.

```
# ifconfig interface group group-name -failover [standby] up
```

4 Have DHCP configure and manage the data addresses on the IPMP interface.

You need to plumb as many logical IPMP interfaces as data addresses, and then have DHCP configure and manage the addresses on these interfaces as well.

```
# ifconfig ipmp-interface dhcp start primary
# ifconfig ipmp-interface:n plumb
# ifconfig ipmp-interface:n dhcp start
```

5 Have DHCP manage the test addresses in the underlying interfaces.

You need to issue the following command for each underlying interface of the IPMP group.

```
# ifconfig interface dhcp start
```

Example 8–1 Configuring an IPMP Group With DHCP

This example shows how to configure an active-standby IPMP group with DHCP. This example is based on Figure 7–1, which contains the following information:

- Three underlying interfaces, `subitops0`, `subitops1`, and `subitops2` are designated members of the IPMP group.
- The IPMP interface `itops0` shares the same name with the IPMP group.
- `subitops2` is the designated standby interface.

- To use probe-based failure detection, all the underlying interfaces are assigned test addresses.

```
# ifconfig itops0 ipmp

# ifconfig subitops0 plumb group itops0 -failover up
# ifconifg subitops1 plumb group itops0 -failover up
# ifconfig subitops2 plumb group itops0 -failover standby up

# ifconfig itops0 dhcp start primary
# ifconfig itops0:1 plumb
# ifconfig itops0:1 dhcp start

# ifconfig subitops0 dhcp start
# ifconfig subitops1 dhcp start
# ifconfig subitops2 dhcp start
```

To make the test address configuration persistent, you would need to type the following commands:

```
# touch /etc/dhcp.itops0 /etc/dhcp.itops0:1
# touch /etc/dhcp.subitops0 /etc/dhcp.subitops1 /etc/dhcp.subitops2

# echo group itops0 -failover up > /etc/hostname.subitops0
# echo group itops0 -failover up > /etc/hostname.subitops1
# echo group itops0 -failover standby up > /etc/hostname.subitops2
# echo ipmp > /etc/hostname.itops0
```

▼ How to Manually Configure an Active-Active IPMP Group

The following procedure describes steps to manually configure an active-active IPMP group.

Before You Begin Make sure that IP interfaces that will be in the prospective IPMP group have been correctly configured over the system's network data links. For procedures to configure links and IP interfaces, see "Data Link and IP Interface Configuration (Tasks)" on page 37. For information about configuring IPv6 interfaces, see "Configuring an IPv6 Interface" in *System Administration Guide: IP Services*.

Additionally, if you are using a SPARC system, configure a unique MAC address for each interface. For procedures, see "SPARC: How to Ensure That the MAC Address of an Interface Is Unique" on page 38.

1 On the system on which you want to configure the IPMP group, assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 Create an IPMP interface.

```
# ifconfig ipmp-interface ipmp [group group-name]
```

ipmp-interface Specifies the name of the IPMP interface. You can assign any meaningful name to the IPMP interface. As with any IP interface, the name consists of a string and a number, such as `ipmp0`.

group-name Specifies the name of the IPMP group. The name can be any name of your choice. Any nun-null name is valid, provided that the name does not exceed 31 characters. Assigning a group name is optional. By default, the name of the IPMP interface also becomes the name of the IPMP group. Preferably, retain this default setting by not using the *group-name* option.

Note – The syntax in this step uses the preferred explicit method of creating an IPMP group by creating the IPMP interface.

An alternative method to create an IPMP group is implicit creation, in which you use the syntax `ifconfig` *interface* `group` *group-name*. In this case, the system creates the lowest available `ipmp`*N* to become the group's IPMP interface. For example, if `ipmp0` already exists for group `acctg`, then the syntax `ifconfig ce0 group fieldops` causes the system to create `ipmp1` for group `fieldops`. All `UP` data addresses of `ce0` are then assigned to `ipmp1`.

However, implicit creation of IPMP groups is not encouraged. Support for implicit creation is provided only to have compatible implementation with previous Solaris releases. Explicit creation provides optimal control over the configuration of IPMP interfaces.

---

3 Add underlying IP interfaces to the group.

```
# ifconfig ip-interface group group-name
```

Note – In a dual-stack environment, placing the IPv4 instance of an interface under a particular group automatically places the IPv6 instance under the same group as well.

---

4 Add data addresses to the IPMP interface.

```
# ifconfig plumb ipmp-interface ip-address up
# ifconfig ipmp-interface addif ip-address up
```

For additional options that you can use with the `ifconfig` command while adding addresses, refer to the `ifconfig(1M)` man page.

5 Configure test addresses on the underlying interfaces.

```
# ifconfig interface -failover ip-address up
```

Note – You need to configure a test address only if you want to use probe-based failure detection on a particular interface.

All test IP addresses in an IPMP group must use the same network prefix. The test IP addresses must belong to a single IP subnet.

---

6 (Optional) Preserve the IPMP group configuration across reboots.

To configure an IPMP group that persists across system reboots, you would edit the `hostname` configuration file of the IPMP interface to add data addresses. Then, if you want to use test addresses, you would edit the `hostname` configuration file of one of the group's underlying IP interface. Note that data and test addresses can be both IPv4 and IPv6 addresses. Perform the following steps:

a. Edit the `/etc/hostname.`*ipmp-interface* file by adding the following lines:

```
ipmp group group-name data-address up

addif data-address
...
```

You can add more data addresses on separate `addif` lines in this file.

b. Edit the `/etc/hostname.`*interface* file of the underlying IP interfaces that contain the test address by adding the following line:

```
group group-name -failover test-address up
```

Follow this same step to add test addresses to other underlying interfaces of the IPMP group.

Caution – When adding test address information on the `/etc/hostname.`*interface* file, make sure to specify the `-failover` option before the `up` keyword. Otherwise, the test IP addresses will be treated as data addresses and would cause problems for system administration. Preferably, set the `-failover` option before specifying the IP address.

How to Manually Configure an Active-Standby IPMP Group

For more information about standby interfaces, see "Types of IPMP Interface Configurations" on page 106. The following procedure configures an IPMP group where one interface is kept as a reserve. This interface is deployed only when an active interface in the group fails.

1 On the system on which you want to configure the IPMP group, assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 Create an IPMP interface.

```
# ifconfig ipmp-interface ipmp [group group-name]
```

ipmp-interface Specifies the name of the IPMP interface. You can assign any meaningful name to the IPMP interface. As with any IP interface, the name consists of a string and a number, such as `ipmp0`.

group-name Specifies the name of the IPMP group. The name can be any name of your choice. Any nun-null name is valid, provided that the name does not exceed 31 characters. Assigning a group name is optional. By default, the name of the IPMP interface also becomes the name of the IPMP group. Preferably, retain this default setting by not using the *group-name* option.

Note – The syntax in this step uses the preferred explicit method of creating an IPMP group by creating the IPMP interface.

An alternative method to create an IPMP group is implicit creation, in which you use the syntax `ifconfig` *interface* `group` *group-name*. In this case, the system creates the lowest available `ipmp`*N* to become the group's IPMP interface. For example, if `ipmp0` already exists for group `acctg`, then the syntax `ifconfig ce0 group fieldops` causes the system to create `ipmp1` for group `fieldops`. All `UP` data addresses of `ce0` are then assigned to `ipmp1`.

However, implicit creation of IPMP groups is not encouraged. Support for implicit creation is provided only to have compatible implementation with previous Solaris releases. Explicit creation provides optimal control over the configuration of IPMP interfaces.

---

3 Add underlying IP interfaces to the group.

```
# ifconfig ip-interface group group-name
```

Note – In a dual-stack environment, placing the IPv4 instance of an interface under a particular group automatically places the IPv6 instance under the same group as well.

4 Add data addresses to the IPMP interface.

```
# ifconfig plumb ipmp-interface ip-address up
# ifconfig ipmp-interface addif ip-address up
```

For additional options that you can use with the `ifconfig` command while adding addresses, refer to the `ifconfig(1M)` man page.

5 Configure test addresses on the underlying interfaces.

- To configure a test address on an active interface, use the following command:

  ```
  # ifconfig interface -failover ip-address up
  ```

- To configure a test address on a designated standby interface, use the following command:

  ```
  # ifconfig interface -failover ip-address standby up
  ```

Note – You need to configure a test address only if you want to use probe-based failure detection on a particular interface.

All test IP addresses in an IPMP group must use the same network prefix. The test IP addresses must belong to a single IP subnet.

6 (Optional) Preserve the IPMP group configuration across reboots.

To configure an IPMP group that persists across system reboots, you would edit the `hostname` configuration file of the IPMP interface to add data addresses. Then, if you want to use test addresses, you would edit the `hostname` configuration file of one of the group's underlying IP interface. Note that data and test addresses can be both IPv4 and IPv6 addresses. Perform the following steps:

a. Edit the `/etc/hostname.`*ipmp-interface* file by adding the following lines:

```
ipmp group group-name data-address up
addif data-address
...
```

You can add more data addresses on separate `addif` lines in this file.

b. Edit the `/etc/hostname.`*interface* file of the underlying IP interfaces that contain the test address by adding the following line:

```
group group-name -failover test-address up
```

Follow this same step to add test addresses to other underlying interfaces of the IPMP group. For a designated standby interface, the line must be as follows:

```
group group-name -failover test-address standby up
```

Caution – When adding test address information on the `/etc/hostname.`*interface* file, make sure to specify the `-failover` option before the `up` keyword. Otherwise, the test IP addresses will be treated as data addresses and would cause problems for system administration. Preferably, set the `-failover` option before specifying the IP address.

Example 8–2 Configuring an Active-Standby IPMP Group

This example shows how to manually create the same persistent active-standby IPMP configuration that is provided in Example 8–1.

```
# ifconfig itops0 ipmp

# ifconfig subitops0 group itops0
# ifconfig subitops1 group itops0
# ifconfig subitops2 group itops0

# ifconfig itops0 192.168.10.10/24 up
# ifconfig itops0 addif 192.168.10.15/24 up

# ifconfig subitops0 -failover 192.168.85.30/24 up
# ifconfig subitops1 -failover 192.168.86.32/24 up
# ifconfig subitops2 -failover 192.168.86.34/24 standby up

# ipmpstat -g
GROUP     GROUPNAME   STATE      FDT        INTERFACES
itops0    itops0      ok         10.00s     subitops0 subitops1 (subitops2)

# ipmpstat -t
INTERFACE      MODE     TESTADDR         TARGETS
subitops0      routes   192.168.10.30    192.168.10.1
subitops1      routes   192.168.10.32    192.168.10.1
subitops2      routes   192.168.10.34    192.168.10.5

# vi /etc/hostname.itops0
ipmp group itops0 192.168.10.10/24 up
addif 192.168.10.15/24 up

# vi /etc/hostname.subitops0
group itops0 -failover 192.168.10.30/24 up
```

```
# vi /etc/hostname.subitops1
group itops0 -failover 192.168.10.32/24 up

# vi /etc/hostname.subitops2
group itops0 -failover 192.168.10.34/24 standby up
```

Maintaining IPMP Groups

This section contains tasks for maintaining existing IPMP groups and the interfaces within those groups. The tasks presume that you have already configured an IPMP group, as explained in "Configuring IPMP Groups" on page 124.

▼ How to Add an Interface to an IPMP Group

Before You Begin Make sure that the interface that you add to the group matches all the constraints to be in the group. For a list of the requirements of an IPMP group, see "How to Plan an IPMP Group" on page 124.

1 On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 Add the IP interface to the IPMP group.

 # ifconfig *interface* **group** *group-name*

The interface specified in *interface* becomes a member of IPMP group *group-name*.

Example 8–3 Adding an Interface to an IPMP Group

To add the interface `hme0` to the IPMP group `itops0`, you would type the following command:

```
# ifconfig hme0 group itops0
# ipmpstat -g
GROUP   GROUPNAME   STATE      FDT       INTERFACES
itops0  itops0      ok         10.00s    subitops0 subitops1 hme0
```

How to Remove an Interface From an IPMP Group

1. **On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Remove the interface from the IPMP group.**

   ```
   # ifconfig interface group ""
   ```

   The quotation marks indicate a null string.

Example 8–4 Removing an Interface From a Group

To remove the interface `hme0` from the IPMP group `itops0`, you would type the following command:

```
# ifconfig hme0 group ""
# ipmpstat -g
GROUP   GROUPNAME   STATE      FDT       INTERFACES
itops0  itops0      ok         10.00s    subitops0 subitops1
```

How to Add or Remove IP Addresses

You use the `ifconfig addif` syntax to add addresses or the `ifconfig removeif` command to remove addresses from interfaces. In the current IPMP implementation, test addresses are hosted on the underlying IP interface, while data addresses are assigned to the IPMP interface. The following procedures describes how to add or remove IP addresses that are either test addresses or data addresses.

1. **Assume the role of Primary Administrator, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Add or remove data addresses.**

   - To add data addresses to the IPMP group, type the following command:

     ```
     # ifconfig ipmp-interface addif ip-address up
     ```

   - To remove an address from the IPMP group, type the following command:

```
# ifconfig ipmp-interface removeif ip-address
```

3 **Add or remove test addresses.**

- To assign a test address to an underlying interface of the IPMP group, type the following command:

  ```
  # ifconfig interface addif -failover ip-address up
  ```

- To remove a test address from an underlying interface of the IPMP group, type the following command:

  ```
  # ifconfig interface removeif ip-address
  ```

Example 8–5 Removing a Test Address From an Interface

The following example uses the configuration of `itops0` in Example 8–2. The step removes the test address from the interface `subitops0`.

```
# ipmpstat -t
INTERFACE      MODE     TESTADDR         TARGETS
subitops0      routes   192.168.10.30    192.168.10.1

# ifconfig subitops0 removeif 192.168.85.30
```

▼ How to Move an Interface From One IPMP Group to Another Group

You can place an interface in a new IPMP group when the interface belongs to an existing IPMP group. You do not need to remove the interface from the current IPMP group. When you place the interface in a new group, the interface is automatically removed from any existing IPMP group.

1 **On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Move the interface to a new IPMP group.**

```
# ifconfig interface group group-name
```

Placing the interface in a new group automatically removes the interface from any existing group.

Example 8–6 Moving an Interface to a Different IPMP Group

This example assumes that the underlying interfaces of your group are subitops0, subitops1, subitops2, and hme0. To change the IPMP group of interface hme0 to the group cs-link1, you would type the following:

```
# ifconfig hme0 group cs-link1
```

This command removes the `hme0` interface from IPMP group `itops0` and then puts the interface in the group `cs-link1`.

▼ How to Delete an IPMP Group

Use this procedure if you no longer need a specific IPMP group.

1 Assume the role of Primary Administrator, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 Identify the IPMP group and the underlying IP interfaces.

```
# ipmpstat -g
```

3 Delete all IP interfaces that currently belong to the IPMP group.

```
# ifconfig ip-interface group ""
```

Repeat this step for all the IP interfaces that belong to the group.

Note – To successfully delete an IPMP interface, no IP interface must exist as part of the IPMP group.

---

4 Delete the IPMP interface.

```
# ifconfig ipmp-interface unplumb
```

After you unplumb the IPMP interface, any IP address that is associated with the interface is deleted from the system.

5 To make the deletion persistent, perform the following additional steps:

a. Delete the IPMP interface's corresponding `hostname` file.

```
# rm /etc/hostname.ipmp-interface
```

b. Remove the "`group`" keywords in the `hostname` files of the underlying interfaces.

Example 8–7 Deleting an IPMP Interface

To delete the interface `itops0` that has the underlying IP interface `subitops0` and `subitops1`, you would type the following commands:

```
# ipmpstat -g
GROUP   GROUPNAME   STATE      FDT        INTERFACES
itops0  itops0      ok         10.00s     subitops0 subitops1

# ifconfig subitops0 group ""
# ifconfig subitops1 group ""
# ifconfig itops0 unplumb
# rm /etc/hostname.itops0
```

You would then edit the files `/etc/hostname.subitops0` and `/etc/hostname.subitops1` to remove "`group`" entries in those files.

Configuring for Probe-Based Failure Detection

Probe-based failure detection involves the use of target systems, as explained in "Probe-Based Failure Detection" on page 109. In identifying targets for probe-based failure detection, the `in.mpathd` daemon operates in two modes: router target mode or multicast target mode. In the router target mode, the multipathing daemon probes targets that are defined in the routing table. If no targets are defined, then the daemon operates in multicast target mode, where multicast packets are sent out to probe neighbor hosts on the LAN.

Preferably, you should set up host targets for the `in.mpathd` daemon to probe. For some IPMP groups, the default router is sufficient as a target. However, for some IPMP groups, you might want to configure specific targets for probe-based failure detection. To specify the targets, set up host routes in the routing table as probe targets. Any host routes that are configured in the routing table are listed before the default router. IPMP uses the explicitly defined host routes for target selection. Thus, you should set up host routes to configure specific probe targets rather than use the default router.

To set up host routes in the routing table, you use the route command. You can use the `-p` option with this command to add persistent routes. For example, `route -p add` adds a route which will remain in the routing table even after you reboot the system. The `-p` option thus allows you to add persistent routes without needing any special scripts to recreate these routes every system startup. To optimally use probe-based failure detection, make sure that you set up multiple targets to receive probes.

The sample procedure that follows shows the exact syntax to add persistent routes to targets for probe-based failure detection. For more information about the options for the `route` command, refer to the `route(1M)` man page.

Consider the following criteria when evaluating which hosts on your network might make good targets.

- Make sure that the prospective targets are available and running. Make a list of their IP addresses.
- Ensure that the target interfaces are on the same network as the IPMP group that you are configuring.
- The netmask and broadcast address of the target systems must be the same as the addresses in the IPMP group.
- The target host must be able to answer ICMP requests from the interface that is using probe-based failure detection.

How to Manually Specify Target Systems for Probe-Based Failure Detection

1. **Log in with your user account to the system where you are configuring probe-based failure detection.**

2. **Add a route to a particular host to be used as a target in probe-based failure detection.**

   ```
   $ route -p add -host destination-IP gateway-IP -static
   ```

   where *destination-IP* and *gateway-IP* are IPv4 addresses of the host to be used as a target. For example, you would type the following to specify the target system `192.168.10.137`, which is on the same subnet as the interfaces in IPMP group `itops0`:

   ```
   $ route -p add -host 192.168.10.137 192.168.10.137 -static
   ```

   This new route will be automatically configured every time the system is restarted. If you want to define only a temporary route to a target system for probe-based failure detection, then do not use the `-p` option.

3. **Add routes to additional hosts on the network to be used as target systems.**

How to Configure the Behavior of the IPMP Daemon

Use the IPMP configuration file `/etc/default/mpathd` to configure the following system-wide parameters for IPMP groups.

- FAILURE_DETECTION_TIME
- TRACK_INTERFACES_ONLY_WITH_GROUPS
- FAILBACK

1 On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 Edit the `/etc/default/mpathd` file.

Change the default value of one or more of the three parameters.

a. Type the new value for the `FAILURE_DETECTION_TIME` parameter.

```
FAILURE_DETECTION_TIME=n
```

where *n* is the amount of time in seconds for ICMP probes to detect whether an interface failure has occurred. The default is 10 seconds.

b. Type the new value for the `FAILBACK` parameter.

```
FAILBACK=[yes | no]
```

- *yes*– The *yes* value is the default for the failback behavior of IPMP. When the repair of a failed interface is detected, network access fails back to the repaired interface, as described in "Detecting Physical Interface Repairs" on page 111.
- *no* – The *no* value indicates that data traffic does not move back to a repaired interface. When a failed interfaces is detected as repaired, the `INACTIVE` flag is set for that interface. This flag indicates that the interface is currently not to be used for data traffic. The interface can still be used for probe traffic.

  For example, the IPMP group `ipmp0` consists of two interfaces, `ce0` and `ce1`. In the `/etc/default/mpathd` file, the *FAILBACK=no* parameter is set. If `ce0` fails, then it is flagged as `FAILED` and becomes unusable. After repair, the interface is flagged as `INACTIVE` and remains unusable because of the *FAILBACK=no* setting.

  If `ce1` fails and only `ce0` is in the `INACTIVE` state, then `ce0`'s `INACTIVE` flag is cleared and the interface becomes usable. If the IPMP group has other interfaces that are also in the `INACTIVE` state, then any one of these `INACTIVE` interfaces, and not necessarily `ce0`, can be cleared and become usable when `ce1` fails.

c. Type the new value for the `TRACK_INTERFACES_ONLY_WITH_GROUPS` parameter.

```
TRACK_INTERFACES_ONLY_WITH_GROUPS=[yes | no]
```

Note – For information about this parameter and the anonymous group feature, see "Failure Detection and the Anonymous Group Feature" on page 110.

- *yes*– The *yes* value is the default for the behavior of IPMP. This parameter causes IPMP to ignore network interfaces that are not configured into an IPMP group.
- *no* – The *no* value sets failure and repair detection for *all* network interfaces, regardless of whether they are configured into an IPMP group. However, when a failure or repair is detected on an interface that is not configured into an IPMP group, no action is triggered in IPMP to maintain the networking functions of that interface. Therefore, the*no* value is only useful for reporting failures and does not directly improve network availability.

3 Restart the `in.mpathd` daemon.

```
# pkill -HUP in.mpathd
```

Recovering an IPMP Configuration With Dynamic Reconfiguration

This section contains procedures that relate to administering systems that support dynamic reconfiguration (DR).

▼ How to Replace a Physical Card That Has Failed

This procedure explains how to replace a physical card on a system that supports DR. The procedure assumes the following conditions:

- You assigned administratively chosen names to the data links over which you configured the IP interfaces. These interfaces are `subitops0` and `subitops1`.
- Both interfaces belong to the IPMP group, `itops0`.
- The interface `subitops0` contains a test address.
- The interface `subitops0` has failed, and you need to remove `subitops0`'s underlying card, `ce`.
- You are replacing the `ce` card with a `bge` card.
- The configuration files correspond to the interfaces and use the interfaces' customized link names, thus `/etc/hostname.subitops0` and `/etc/hostname.subitops1`.

Before You Begin The procedures for performing DR vary with the type of system. Therefore, make sure that you complete the following:

- Ensure that your system supports DR.
- Consult the appropriate manual that describes DR procedures on your system. For Sun hardware, all systems that support DR are servers. To locate current DR documentation on Sun systems, search for "dynamic reconfiguration" on `http://docs.sun.com`.

Note – The steps in the following procedure refer only to aspects of DR that are specifically related to IPMP and the use of link names. The procedure does not contain the complete steps to perform DR. For example, some layers beyond the IP layer require manual configuration steps, such as for ATM and other services, if the configuration is not automated. Follow the appropriate DR documentation for your system.

---

1 On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 Perform the appropriate DR steps to remove the failed NIC from the system.

- If you are removing the card without intending to insert a replacement, then skip the rest of the steps after you remove the card.
- If you are replacing a card, then proceed to the subsequent steps .

3 Make sure that the replacement NIC is not being referenced by other configurations in the system.

For example, the replacement NIC you install is `bge0`. If a `/etc/hostname.bge0` file exists on the system, remove that file.

```
# rm /etc/hostname.bge0
```

4 Replace the default link name of the replacement NIC with the link name of the failed card.

By default, the link name of the `bge` card that replaces the failed `ce` card is `bge`*n*, where *n* is the instance number, such as `bge0`.

```
# dladm rename-link bge0 subitops0
```

This step transfers the network configuration of `subitops0` to `bge0`.

5 Attach the replacement NIC to the system.

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

For example, you use the `cfgadm` command to perform this step. For more information, see the `cfgadm(1M)` man page.

After this step, the new interface is configured with the test address, added as an underlying interface of the IPMP group, and deployed either as an active or a standby interface, all depending on the configurations that are specified in `/etc/hostname.subitops0`. The kernel can then allocate data addresses to this new interface according to the contents of the `/etc/hostname.`*ipmp-interface* configuration file.

About Missing Interfaces at System Boot

Certain systems might have the following configurations:

- An IPMP group is configured with underlying IP interfaces
- A `/etc/hostname.`*interface* file exists for one underlying IP interface.
- The physical hardware that is associated with the `/etc/hostname` file is missing.

With the new IPMP implementation where data addresses belong to the IPMP interface, recovering the missing interface becomes automatic. During system boot, the boot script constructs a list of failed interfaces, including interfaces that are missing. Based on the `/etc/hostname` file of the IPMP interface as well as the `hostname` files of the underlying IP interfaces, the boot script can determine to which IPMP group an interface belongs. When the missing interface is subsequently dynamically reconfigured on the system, the script then automatically adds that interface to the appropriate IPMP group and the interface becomes immediately available for use.

Monitoring IPMP Information

The following procedures use the `ipmpstat` command, enabling you to monitor different aspects of IPMP groups on the system. You can observe the status of the IPMP group as a whole or its underlying IP interfaces. You can also verify the configuration of data and test addresses for the group. Information about failure detection is also obtained by using the `ipmpstat` command. For more details about the `ipmpstat` command and its options, see the PLACEHOLDER IPMPSTAT MAN PAGE.

By default, host names are displayed on the output instead of the numeric IP addresses, provided that the host names exist. To list the numeric IP addresses in the output, use the `-n` option together with other options to display specific IPMP group information.

Note – In the following procedures, use of the `ipmpstat` command does not require system administrator privileges, unless stated otherwise.

How to Obtain IPMP Group Information

Use this procedure to list the status of the various IPMP groups on the system, including the status of their underlying interfaces. If probe-based failure detection is enabled for a specific group, the command also includes the failure detection time for that group.

- **Display the IPMP group information.**

```
$ ipmpstat -g
GROUP   GROUPNAME   STATE      FDT        INTERFACES
itops0  itops0      ok         10.00s     subitops0 subitops1
acctg1  acctg1      failed     --         [hme0 hme1]
field2  field2      degraded   20.00s     fops0 fops3 (fops2) [fops1]
```

`GROUP` Specifies the IPMP interface name. In the case of an anonymous group, this field will be empty. For more information about anonymous groups, see the `in.mpathd(1M)` man page.

`GROUPNAME` Specifies the name of the IPMP group. In the case of an anonymous group, this field will be empty.

`STATE` Indicates a group's current status, which can be one of the following:

- `ok` indicates that all underlying interfaces of the IPMP group are usable.
- `degraded` indicates that some of the underlying interfaces in the group are unusable.
- `failed` indicates that all of the group's interfaces are unusable.

`FDT` Specifies the failure detection time, if failure detection is enabled. If failure detection is disabled, this field will be empty.

`INTERFACES` Specifies the underlying interfaces that belong to the group. In this field, active interfaces are listed first, then inactive interfaces, and finally unusable interfaces.The status of the interface is indicated by the manner in which it is listed:

- *interface* (without parentheses or brackets) indicates an active interface. Active interfaces are those interfaces that being used by the system to send or receive data traffic.
- *(interface)* (with parentheses) indicates a functioning but inactive interface. The interface is not in use as defined by administrative policy.
- *[interface]* (with brackets) indicates that the interface is unusable because it has either failed or been taken offline.

How to Obtain IPMP Data Address Information

Use this procedure to display data addresses and the group to which each address belongs. The displayed information also includes which address is available for use, depending on whether the address has been toggled by the `ifconfig [up/down]` command. You can also determine on which inbound or outbound interface an address can be used.

- **Display the IPMP address information.**

```
$ ipmpstat -an
ADDRESS         STATE    GROUP      INBOUND     OUTBOUND
192.168.10.10   up       itops0     subitops0   subitops0 subitops1
192.168.10.15   up       itops0     subitops1   subitops0 subitops1
192.0.0.100     up       acctg1     --          --
192.0.0.101     up       acctg1     --          --
128.0.0.100     up       field2     fops0       fops0 fops3
128.0.0.101     up       field2     fops3       fops0 fops3
128.0.0.102     down     field2     --          --
```

`ADDRESS` Specifies the hostname or the data address, if the `-n` option is used in conjunction with the `-a` option.

`STATE` Indicates whether the address on the IPMP interface is `up`, and therefore usable, or `down`, and therefore unusable.

`GROUP` Specifies the IPMP IP interface that hosts a specific data address.

`INBOUND` Identifies the interface that receives packets for a given address. The field information might change depending on external events. For example, if a data address is down, or if no active IP interfaces remain in the IPMP group, this field will be empty. The empty field indicates that the system is not accepting IP packets that are destined for the given address.

`OUTBOUND` Identifies the interface that sends packets that are using a given address as a source address. As with the `INBOUND` field, the `OUTBOUND` field information might also change depending on external events. An empty field indicates that the system is not sending out packets with the given source address. The field might be empty either because the address is down, or because no active IP interfaces remain in the group.

How to Obtain Information About Underlying IP Interfaces of a Group

Use this procedure to display information about an IPMP group's underlying IP interfaces. For a description of the corresponding relationship between the NIC, data link, and IP interface, see "Overview of the Networking Stack" on page 13.

- **Display the IPMP interface information.**

```
$ ipmpstat -i
INTERFACE   ACTIVE   GROUP      FLAGS      LINK       PROBE      STATE
subitops0   yes      itops0     --mb---    up         ok         ok
subitops1   yes      itops0     -------    up         disabled   ok
hme0        no       acctg1     -------    unknown    disabled   offline
hme1        no       acctg1     is-----    down       unknown    failed
fops0       yes      field2     --mb---    unknown    ok         ok
fops1       no       field2     -i-----    up         ok         ok
fops2       no       filed2     -------    up         failed     failed
fops3       yes      field2     --mb---    up         ok         ok
```

`INTERFACE` Specifies each underlying interface of each IPMP group.

`ACTIVE` Indicates whether the interface is functioning and is in use (`yes`) or not (`no`).

`GROUP` Specifies the IPMP interface name. In the case of anonymous groups, this field will be empty. For more information about anonymous groups, see the `in.mpathd(1M)` man page.

`FLAGS` Indicates the status of the underlying interface, which can be one or any combination of the following:

- `i` indicates that the `INACTIVE` flag is set for the interface and therefore the interface is not used to send or receive data traffic.
- `s` indicates that the interface is configured to be a standby interface.
- `m` indicates that the interface is designated by the system to send and receive IPv4 multicast traffic for the IPMP group.
- `b` indicates that the interface is designated by the system to receive broadcast traffic for the IPMP group.
- `M` indicates that the interface is designated by the system to send and receive IPv6 multicast traffic for the IPMP group.
- `d` indicates that the interface is down and therefore unusable.
- `h` indicates that the interface shares a duplicate physical hardware address with another interface and has been taken offline. The `h` flag indicates that the interface is unusable.

`LINK` Indicates the state of link-based failure detection, which is one of the following states:

- `up` or `down` indicates the availability or unavailability of a link.
- `unknown` indicates that the driver does not support notification of whether a link is `up` or `down` and therefore does not detect link state changes.

`PROBE` Specifies the state of the probe–based failure detection for interfaces that have been configured with a test address, as follows:

- `ok` indicates that the probe is functional and active.
- `failed` indicates that probe-based failure detection has detected that the interface is not working.
- `unknown` indicates that no suitable probe targets could be found, and therefore probes cannot be sent.
- `disabled` indicates that no IPMP test address is configured on the interface. Therefore probe-based failure detection is disabled.

`STATE` Specifies the overall state of the interface, as follows:

- `ok` indicates that the interface is online and working normally based on the configuration of failure detection methods.
- `failed` indicates that the interface is not working because either the interface's link is down, or the probe detection has determined that the interface cannot send or receive traffic.
- `offline` indicates that the interface is not available for use. Typically, the interface is switched offline under the following circumstances:
  - The interface is being tested.
  - Dynamic reconfiguration is being performed.
  - The interface shares a duplicate hardware address with another interface.
- `unknown` indicates the IPMP interface's status cannot be determined because no probe targets can be found for probe-based failure detection.

How to Obtain IPMP Probe Target Information

Use this procedure to monitor the probe targets that are associated with each IP interface in an IPMP group.

- **Display the IPMP probe targets.**

```
$ ipmpstat -nt
INTERFACE   MODE          TESTADDR        TARGETS
subitops0   routes        192.168.85.30   192.168.85.1 192.168.85.3
subitops1   disabled      --              --
hme0        disabled      --              --
hme1        routes        192.1.2.200     192.1.2.1
fops0       multicast     128.9.0.200     128.0.0.1 128.0.0.2
fops1       multicast     128.9.0.201     128.0.0.2 128.0.0.1
fops2       multicast     128.9.0.202     128.0.0.1 128.0.0.2
fops3       multicast     128.9.0.203     128.0.0.1 128.0.0.2
```

`INTERFACE` Specifies the underlying interfaces of the IPMP group.

`MODE` Specifies the method for obtaining the probe targets.

- `routes` indicates that the system routing table is used to find probe targets.
- `mcast` indicates that multicast ICMP probes are used to find targets.
- `disabled` indicates that probe-based failure detection has been disabled for the interface.

`TESTADDR` Specifies the hostname or, if the `-n` option is used in conjunction with the `-t` option, the IP address that is assigned to the interface to send and receive probes. This field will be empty if a test address has not been configured.

Note – If an IP interface is configured with both IPv4 and IPv6 test addresses, the probe target information is displayed separately for each test address.

`TARGETS` Lists the current probe targets in a space-separated list. The probe targets are displayed either as hostnames or IP addresses, if the `-n` is used in conjunction with the `-t` option.

How to Observe IPMP Probes

Use this procedure to observe ongoing probes. When you issue the command to observe probes, information about probe activity on the system is continuously displayed until you terminate the command with `Ctrl-C`. You must have Primary Administrator privileges to run this command.

1 Assume the role of Primary Administrator, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 Display the information about ongoing probes.

```
# ipmpstat -pn
TIME    INTERFACE   PROBE   TARGET        NETRTT   RTT      RTTAVG     RTTDEV
0.11s   subitops0   589     192.168.85.1  0.51ms   0.76ms   0.76ms     --
0.17s   hme1        612     192.1.2.1     --       --       --         --
0.25s   fops0       602     128.0.0.1     0.61ms   1.10ms   1.10ms     --
0.26s   fops1       602     128.0.0.2     --       --       --         --
0.25s   fops2       601     128.0.0.1     0.62ms   1.20ms   1.00ms     --
0.26s   fops3       603     128.0.0.1     0.79ms   1.11ms   1.10ms     --
1.66s   hme1        613     192.1.2.1     --       --       --         --
1.70s   subitops0   603     192.168.85.3  0.63ms   1.10ms   1.10ms     --
^C
```

TIME	Specifies the time a probe was sent relative to when the `ipmpstat` command was issued. If a probe was initiated prior to `ipmpstat` being started, then the time is displayed with a negative value, relative to when the command was issued.
PROBE	Specifies the identifier that represents the probe.
INTERFACE	Specifies the interface on which the probe is sent.
TARGET	Specifies the hostname or, if the `-n` option is used in conjunction with `-p`, the target address to which the probe is sent.
NETRTT	Specifies the total network round-trip time of the probe and is measured in milliseconds. `NETRTT` covers the time between the moment when the IP module sends the probe and the moment the IP module receives the `ack` packets from the target. If the `in.mpathd` daemon has determined that the probe is lost, then the field will be empty.
RTT	Specifies the total round-trip time for the probe and is measured in milliseconds. `RTT` covers the time between the moment the daemon executes the code to send the probe and the moment the daemon completes processing the `ack` packets from the target. If the `in.mpathd` daemon has determined that the probe is lost, then the field will be empty. Spikes that occur in the `RTT` which are not present in the `NETRTT` might indicate that the local system is overloaded.
RTTAVG	Specifies the probe's average round-trip time over the interface between local system and target. The average round-trip time helps identify slow targets. If data is insufficient to calculate the average, this field will be empty.
RTTDEV	Specifies the standard deviation for the round-trip time to the target over the interface. The standard deviation helps identify jittery targets whose `ack` packets are being sent erratically. For jittery targets, the `in.mpathd` daemon is forced to increase the failure detection time. Consequently, the daemon would take a longer time before it can detect such a target's outage. If data is insufficient to calculate the standard deviation, this field will be empty.

▼ How to Customize the Output of the `ipmpstat` Command in a Script

When you use the `ipmpstat`, by default, the most meaningful fields that fit in 80 columns are displayed. In the output, all the fields that are specific to the option that you use with the `ipmpstat` command are displayed, except in the case of the `ipmpstat -p` syntax. If you want to specify the fields to be displayed, then you use the `-o` option in conjunction with other options that determine the output mode of the command. This option is particularly useful when you issue the command from a script or by using a command alias

- **To customize the output, issue one of the following commands:**
  - To display selected fields of the `ipmpstat` command, use the `-o` option in combination with the specific output option. For example, to display only the `GROUPNAME` and the `STATE` fields of the group output mode, you would type the following:

    ```
    $ ipmpstat -g -o groupname,state

    GROUPNAME  STATE
    itops0     ok
    accgt1     failed
    field2     degraded
    ```

  - To display all the fields of a given `ipmpstat` command, use the following syntax:

    ```
    # ipmpstat -o all
    ```

How to Generate Machine Parseable Output of the `ipmpstat` Command

You can generate machine parseable information by using the `ipmpstat -P` syntax. The `-P` option is intended to be used particularly in scripts. Machine-parseable output differs from the normal output in the following ways:

- Headers are omitted.
- Fields are separated by colons (`:`).
- Fields with empty values are empty rather than being filled with the double dash (`--`).
- In the case of multiple fields being requested, if a field contains a literal colon (`:`) or back slash (`\`), these can be escaped or excluded by prefixing these characters with a back slash (`\`) .

To correctly use the `ipmpstat -P` syntax, observe the following rules:

- Use the `-o` *option fields* together with the `-P` option.
- Never use `-o all` with the `-P` option.

Ignoring either one of these rules will cause `ipmpstat -P` to fail.

- **To display in machine parseable format the group name, the failure detection time, and the underlying interfaces, you would type the following:**

```
$ ipmpstat -P -o -g groupname,fdt,interfaces
itops0:10.00s:subitops0 subitops1
acctg1::[hme0 hme1]
field2:20.00s:fops0 fops3 (fops2) [fops1]
```

The group name, failure detection time, and underlying interfaces are group information fields. Thus, you use the `-o -g` options together with the `-P` option.

Example 8–8 Using `ipmpstat -P` in a Script

This sample script displays the failure detection time of a particular IPMP group.

```
getfdt() {
         ipmpstat -gP -o group,fdt | while IFS=: read group fdt; do
             [[ "$group" = "$1" ]] && { echo "$fdt"; return; }
         done
     }
```

PART III

Network Virtualization and Resource Management

◆ ◆ ◆ CHAPTER 9

Introducing Network Virtualization and Resource Control (Overview)

This chapter explains the basic concepts involved in network virtualization and resource control. The following topics are covered:

- Network virtualization
- Types of virtual networks
- Virtual machines and zones
- Resource control, including flow management
- Enhanced network observability

These features help you to manage flow control, improve system performance, and configure the network utilization needed to achieve OS virtualization, utility computing, and server consolidation.

For specific tasks, refer to the following chapters:

- Chapter 11, "Configuring Virtual Networks (Tasks)"
- Chapter 13, "Configuring Resource Management on an Interface"
- Chapter 12, "Administering Virtual Networks and Resource Controls (Tasks)"

Network Virtualization and Virtual Networks

Network virtualization is the process of combining hardware network resources and software network resources into a single administrative unit. The goal of network virtualization is to provide systems and users with efficient, controlled, and secure sharing of the networking resources.

The end product of network virtualization is the *virtual network*. Virtual networks are classified into two broad types, external and internal. *External virtual networks* consist of several local networks that are administered by software as a single entity. The building blocks of classic external virtual networks are switch hardware and VLAN software technology. Examples of external virtual networks include large corporate networks and data centers.

An *internal virtual network* consists of one system using virtual machines or zones that are configured over at least one pseudo-network interface. These containers can communicate with each other as though on the same local network, providing a virtual network on a single host. The building blocks of the virtual network are *virtual network interface cards or virtual NICs (VNICs)* and virtual switches. Solaris network virtualization provides the internal virtual network solution.

You can combine networking resources to configure both internal and external virtual networks. For example, you can configure individual systems with internal virtual networks onto LANs that are part of a large, external virtual network. The network configurations that are described in this part include examples of combined internal and external virtual networks.

Types of Containers for Network Virtualization on the Solaris OS

You can use several different types of virtual containers in a Solaris OS-based virtual network. These containers include machines and zones. A *virtual machine* is a container with its own kernel and IP protocol stack. A zone is a container that provides an isolated environment for running applications.

Sun xVM Virtual Machines

Sun™ xVM is virtual machine technology that enables you to create multiple instances of an operating system on the interfaces of a single x86–based system. The Sun xVM hypervisor controls the allocation and operation of the domains. For more information on xVM, refer to Introduction to the Sun xVM Hypervisor. xVM is based on the Open Source XEN hypervisor, which is described on the xen.org website.

Non-Global Zones and Exclusive IP Zones

Though not true virtual machines, zones are light weight application environments that share a host's kernel and IP stack. You can configure exclusive IP instances for a non-global zone, which provides that zone with its own, exclusive TCP/IP protocol stack. Both standard non-global zones and exclusive IP zones can be configured on a Solaris-based virtual network. For basic information about zones, refer to Chapter 16, "Introduction to Solaris Zones," in *System Administration Guide: Virtualization Using the Solaris Operating System*.

LDOMs Virtual Machines

The Libvert for LDOMs (Logical Domains) software provides a hypervisor and set of commands that enable you to set up and administer logical domains on a Solaris OS-based virtual network. Each logical domain can run an instance of an operating system to enable multiple operating systems on the same computer. For information on LDOMs, refer to the *Logical Domains (LDoms) 1.0.1 Administration Guide*.

Parts of the Internal Virtual Network

An internal virtual network built on the Solaris OS contains the following parts:

- At least one network interface card, or NIC.
- A virtual NIC, or VNIC, which is configured on top of the network interface
- A virtual switch, which is configured at the same time as the first VNIC on the interface.
- A container, such as a zone or virtual machine , which is configured on top of the VNIC.

The next figure shows these parts and how they fit together on a single system.

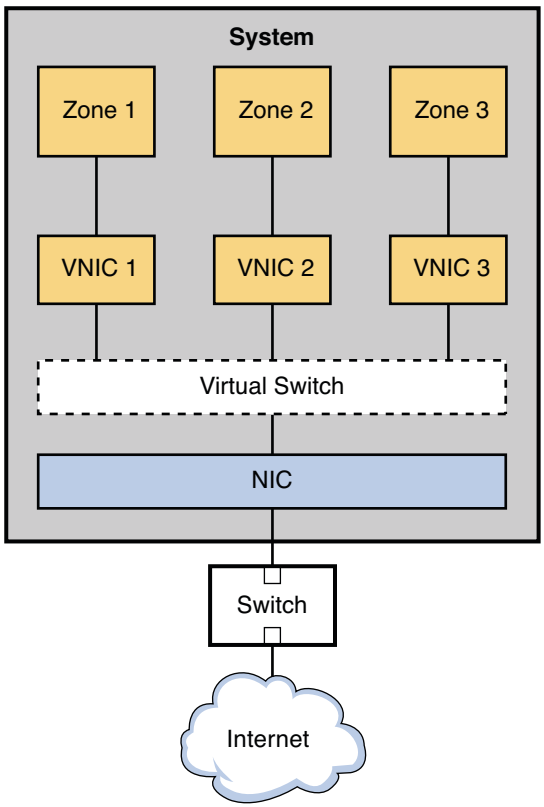

FIGURE 9–1 VNIC Configuration for a Single Interface

The figure shows a single system with one NIC. The NIC is configured with three VNICs. Each VNIC supports a single zone. Therefore, Zone 1, Zone 2, and Zone 3 are configured over VNIC 1, VNIC 2, and VNIC 3, respectfully. The three VNICs are virtually connected to one virtual

switch. This switch provides the connection between the VNICs and the physical NIC upon which the VNICs are built. The physical interface provides the system with its external network connection.

Alternatively, you can create a virtual network based on the etherstub. Etherstubs are purely software and do not require a network interface as the basis for the virtual network.

A *VNIC* is a virtual network device with the same data-link interface as a physical interface. You configure VNICs on top of a physical interface. For the current list of physical interfaces that support VNICs, refer to the Network Virtualization and Resource Control FAQ (`http://www.opensolaris.org/os/project/crossbow/faq/`). You can configure up to 900 VNICs on a single physical interface. When VNICs are configured, they behave like physical NICs. In addition, the system's resources treat VNICs as if they were physical NICs.

Each VNIC is implicitly connected to a *virtual switch* that corresponds to the physical interface. The virtual switch provides the same connectivity between VNICs on a virtual network that switch hardware provides for the systems connected to a switch's ports.

In accordance with Ethernet design, if a switch port receives an outgoing packet from the host connected to that port, that packet cannot go to a destination on the same port. This design is a drawback for systems that are configured with zones or virtual machines. Without network virtualization, outgoing packets from a virtual machine or a zone with an exclusive stack cannot be passed to another virtual machine or zone on the same system. The outgoing packets go through a switch port out onto the external network. The incoming packets cannot reach their destination zone or virtual machine because the packets cannot return through the same port as they were sent. Therefore, when virtual machines and zones on the same system need to communicate, a data path between the containers must open on the local machine. Virtual switches provide these containers with the method to pass packets.

How Data Travels Through a Virtual Network

Figure 9–1 illustrates a simple VNIC configuration for a virtual network on a single system.

When the virtual network is configured, a zone sends traffic to an external host in the same fashion as a system without a virtual network. Traffic flows from the zone, through the VNIC to the virtual switch, and then to the physical interface, which sends the data out onto the network.

But what happens if one zone on a virtual network wants to send packets to another zone on the virtual network, given the previously mentioned Ethernet restrictions? As shown in Figure 9–1, suppose Zone 1 needs to send traffic to Zone 3? In this case packets pass from Zone 1 through its dedicated VNIC 1. The traffic then flows through the virtual switch to VNIC 3. VNIC 3 then passes the traffic to Zone 3. The traffic never leaves the system, and therefore never violates the Ethernet restrictions.

Who Should Implement Virtual Networks?

If you need to consolidate resources on Sun servers, consider implementing VNICs and virtual networks. Consolidators at ISPs, telecommunications companies, and large financial institutions can use the following network virtualization features to improve the performance of their servers and networks.

- NIC hardware, including the powerful new interfaces that support hardware rings
- Multiple MAC addresses for the VNICs
- The large amount of bandwidth provided by newer interfaces

You can replace many systems with a single system that implements running multiple zones or virtual machines, without significantly losing separation, security, and flexibility.

What Is Resource Control?

Resource control is the process of allocating a system's resources in a controlled fashion. The Solaris OS resource control features enable bandwidth to be shared among the VNICs on a system's virtual network. You can also use resource control features to allocate and manage bandwidth on a physical interface without VNICs and virtual machines. This section introduces the major features of resource control and briefly explains how these features work.

How Bandwidth Management and Flow Control Works

Searchnetworking.com (`http://searchnetworking.techtarget.com`) defines bandwidth as "the amount of data that can be carried from one point to another in a given time period (usually a second)." *Bandwidth management* enables you to assign a portion of the available bandwidth of a physical NIC to a consumer, such as an application or customer. You can control bandwidth on a per- application, per-port, per-protocol, and per-address basis. Bandwidth management assures efficient use of the large amount of bandwidth available from the new GLDv3 network interfaces.

Resource control features enable you implement a series of controls on an interface's available bandwidth. For example, you can set a *guarantee* of an interface's bandwidth to a particular consumer. That guarantee is the minimum amount of assured bandwidth allocated to the application or enterprise The allocated portion of bandwidth is known as a *share*. By setting up guarantees, you can allocate enough bandwidth for applications that cannot function properly without a certain amount of bandwidth. For example, streaming media and Voice over IP consume a great deal of bandwidth. You can use the resource control features to guarantee that these two applications have enough bandwidth to successfully run.

You can also set a *limit* on the share. The limit is the maximum allocation of bandwidth the share can consume. Using limits, you can contain non-critical services from taking away bandwidth from critical services.

Finally, you can prioritize among the various shares allotted to consumers. You can give highest priority to critical traffic, such as heartbeat packets for a cluster, and lower priority for less critical applications.

For example, application service providers (ASPs) can offer customers fee-based levels of service that are based on the bandwidth share that the customer purchases. As part of the service level agreement (SLA), each share is then guaranteed an amount of bandwidth, to not exceed the purchased limit. (For more information on service level agreements, see "Implementing Service-Level Agreements" in *System Administration Guide: IP Services*. Priority controls might be based on different tiers of the SLA, or different prices paid by the SLA customer.

Bandwidth usage is controlled through management of flows. A *flow* is a stream of packets that all have certain characteristics, such as the port number or destination address. These flows are managed by transport, service, or virtual machine, including zones. Flows cannot exceed the amount of bandwidth that is guaranteed to the application or to the customer's purchased share.

When a VNIC or flow is assigned a guarantee, the VNIC is assured its designated bandwidth even if other flows or VNICs also use the interface. However, assigned guarantees are workable only if they do not exceed the maximum bandwidth of the physical interface.

Allocating Resource Control and Bandwidth Management on a Network

The following figure shows a corporate network topology that uses resource control to manage various applications.

Network With Resource Controls in Place

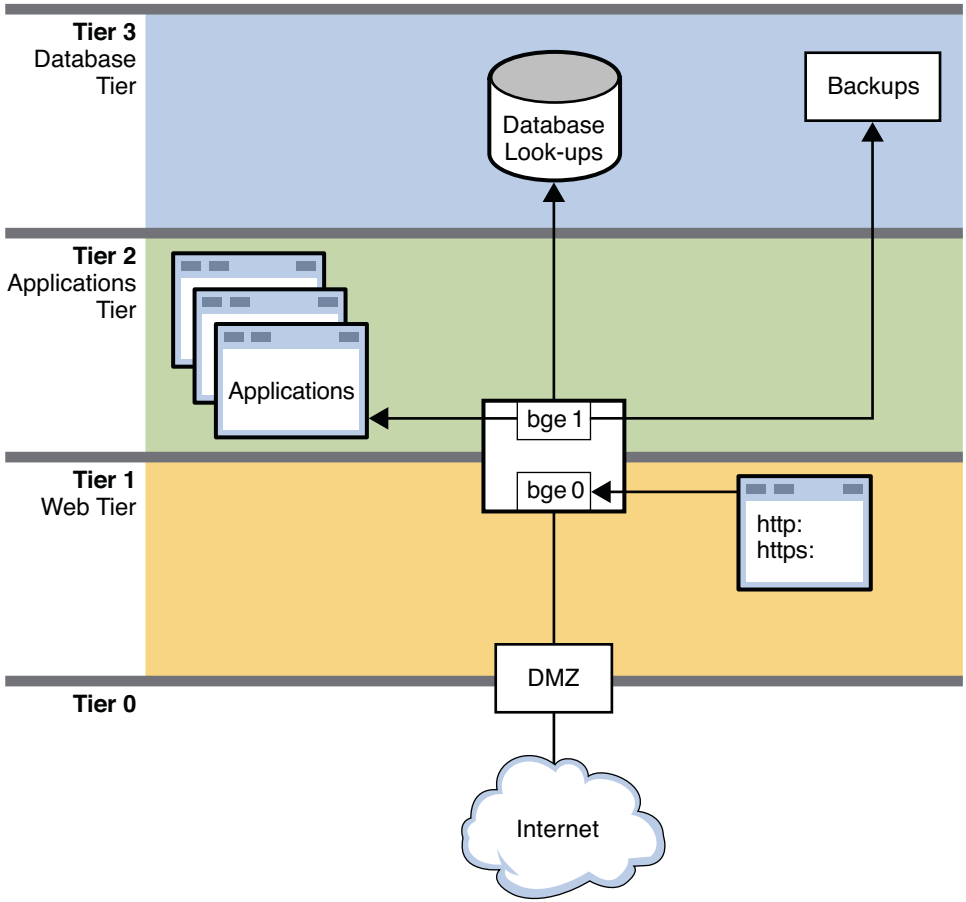

FIGURE 9–2 Network With Resource Controls in Place

This figure shows a typical network topology that uses resource controls to improve network efficiency and performance. The network does not implement VNICs and containers, such as exclusive zones and virtual machines. However, VNICs and containers could be used on this network for consolidation and other purposes.

The network is divided into four tiers:

- **Tier 0** is the demilitarized zone (DMZ). This is a small local network that controls access to and from the outside world. Resource control is not used on the systems of the DMZ.
- **Tier 1** is the web tier and includes two systems. The first system is a proxy server that does filtering. This server has two interfaces, `bge0` and `bge1`. The `bge0` link connects the proxy server to the DMZ on Tier 0. The `bge0` link also connects the proxy server to the second system, the web server. The `http` and `https` services share the bandwidth of the web server

with other standard applications. Due to the size and critical nature of web servers, shares of `http` and `https` require guarantees and prioritization.

- **Tier 2** is the applications tier and also includes two systems. The second interface of the proxy server, `bge1`, provides the connection between the web tier and the applications tier. Through a switch, an applications server connects to `bge1` on the proxy server. The applications server requires resource control to manage the shares of bandwidth given to the various applications that are run. Critical applications that need a lot of bandwidth must be given higher guarantees and priorities than smaller, or less critical applications.
- **Tier 3** is the database tier. The two systems on this tier connect through a switch to the proxy server's `bge1` interface. The first system, a database server, needs to issue guarantees and to prioritize the various processes involved in database lookups. The second system is a backup server for the network. This system must consume a great deal of bandwidth during backups. However, backup activities are typically carried out overnight. Using resource controls, you can control when the backup processes have the highest bandwidth guarantees and highest priorities.

Who Should Implement Resource Control Features

Any system administrator who wants to improve a system's efficiency and performance should consider implementing the resource control features. Consolidators can delegate bandwidth shares in combination with VNICs to help balance the load of large servers. Server administrators can use share allocation features to implement SLA's, such as those offered by ASPs. Traditional system administrators can use the bandwidth management features to isolate and prioritize certain applications. Finally, share allocation makes it easy for you to observe bandwidth usage by individual consumers.

Observability Features for Network Virtualization and Resource Control

Network virtualization and resource control includes observability features to help you view resource usage before setting up controls such as VNICs and flows. In tandem with Solaris extended accounting, the resource control observability features allow you to accumulate systems statistics into logs. The observability features of network virtualization and resource control include:

- Ability to monitor a running system.
- Ability to log and report statistics.
- Extended accounting features to log historical data

The new `flowadm` command and extensions to the `dladm` and `netstat` commands implement the network virtualization observability features. You can use these commands to monitor current system usage and to gather statistical data into logs.

By analyzing the historical logs, you can determine the following:

- Where network resources can be consolidated from many systems to a single system, possibly with greater bandwidth through the new generation of network interfaces. Do this prior to setting up VNICs and virtual machines or exclusive zones.
- Which applications consume the most bandwidth. This information can help you to set up bandwidth management, so that critical applications are guaranteed the most bandwidth within a particular time slot. For example, you can guarantee a video stream the greatest amount of an interface's bandwidth for 20 hours a day. For a designated four hours a day, you can give highest priority to the system's backup program. Do this as part of bandwidth management implementation.
- How to much bill customers for bandwidth used. Application service providers and other businesses that rent out system space can use the Resource control observability features to determine usage by paying customers. Some businesses offer customers Service Level Agreements, wherein the customer buys a guaranteed percentage of bandwidth from the provider. The observability features let you view how much bandwidth each customer uses and bill for possible overages. Other businesses offer customers bandwidth on a per use basis. Here the observability features directly help in billing. Do this after you have implemented resource control and, possibly, VNICs and virtual machines on a system.

The next chapter, Chapter 10, “Planning for Network Virtualization and Resource Control,” contains scenarios that show where the observability features are used for planning consolidation and resource control.

◆ ◆ ◆ CHAPTER 10

Planning for Network Virtualization and Resource Control

This chapter contains information and example scenarios to help you evaluate and then design network virtualization and resource control solutions for your site. The chapter discusses the following scenarios:

- "Basic Virtual Network on a Single System" on page 166
- "Private Virtual Network on a Single System" on page 168
- "Interface-based Resource Control for a Traditional Network" on page 171

Each scenario contains "best usage" suggestions that explain the types of networks that best benefit from the particular scenario.

Network Virtualization and Resource Control Task Map

Task	Description	For Instructions
Design and plan a virtual network on a single host	Consolidate network services and applications offered by the local network onto a single host. This scenario is especially useful for consolidators and service providers.	"Planning and Designing a Virtual Network" on page 166
Design and plan for a private virtual network on a single host	Run a virtual network that does not allow public access. This scenario is recommended for system administrators who need to run a development environment.	"Private Virtual Network on a Single System" on page 168

Task	Description	For Instructions
Provide bandwidth management and resource control for systems on a per-interface basis.	Isolate, prioritize, and assign a specific amount of interface bandwidth for packet traffic. This scenario is useful for systems that handle heavy traffic for particular services, such as a web service or a database server.	"Interface-based Resource Control for a Traditional Network" on page 171
Provide bandwidth management and resource control for a virtual network	Create a scenario to isolate, control, and prioritize traffic for particular applications on the individual containers of the virtual network. This scenario is particularly useful for service providers who bill customers for usage of particular application or who rent out containers to customers.	Information to come

Planning and Designing a Virtual Network

This section describes two different scenarios for configuring a virtual network. Look over the scenarios to help determine which most closely fits the needs of your site. Then use that scenario as the basis for designing your specific virtualization solution. The scenarios include:

- Basic virtual network of two zones, especially useful for consolidating network services from the local network onto a single host.
- Private virtual network, useful for a development environment where you isolate applications and services from the public network.

Basic Virtual Network on a Single System

Figure 10–1 shows the basic virtual network, or "network in a box" that is used in examples throughout the section "Configuring a Basic Virtual Network" on page 176.

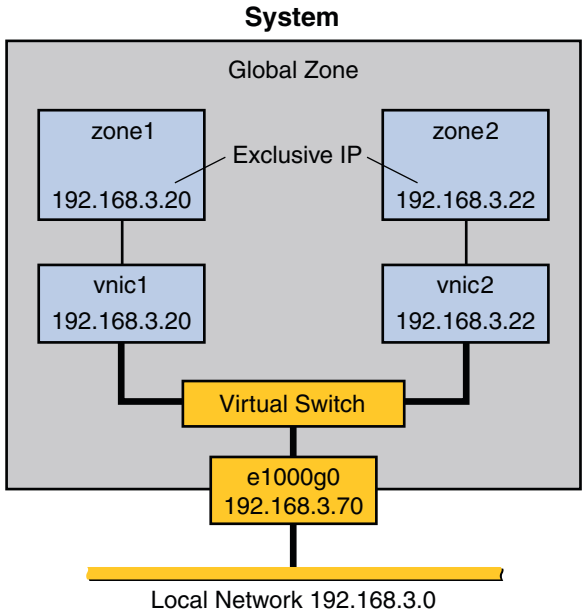

FIGURE 10–1 Virtual Network on a Single Host

This virtual network consists of the following:

- A single GLDv3 network interface `e1000g0`. This interface connects to the public network `192.168.3.0/24`. Interface `e1000g0` has the IP address `192.168.3.70`.
- A virtual switch, which is automatically configured when you create the first VNIC.
- Two VNICs. `vnic1` has the IP address `192.168.3.20`, and `vnic2` has the IP address `192.168.3.22`.
- Two exclusive IP zones. `zone1` is configured over `vnic1`, and `zone2` is configured over `vnic2`.

Note – Though the example uses exclusive IP zones, you can also use other types of virtual machines such as LDOMS in this scenario.

The VNICs and zones in this configuration allow access to the public. Therefore, the zones can pass traffic beyond the `e1000g0` interface. Likewise, users on external networks can reach applications and services offered by the zones.

Best Uses for the Basic Virtual Network

The network in a box scenario enables you to isolate processes and applications into individual virtual machines or zones on a single host. Furthermore, this scenario is expandable to include

many containers, each of which could run a completely isolated set of applications. The scenario improves a system's efficiency and, by extension, the efficiency of the local network. Therefore, this scenario is ideal for the following users:

- Network consolidators and others who want to consolidate the services of a LAN onto a single system.
- Any site that rents out services to customers. You can rent out individual zones or virtual machines, observe traffic, and take statistics for performance measuring or for billing purposes on each zone in the virtual network.
- Any administrator who wants to isolate processes and applications to separate containers to improve system efficiency .

For More Information

- For tasks that implement the basic virtual network, go to "Configuring a Basic Virtual Network" on page 176.
- For examples that show how to configure the virtual network shown in this section, go to "Configuring a Basic Virtual Network" on page 176.
- For conceptual information about VNICs and virtual networks, go to "Network Virtualization and Virtual Networks" on page 155.
- For conceptual information about zones, go to Chapter 16, "Introduction to Solaris Zones," in *System Administration Guide: Virtualization Using the Solaris Operating System*.

Private Virtual Network on a Single System

Figure 10–2 shows a single system with a private network behind packet filtering software that performs network address translation (NAT). This figure illustrates the scenario that is built in Example 11–7.

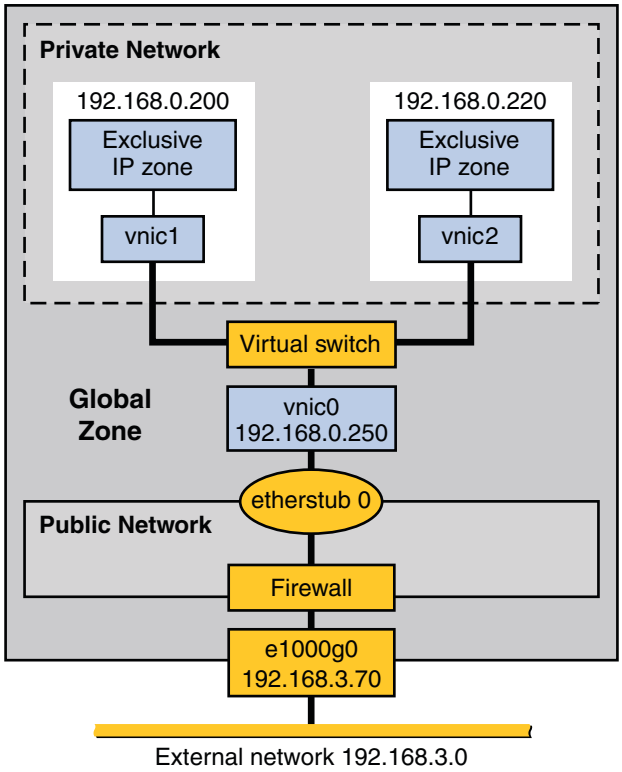

FIGURE 10–2 Private Virtual Network on a Single Host

The topology features a single system with a public network, including a firewall, and a private network built on an etherstub pseudo-interface. The public network runs in the global zone and consists of the following elements:

- GLDv3 network interface `e1000g0` with the IP address `192.168.3.70`.
- A firewall implemented in the IP Filter software. For an introduction to IP Filter, refer to "Introduction to Solaris IP Filter" in *System Administration Guide: IP Services*.
- `etherstub0`, a pseudo-interface upon which the virtual network topology is built. *Etherstubs* provide the ability to create a virtual network on a host. That network is totally isolated from the external network.

The private network consists of the following elements:

- A virtual switch which provides packet forwarding among the VNICs of the private network.
- `vnic0`, which is the VNIC for the global zone, and has the IP address `192.168.0.250`.

- `vnic1` with the IP address `192.168.0.200` and `vnic2` with the IP address `192.168.0.220`. All three VNICs are configured over `etherstub0`.
- `zone1`, which is configured over `vnic1`, and `zone2`, which is configured over `vnic2`.

Best Uses for a Private Virtual Network

Consider creating a private virtual network for a host that is used in a development environment. Using the etherstub framework, you can totally isolate software or features under development to the containers of the private network. Moreover, you can use firewalling software for network address translation of outgoing packets that originate in the containers of the private network. The private network is a smaller version of the eventual deployment environment.

For More Information

- For tasks for implementing the private virtual network, go to "Configuring a Private Virtual Network" on page 202
- For the example that shows how to configure the private virtual network shown in this section, go toExample 11–7.
- For conceptual information about VNICs and virtual networks, go to "Network Virtualization and Virtual Networks" on page 155.
- For conceptual information about zones, go to Chapter 16, "Introduction to Solaris Zones," in *System Administration Guide: Virtualization Using the Solaris Operating System*.
- For information about Solaris IP Filter, go to "Introduction to Solaris IP Filter" in *System Administration Guide: IP Services*.

Network Interface-Based Flow Control

This section contains scenarios for using interface-based flow control. Flow control incorporates resource control and bandwidth management. This process involves organizing packet traffic into flows of related applications, and then assigning portions of an interface's overall bandwidth to the individual flows. You can also assign a specific application's traffic to a flow to isolate the traffic without assigning the flow an amount of bandwidth. Isolating particular types of traffic to flows makes the traffic more easily observed, evaluated, and possibly billed. The following scenarios are described:

- "Interface-based Resource Control for a Traditional Network" on page 171
- "Flow Control for the Virtual Network" on page 172

Look over these scenarios to see which is most applicable for your site. For the tasks that implementing these scenarios, refer to Chapter 13, "Configuring Resource Management on an Interface."

Interface-based Resource Control for a Traditional Network

Figure 10–3 shows the network topology for a small business that needs to manage the bandwidth on its proxy server. The proxy server offers a public web site as well as a proxy for internal clients that require services from various servers on the site's internal network.

Note – This scenario does not show how to configure flow control for a virtual network, and consequentially does not include VNICs. For flow control on a virtual network, refer to Flow Control for a Virtual Network.

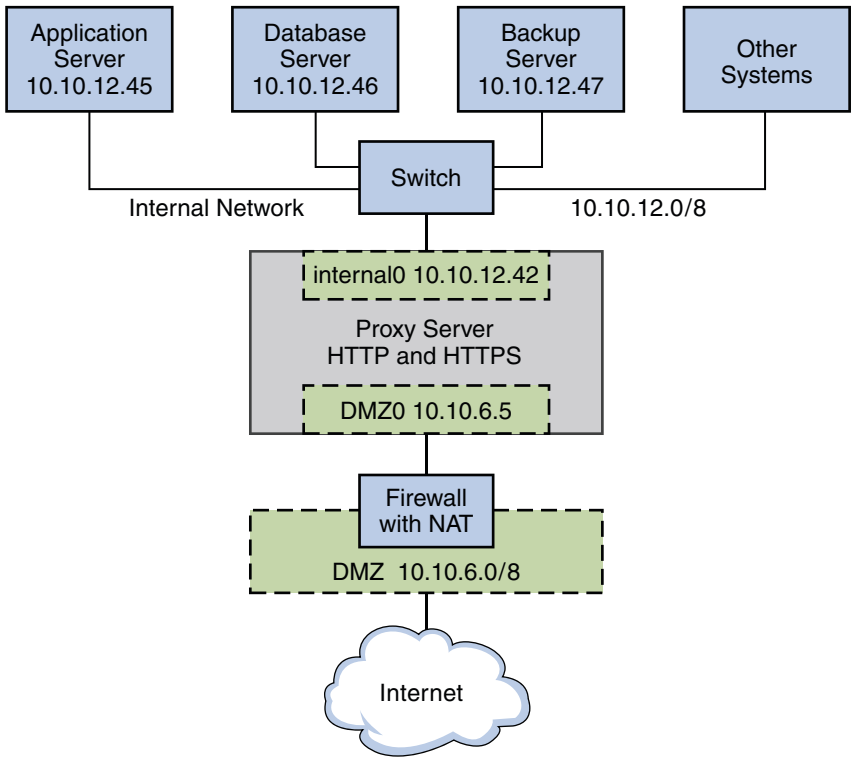

FIGURE 10–3 Resource Control for a Proxy Server on a Traditional Network

The figure shows that the company has a public network, `10.10.6.0/8`, that also serves as a demilitarized zone (DMZ). A system on the DMZ provides name-to-address translation (NAT) through an IP Filter firewall. The company has a large system that functions as the proxy server.

The system has two wired interfaces and 16 processor sets with IDs 0–16. This system is connected to the public network through the interface `nge0`, with IP address `l0 10.6.5`. The link name for the interface is `DMZ0`. Through `DMZ0`, the proxy server offers HTTP and HTTPS service through the company's public web site.

The figure also illustrates the company's internal network, `10.10.12.0/24`. The proxy server connects to the internal `10.10.12.0/8` network through interface `nge1`, with the IP address `10.10.12.42`. The link name for this interface is `internal0`. Through the `internal0` data link, the proxy server operates on behalf of internal clients that request the services of an application server, `10.10.12.45`, database server, `10.10.12.46`, and backup server, `10.10.12.47`.

Best Use of Interface–based Resource Control on a Traditional Network

Consider establishing flow control for heavily used systems, especially those with newer GLD.v3 interfaces with large amounts of available bandwidth. Interface-based flow control improves the efficiency of the interface, the system, and potentially the network. You can apply flow control to any system on any type of network. Furthermore, if your goal is to improve network efficiency, you can separate various services into individual flows. This action assigns separate hardware and software resources to the individual flows, thus isolating them from other services on a particular system. After you establish flows, you can observe traffic for each flow and gather statistics. Thereafter, you can assign bandwidth amount and priorities to control usage on the interfaces.

For More Information

- For tasks for implementing flow control, refer to Chapter 13, “Configuring Resource Management on an Interface”
- For conceptual information about bandwidth management and resource control, refer to “What Is Resource Control?” on page 159
- For detailed technical information, refer to the `dladm`(1M) and `flowadm`(1M)man pages.

Flow Control for the Virtual Network

This scenario shows how flow control is used within a virtual network, such as the basic virtual network that is introduced in “Basic Virtual Network on a Single System” on page 166.

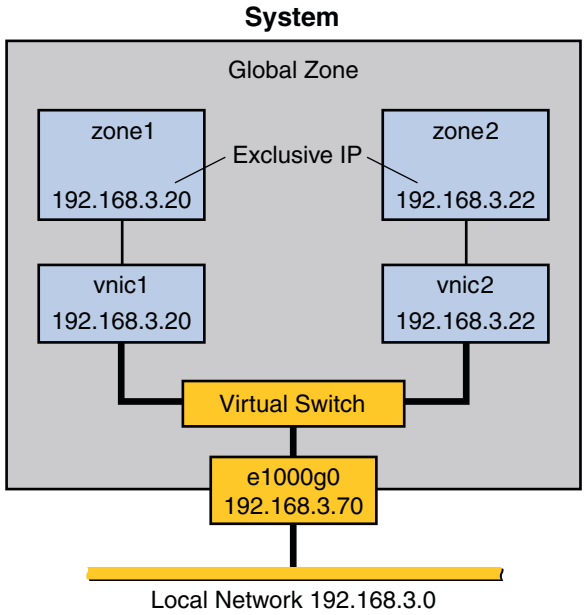

FIGURE 10–4 Basic Virtual Network With Flow Controls

The topology is described in "Basic Virtual Network on a Single System" on page 166. Here a host has one network interface, `e1000g0`, with two VNICs, `vnic1` and `vnic2`. `zone1` is configured over `vnic1`, and `zone2` is configured over `vnic2`. Resource management for the virtual network involves creating flows on a per-VNIC basis. These flows define and isolate packets with similar characteristics, such as port number or IP address of the sending host. You assign bandwidth based on the usage policy for the system.

Another very common usage for flow controls on VNIC traffic is by companies that rent out zones. You create different service level agreements for customers, and rent out zones with a guaranteed amount of bandwidth. When you create flows on a per-zone basis, you can isolate and observe each customer's traffic and monitor bandwidth usage. If your service level agreement is based strictly on usage, you can use Solaris statistics and accounting features to bill customers.

Flow controls are effective for any network that requires bandwidth management for traffic over zones. Larger organizations, such as application service providers (ASPs) or Internet service providers (ISP), can take advantage of resource control for VNICs for data centers and for multiprocessor systems. The individual zones can be rented out to customers for different levels of service. Therefore, you could rent out zone1 at the standard price and offer a standard bandwidth. Then, you could rent out zone2 at a premium price and give that customer a high level of bandwidth.

How to Create a Usage Policy for Applications on a Virtual Network

1. **List the applications that you want to run on the host.**
2. **Determine which applications have historically used the most bandwidth or require the most bandwidth.**

   For example, the `telnet` application might not consume huge amounts of bandwidth on your system, but it could be heavily used. Conversely, database applications consume a huge amount of bandwidth, but might only be used on a sporadic basis. Consider monitoring traffic for these applications prior to assigning them to zones. You can use the statistical option of the `dladm show-link` command to gather statistics, as described in "Gathering Usage Statistics for VNICs and Flows" on page 216.
3. **Assign these applications to separate zones.**
4. **Create flows for any application running in `zone1` whose traffic you want to isolate and control.**
5. **Assign bandwidth to flows based on usage policies in place for your site.**

How to Create a Service Level Agreement for the Virtual Network

1. **Design a policy that offers different levels of services at different prices.**

   For example, you might create a basic, superior, and high levels of service, and price each level accordingly.
2. **Decide whether you want to charge customers on a monthly, per service level basis, or charge customers on an actual bandwidth consumed basis.**

   If you choose the latter pricing structure, you need to gather statistics on each customer's usage.
3. **Create a virtual network on a host, with containers for each customer.**

   A very common implementation is to give each customer their own zone running over a VNIC.
4. **Create flows that isolate traffic for each zone.**

   To isolate all traffic for the zone, you use the IP address that is assigned to the zone's VNIC.
5. **Assign bandwidth to each VNIC based on the service level purchased by the customer assigned to that VNIC's zone.**

◆ ◆ ◆ CHAPTER 11

Configuring Virtual Networks (Tasks)

This chapter contains tasks for configuring internal virtual networks, or "networks in a box." The topics that are covered include:

- "Configuring a Basic Virtual Network" on page 176
- "Configuring a Private Virtual Network" on page 202

Virtual Networks Task Map

This table lists the tasks for configuring a virtual network, including links to the specific tasks. Note that not all tasks will apply to your virtual network scenario.

Task	Description	For Instructions
Begin creating a virtual network on a single host with access to the external network.	Create one or more virtual network interfaces (VNICs). VNICs are the pseudo-interfaces upon which you build the virtual network	"How to Create a Virtual Network Interface" on page 177
Create exclusive IP zones as the containers for the virtual network. **Note –** Use these tasks only if you want zones as the containers in the virtual network. To set up Sun xVM Server domains for network virtualization, refer to the Sun xVM Server Information Wiki.	Create, install, and boot one or more exclusive IP zones.	"How to Create an Exclusive IP Zone Over a VNIC" on page 181 and "How to Install the Exclusive IP Zone on a VNIC" on page 185

Task	Description	For Instructions
Complete virtual network configuration.	Complete initial zone configuration through the zone console, or manually configure IP addresses for the VNICs, and update the associated network databases.	"How to Configure an Exclusive IP Zone Over a VNIC Through the Zone Console" on page 188 or "How to Manually Configure the VNIC and Exclusive IP Zone" on page 190
Verify that the exclusive IP zone and VNIC are configured properly.	Perform a series of checks to validate the zone and VNIC configuration.	"How to Verify the Exclusive IP Zone Over VNIC Configuration" on page 193
Take down the existing virtual network.	Delete the VNICs and halt the zones prior to reconfiguration or other purposes.	"How to Remove the Virtual Network Without Removing the Zones" on page 200
Create a private virtual network on a single host.	Create the etherstub pseudo-interface that isolates the private network, plus the VNICs, and zones that complete the private network's structure.	"How to Create Etherstubs and VNICs for the Private Virtual Network" on page 202
Configure network-address translation and routing on the private virtual network.	Allow outbound traffic from the private network while denying inbound traffic from the external network.	"How to Configure Routing and Network Address Translation for the Private Virtual Network" on page 204

Configuring a Basic Virtual Network

This section contains tasks for configuring a basic virtual network. For a topology diagram of a virtual network, see Figure 10–1. Use the following tasks to build the virtual network.

Tip – The steps in all tasks in this chapter use the `vi` text editor in a terminal window. Alternatively, you can use the text editor of your choice.

How to Create a Virtual Network Interface

This procedure shows how to create a virtual network interface card (VNIC). *VNICs* are pseudo-interfaces upon which to build the containers of the virtual network. The resulting VNIC has an automatically generated MAC address. Depending on the network interface in use, you can instead explicitly assign a MAC address to a VNIC, as described in the `dladm(1M)`.

When you first log in to a system, you are automatically in its *global zone*, which is where you configure VNICs. You can use VNICs in the global zone or as the building blocks for a particular type of non-global zone, the exclusive IP zone. For an introduction to zones, refer to "Zones Overview" in *System Administration Guide: Virtualization Using the Solaris Operating System*.

1 Become superuser or assume the equivalent root role.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2 View information about the system's available physical interfaces.

```
# dladm show-phys
LINK         MEDIA                STATE      SPEED DUPLEX   DEVICE
e1000g2      Ethernet             unknown    0    half      e1000g2
e1000g0      Ethernet             up         1000 full      e1000g0
```

Currently the system has two installed interfaces, `e1000g0` and `e1000g2`.

3 Check the status of the data links on the system.

```
# dladm show-link
LINK        CLASS     MTU    STATE    OVER
e1000g2     phys      1500   unknown  --
e1000g0     phys      1500   up       --
```

Only the `e1000g0` data link is running over that interface and is configured "UP".

Unless you create customized names for your data links, the data link has the same name as the network interface device name that is displayed by `dladm show-phys`. For example, network interface `e1000g0` has the data link name `e1000g0` until you customize it. For more information on customized data link names, refer to "Data Link and IP Interface Configuration (Tasks)" on page 37.

4 Check the status of any interfaces on the IP layer.

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
         inet 127.0.0.1 netmask ff000000
e1000g0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 192.168.3.70 netmask ffffff00 broadcast 192.168.3.255
        ether 0:14:4f:94:d0:40
```

The output indicates that interface `e1000g0` has the IP address `192.168.3.70`. Therefore, the system is connected to the `192.168.3.0/24` network. `e1000g0` has the MAC address `0:14:4f:94:d0:40`.

5 Create a VNIC in the system's global zone.

`# dladm create-vnic -l` *data-link vnic-name*

- *data-link* is the name of the interface where the VNIC is to be configured.
- *vnic-name* is the name that you want to give the VNIC.

For example, to create a VNIC named `vnic0` on interface `e1000g0`, you would type the following:

```
# dladm create-vnic -l e1000g0 vnic0
```

Repeat this step for all planned VNICs in the virtual network.

6 Plumb the VNIC and assign it an IP address.

All VNICs must be configured and plumbed on the IP level. VNICs that are used in conjunction with an exclusive IP zone can be plumbed as part of the initial zone configuration or manually, using the steps in "How to Manually Configure the VNIC and Exclusive IP Zone" on page 190.

For VNICs to be configured in the *global zone*, do the following:

a. Use the `ifconfig` command as shown to configure the interface.

```
# ifconfig vnic-name plumb
# ifconfig vnic-name IP-address
# ifconfig vnic-name up
```

For example, you would configure and plumb `vnic0` over interface `e1000g0`as follows:

```
# ifconfig vnic0 plumb
# ifconfig vnic0 192.168.3.250
# ifconfig vnic0 up
```

b. Verify that the VNIC is configured and plumbed.

```
# ifconfig -a
```

Your output should resemble the following:

```
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL>
        mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0:flags=201100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4,CoS>
        mtu 1500 index 2
        inet 192.168.3.70 netmask ffffff00 broadcast 192.168.3.255
        ether 0:14:4f:94:d0:40
vnic0: flags=201100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4,CoS>
        mtu 9000 index 5
        inet 192.168.3.250 netmask ffffff00 broadcast 192.168.0.255
        ether 2:8:20:c2:39:38
```

Look for the VNIC that you just configured in the `ifconfig` output. For example, `vnic0` is in the previous output. The IP address that you specified and the `ifconfig` "UP" flag in the output must also be present. These items indicate that the VNIC is correctly configured and plumbed.

7 Ensure that the VNIC configuration persists across reboots

Create the file `/etc/hostname.`*vnic-name*.

- In the global zone, do the following:

  ```
  # cd /etc
  # vi hostname.vnic-name
  IP address of vnic-name
  ```

  For example, you type the following:

  ```
  # cd /etc
  # vi hostname.vnic0
  192.168.3.250
  ```

- Update the `/etc/inet/hosts` file with entries for all the VNICs you have created.

  The entries in the file should have the following format:

  vnic-IP-address **zone***ID*-*vnic-IP-address*

  For example, you might create the following entries:

  ```
  192.168.3.250      zone0-192-168-3-250
  ```

Note – When creating the zone alias entry, be sure to put a dash after the zoneID. Additionally, substitute dashes for the dot delimeters in the IP address, as shown previously.

- For exclusive IP zones, refer to the instructions in "How to Verify the Exclusive IP Zone Over VNIC Configuration" on page 193

8 Verify that the new VNIC is created.

```
# dladm show-vnic
LINK        SPEED  MACADDRESS         MACADDRTYPE
vnic0       0 Mbps  2:8:20:c2:39:38    random
```

Example 11–1 Creating Virtual Network Interfaces (VNIC)

This example contains the commands to use to create and verify three VNICs. One VNIC is used in the global zone. Two other VNICs are used with the exclusive IP zones in the upcoming tasks. This example illustrates the steps in "Configuring a Basic Virtual Network" on page 176 to accomplish the following:

- Create three VNICs, `vnic0`, `vnic1`, and `vnic2` on the data link `e1000g0`.
- Verify that the VNICs were created

  .
- Configure and plumb `vnic0` in the global zone.
- Make `vnic0` persist across reboots.

Note – You must log in to the system as superuser or equivalent role to run the next commands.

```
# dladm show-phys
LINK         MEDIA                STATE      SPEED DUPLEX   DEVICE
e1000g2      n                    unknown    0    half      e1000g2
e1000g0      Ethernet             up         1000 full      e1000g0
# dladm show-link
LINK        CLASS    MTU    STATE    OVER
e1000g2     phys     1500   unknown  --
e1000g0     phys     1500   up       --
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
         inet 127.0.0.1 netmask ff000000
e1000g0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 192.168.3.70 netmask ffffff00 broadcast 192.168.3.255
        ether 0:14:4f:94:d0:40

# dladm create-vnic -l e1000g0 vnic0
# dladm create-vnic -l e1000g0 vnic1
# dladm create-vnic -l e1000g0 vnic2
# dladm show-vnic
```

```
LINK        OVER             SPEED  MACADDRESS         MACADDRTYPE
vnic0       e1000g0      1000 Mbps  2:8:20:c2:39:38    random
vnic1       e1000g0      1000 Mbps  2:8:20:5f:84:ff    random
vnic2       e1000g0      1000 Mbps  2:8:20:54:f4:74    random

# ifconfig vnic0 plumb
# ifconfig vnic0 192.168.3.250
# ifconfig vnic0 up

# ifconfig -a

lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0:flags=201100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4,CoS>mtu 1500 index 2
        inet 192.168.3.70 netmask ffffff00 broadcast 192.168.3.255
        ether 0:14:4f:94:d0:40
vnic0: flags=201100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4,CoS> mtu 9000 index 5
        inet 192.168.3.250 netmask ffffff00 broadcast 192.168.0.255
        ether 2:8:20:c2:39:38

# vi /etc/hostname.vnic0
192.168.3.250
# vi /etc/inet/hosts
# Internet host table
#
::1     localhost
127.0.0.1       localhost
192.168.3.70    myhost     loghost
192.168.3.250      zone0-192-168-3-250
```

Next Steps

- To configure a zone over the VNIC, see "How to Create an Exclusive IP Zone Over a VNIC" on page 181.
- To configure an xVM domain over VNICs, information is to come.
- For an example of the configuration of a basic virtual network, see Example 11–6.

▼ How to Create an Exclusive IP Zone Over a VNIC

The following task explains how to create two exclusive IP zones for a virtual network. If you want to use zones as the containers for the virtual network, always use exclusive IP zones. You cannot create non–global shared IP zones over VNICs in a virtual network scenario.

As an alternative, you can use Sun xVM domains as the containers in the virtual network. For information about configuring Sun xVM Server and its domains, refer to theSun xVM Server Information Wiki.

Before You Begin This procedure assumes that you have already configured at least two VNICs over a data link, as shown in Example 11–1. The VNICs are named `vnic0`, `vnic1`, and `vnic2`.

1 On the system where you create the virtual network, become superuser or assume the equivalent root role.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2 View the state of the VNICs on the system.

```
# dladm show-vnic

LINK        OVER              SPEED  MACADDRESS         MACADDRTYPE
vnic1       e1000g0      1000 Mbps  2:8:20:5f:84:ff    random
vnic2       e1000g0      1000 Mbps  2:8:20:54:f4:74    random
```

The output indicates that `vnic1` and `vnic2` are currently configured over interface `e1000g0`.

3 Begin the creation process for the exclusive IP zone by running the `zonecfg` interactive utility.

Tip – Alternatively, you can run `zonecfg` as a command with appropriate subcommands and options to create the zone. For more information, refer to "How to Configure the Zone" in *System Administration Guide: Virtualization Using the Solaris Operating System* and the `zonecfg(1M)` man page.

```
# zonecfg -z zoneID
```

where *ID* represents the number to identify the zone. For example, the following command creates "`zone1`."

```
# zonecfg -z zone1
```

The `zonecfg` program runs and prompts for information about the new zone.

```
zonecfg:zone1>
```

4 Start zone creation through the `zonecfg` interactive utility.

```
zonecfg:zone1> create
```

The remaining steps show how to create the exclusive IP zone and set other parameters. For a detailed description of parameters available for the zone, see "How to Configure the Zone" in *System Administration Guide: Virtualization Using the Solaris Operating System*.

5 Create the zone path by setting a home directory for the zone, and then enable automatic booting.

```
zonecfg:zone1> set zonepath=zone-home-directory
zonecfg:zone1> set autoboot=true
```

For example, *zone-home-directory* might be `/export/home/zone1`.

The global zone will include home directories for all zones that you create through `zonecfg`. Thus, the `/export/home` directory in the global zone must contain an entry for `zone1`.

6 Create the zone as exclusive IP.

```
zonecfg:zone1> set ip-type=exclusive
```

7 Create the network interface for the zone.

```
zonecfg:zone1> add net
```

This response starts the network configuration subprogram of `zonecfg`.

8 Set the previously configured VNIC as the interface for the zone.

```
zonecfg:zone1:net> set physical=vnic-data-link
```

For example, you create `vnic1` for `zone1` as follows:

```
zonecfg:zone1:net> set physical=vnic1
```

Note – Although `zonecfg` has many options for describing a network interface, only use the `set-physical` parameter of `add net` for an IP exclusive zone.

---

9 Complete zone configuration and verify the results.

```
zonecfg:zone1:net> end
zonecfg:zone1> verify
```

The `verify` command checks for any configuration errors. If you have received errors, fix the configuration. If `verify` does not respond, assume the configuration is correct and continue.

10 View information about the zone you just created.

Use the `info` directive, as shown below:

```
zonecfg:zone1> info
zonename: zone1
zonepath: /export/home/zone1
brand: native
autoboot: true
.
.
net:
```

```
        address not specified
        physical: vnic1
```

The message "`address not specified`" verifies that you have not specified an IP address for the zone. You create IP addresses for the zone's VNIC outside the `zonecfg` utility, as described in the upcoming procedure "How to Configure an Exclusive IP Zone Over a VNIC Through the Zone Console" on page 188.

If `info` displays other incorrect information, you can modify the parameters, as explained in "Using the zonecfg Command to Modify a Zone Configuration" in *System Administration Guide: Virtualization Using the Solaris Operating System*. If the information is correct, continue to the next step.

11 Commit the zone and close `zonecfg`.

```
zonecfg:zone1> commit
zonecfg:zone1> exit
```

Be sure to commit the zone before exiting `zonecfg`.

12 Create more zones, as needed, by following Steps 3 through 11.

Example 11–2 Creating an Exclusive IP Zone Over a VNIC

The following example contains the commands for creating a zone using the `zonecfg` utility. When the example is complete, the result is a zone called `zone1` that is configured on `vnic1`. This example assumes that the VNIC is already created, as shown in Example 11–1. You can use this example for configuring as many exclusive IP zones over VNICs as you need for your virtual network. For an illustration of a basic virtual network, refer to Figure 10–1.

You must log in to the global zone of the system as superuser or equivalent role to run the next commands.

```
# dladm show-vnic

LINK        OVER              SPEED  MACADDRESS         MACADDRTYPE
vnic1       e1000g0      1000 Mbps  2:8:20:5f:84:ff    random
vnic2       e1000g0      1000 Mbps  2:8:20:54:f4:74    random

# zonecfg -z zone1

zonecfg:zone1> create
zonecfg:zone1> set zonepath=/export/home/zone1
zonecfg:zone1> set autoboot=true
zonecfg:zone1> set ip-type=exclusive
zonecfg:zone1> add net
zonecfg:zone1:net> set physical=vnic1
zonecfg:zone1:net> end
zonecfg:zone1> verify
```

```
zonecfg:zone1> info
zonename: zone1
zonepath: /export/home/zone1
brand: native
autoboot: true
.
.
net:
        address not specified
        physical: vnic1

zonecfg:zone1> commit
zonecfg:zone1> exit
```

Next Steps

- To continue with zone creation, go to "How to Install the Exclusive IP Zone on a VNIC" on page 185.
- For detailed information about the `zonecfg` command, refer to `zonecfg(1M)`.

▼ How to Install the Exclusive IP Zone on a VNIC

Before You Begin This procedure assumes that you have completed VNIC creation, as described in "How to Create a Virtual Network Interface" on page 177. You also must have created and committed an exclusive IP zone, as described in "How to Create an Exclusive IP Zone Over a VNIC" on page 181.

In this procedure you install the newly created `zone1` over `vnic1`.

1 On the system where you create the virtual network, become superuser or assume the equivalent root role.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

Note – When you first log in to a system, you are automatically in its *global zone*. For an introduction to zones, refer to "Zones Overview" in *System Administration Guide: Virtualization Using the Solaris Operating System*.

---

2 Verify that the new zone exists.

```
# zoneadm -z zoneID verify
```

The `zoneadm` command displays output similar to the following for a zone that is not yet installed:

```
WARNING: /export/home/zone1 does not exist, so it could not be verified.
When 'zoneadm install' is run, 'install' will try to create
```

```
/export/home/zone1, and 'verify' will be tried again,
but the 'verify' may fail if:
the parent directory of /export/home/zone1 is group- or other-writable
or
/export/home/zone1 overlaps with any other installed zones.
```

This message indicates that zone is ready to be installed.

3 Install the new zone.

Use the following syntax:

```
# zoneadm -z zoneID install
```

For example, you would type:

```
# zoneadm -z zone1 install
Preparing to install zone <zone1>
Creating list of files to copy from the global zone.
.
.

Zone <zone1> is initialized.
```

4 Verify that the zone is installed.

```
zoneadm list -iv
```

You receive output similar to the following:

```
 ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   - zone1            installed  /export/home/zone1             native   excl
```

The output indicates that the exclusive IP zone is installed but not yet running.

5 Boot the zone and then observe its new status.

```
# zoneadm -z zone1 boot
# zoneadm list -v
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   1 zone1            running    /export/home/zone1             native   excl
```

Note that `zone1` has changed its state to `running`.

6 Repeat this procedure for all exclusive IP zones in your virtual network.

Example 11–3 Installing and Booting an Exclusive IP Zone Over a VNIC

The following example contains the `zoneadm` and `zlogin -C` commands for installing the exclusive IP zone `zone1` that is configured over `vnic1`. This example assumes that both the VNIC and zone are created, as shown in Example 11–2. You can use this example for installing every exclusive IP zone over a VNIC for your virtual network. For an illustration of a basic virtual network, refer to Figure 10–1.

You must log in to the global zone of the system as superuser or equivalent role to run the next commands.

```
# zoneadm -z zone1 verify
WARNING: /export/home/zone1 does not exist, so it could not be verified.
When 'zoneadm install' is run, 'install' will try to create
/export/home/zone1, and 'verify' will be tried again,
but the 'verify' may fail if:
the parent directory of /export/home/zone1 is group- or other-writable
or
/export/home/zone1 overlaps with any other installed zones.

# zoneadm -z zone1 install
Preparing to install zone <zone1>.
Creating list of files to copy from the global zone.
.
.
Zone <zone1> is initialized.

zoneadm list -iv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   - zone1            installed  /export/home/zone1             native   excl


# zoneadm -z zone1 boot
# zoneadm list -v
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   1 zone1            running    /export/home/zone1             native   excl
```

Next Steps After booting the zone, you need to perform initial configuration steps for the exclusive IP zone over a VNIC. Use one of the following methods to complete zone configuration:

- Perform initial zone configuration on the newly-booted zone from the zone console, as explained in "How to Configure an Exclusive IP Zone Over a VNIC Through the Zone Console" on page 188.

- Manually perform basic zone and VNIC configuration, including plumbing the VNIC and updating the affected network files. For instructions, see "How to Manually Configure the VNIC and Exclusive IP Zone" on page 190.
- Configure the necessary parameters for zone configuration in the `/etc/sysidcfg` file, as explained in "How to Use an /etc/sysidcfg File to Perform the Initial Zone Configuration" in *System Administration Guide: Virtualization Using the Solaris Operating System*. If you configure the zone through `/etc/sysidcfg`, you might need to manually add IP addresses for the zone, as shown in "How to Manually Configure the VNIC and Exclusive IP Zone" on page 190.

▼ How to Configure an Exclusive IP Zone Over a VNIC Through the Zone Console

After you have installed and booted all zones for the virtual network, your final step is to configure the zones.

Before You Begin You must have created, installed, and booted exclusive IP zones over VNICs, as explained in the following procedures:

- "How to Create an Exclusive IP Zone Over a VNIC" on page 181
- "How to Install the Exclusive IP Zone on a VNIC" on page 185

1 On the system where you create the virtual network, become superuser or assume the equivalent root role.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2 Log in to the console of a zone

Begin initial zone configuration through the zone console.

```
# zlogin -C zone-name
```

where *zone-name* represents the name of the zone that you want to configure. For example, to log in to the console for `zone1`, type the following:

```
# zlogin -C zone1
```

Depending on your system, you might receive prompts from the console to set language preference and other parameters. Answer these prompts and continue.

3 Select a terminal type.

The zone configuration program offers choices such as the following

```
What type of terminal are you using?
      1) ANSI Standard CRT
      2) DEC VT52
.
.
      8) Sun Workstation
      9) Televideo 910
      10) Televideo 925
      11) Wyse Model 50
      12) X Terminal Emulator (xterms)
```

Type the number for the console terminal type for your system, for example **12** for an X terminal window.

4 Confirm or change the information displayed by the zone configuration program.

You receive a series of prompts for information about the new zone. Most of the responses are automatically generated. If the information is incorrect, you can press F4 and supply the correct information. Otherwise, press F2 to accept and continue to the next parameter.

The information that you need to supply or verify includes:

- IP address for the zone. Each exclusive IP zone and its corresponding VNIC must have a unique IP address. You can use a DHCP address or a static IP address.
- Host name. Enter the host name for the zone, for example, `zone1`.
- Whether the system with the virtual network is part of a subnet.
- Netmask of the IP address.
- Default route. You can use the IP address of the interface on which the virtual network is built.
- IP address of a router on the system's network

When you are finished configuring the zone, the system reboots. After the reboot, the zone is ready for use.

5 Repeat the initial configuration steps for all zones in the virtual network.

Example 11–4 Final Configuration of an Exclusive IP Zone Over a VNIC

This example shows a typical zone configuration session using the zone console configuration program.

```
# zlogin -C zone1
What type of terminal are you using?
.
```

```
.
.
8) Sun Workstation
9) Televideo 910
10) Televideo 925
11) Wyse Model 50
12) X Terminal Emulator (xterms)
13) CDE Terminal Emulator (dtterm)
14) Other
Type the number of your choice and press Return: 13
.
.
IP address for zone1: 192.168.3.20
.
Confirm the following information. If it is correct, press F2;
to change any information, press F4.

Hostname: zone1
IP address: 192.168.3.20
System part of a subnet: Yes
Netmask: 255.255.255.0
Enable IPv6: No
Default route: 192.168.3.70
Router IP address: 192.168.3.25
```

System reboots.

Next Steps Verify that zone configuration is correct, as explained in "How to Verify the Exclusive IP Zone Over VNIC Configuration" on page 193.

▼ How to Manually Configure the VNIC and Exclusive IP Zone

This procedure explains how to manually configure IP addresses for VNICs and their associated zones. If you configured zones through the zone console after the initial booting, these addresses are configured automatically. You need to follow the next steps only if one of the following conditions is true:

- You did not run the zone console configuration program after booting the zones and want to configure IP addresses manually. In this case, you should perform all the steps in the procedure.
- You performed the validation checks in "How to Verify the Exclusive IP Zone Over VNIC Configuration" on page 193and uncovered problems. Some typical problems include the VNIC was not plumbed, or problems with a relevant files, such as `hostname.`*vnic-name*. In this case, complete only the steps that relate to the specific problems that you found.

Before You Begin The procedure assumes that both the VNIC and zone are created, installed, and booted in the global zone.

1 On the system where you create the virtual network, become superuser or assume the equivalent root role.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2 Log in to the zone.

For example, you would type:

```
# zlogin zone1
# pwd
/
```

3 Verify that the VNIC is configured.

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
         inet 127.0.0.1 netmask ff000000
lo0: flags=2002000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6,VIRTUAL> mtu 8252 index 1
     inet6 ::1/128
```

In this output, only the IPv4 and IPv6 loopback addresses are plumbed and up. No entry exists for the VNIC.

4 Manually configure and plumb the VNIC from within the exclusive IP zone.

You must plumb a VNIC in the following order for it to function properly in the virtual network.

```
# ifconfig vnic-data-link plumb
# ifconfig vnic-data-link IP-address
# ifconfig vnic-data-link up
```

For example, to add IP address `192.168.3.20` to `vnic1`, do the following:

```
# ifconfig vnic1 plumb
# ifconfig vnic1 192.168.3.20
# ifconfig vnic1 up
```

5 Verify that the VNIC is now configured and plumbed.

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
         inet 127.0.0.1 netmask ff000000
vnic1: flags=201000842<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
           inet 192.168.3.20 netmask ffffff00 broadcast 192.168.3.255
           ether 2:8:20:54:f4:74
```

6 Exit the exclusive IP zone, and go to the zone's subdirectory tree in the global zone.

```
# exit
# cd /export/home/zone1
```

7 Create a `hostname.`*vnic–name* file for the VNIC.

```
# cd root/etc
# vi hostname.vnic1
zoneID-IP address
```

For example, for `zone1` you type:

```
zone1-192.183.3.20
```

8 Add an entry for the zone in the `root/etc/inet/hosts` file.

```
# cd inet
# vi hosts
# Internet host table
#
::1                 localhost
127.0.0.1           localhost
192.168.3.20  zone1  loghost
```

9 If the entry does not already exist, add the VNIC and its zone to the global zone's `/etc/inet/hosts` file.

```
# cd /etc/inet
# vi hosts
# Internet host table
#
::1     localhost
127.0.0.1         localhost
192.168.3.70      myhost     loghost
192.168.3.20      zone1-192-168-3-20
```

Example 11–5 Manually Configuring a VNIC and Exclusive IP Zone

This example illustrates the following procedures:

- Plumbing `vnic1` from within a zone and assigning an IP address to the VNIC.
- Adding the IP address for `zone1` and `vnic1` to the appropriate files, so that this IP address persists across reboots.

You must log in to the global zone of the system as superuser or equivalent role to run the next commands.

```
# zlogin zone1
/
```

```
# ifconfig vnic1 plumb
# ifconfig vnic1 192.168.3.20
# ifconfig vnic1 up
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
         inet 127.0.0.1 netmask ff000000
vnic1: flags=201000842<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
           inet 192.168.3.20 netmask ffffff00 broadcast 192.168.3.255
           ether 2:8:20:54:f4:74
# exit
# cd /export/home
# cd zone1/root/etc
# vi hostname.vnic1
zone1-192.168.3.20

# vi inet/hosts
# Internet host table
#
::1                 localhost
127.0.0.1           localhost
192.168.3.20  zone1  loghost

# cd /etc/inet
# vi hosts
# Internet host table
#
::1     localhost
127.0.0.1        localhost
192.168.3.70     myhost     loghost
192.168.3.20     zone1-192-168-3-20
```

Next Steps After you are finished, verify that your configuration is correct, as explained in "How to Verify the Exclusive IP Zone Over VNIC Configuration" on page 193.

▼ How to Verify the Exclusive IP Zone Over VNIC Configuration

After you complete zone configuration, confirm that the zones and VNICs are now configured as you expected.

Before You Begin The procedures in this task assume that you have installed and configured two or more exclusive IP zones over a VNIC. If you have not done this, perform the following procedures, in sequential order:

- "How to Create an Exclusive IP Zone Over a VNIC" on page 181

- "How to Install the Exclusive IP Zone on a VNIC" on page 185
- "How to Configure an Exclusive IP Zone Over a VNIC Through the Zone Console" on page 188 or "How to Manually Configure the VNIC and Exclusive IP Zone" on page 190

1 On the system where you build the virtual network, become superuser or assume the equivalent root role in the global zone.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2 Go to the parent directory for all zones that you created.

You supply this directory to the `zonecfg` command as the first part of the *zone path*.

```
# cd parent-zone-path
```

For example, to access the parent directory for both zones created in the procedure "How to Create an Exclusive IP Zone Over a VNIC" on page 181, type:

```
# cd /export/home
```

If the parent directory for the zones does not exist, check your zone configuration.

3 Verify that the zone home directory trees exist in the correct parent directory in the global zone.

```
# pwd
/export/home
# ls
zone-name
```

For example, to verify that the zone subdirectories have been created in the parent `/export/home` directory, in the global zone, type:

```
# ls
zone1 zone2
```

The subdirectories for the two new zones have been created. If these subdirectories do not exist, check your zone configuration.

4 Verify that the `hostname.`*vnic-name* file exists and that its entry is correct.

Each VNIC that you configure for a zone requires a `hostname.`*vnic-name* file to ensure that the IP address of the VNIC and zone persist after reboots. First, verify that a `hostname.`*vnic-name* file exists:

```
cd /export/home/zone-name/root/etc
# ls host*
 hostname.vnic1  hosts
```

This output indicates that a `hostname.vnic1` file exists. The file should contain one entry with the name of the zone, for example:

```
cat hostname.vnic1
zone1
```

If this file does not exist, create it as shown in "How to Manually Configure the VNIC and Exclusive IP Zone" on page 190.

5 Check the contents of the zone's `hosts` file.

```
# pwd
/export/home/zone-name/root/etc/
# cat hosts
# Internet host table
#
::1                 localhost
127.0.0.1           localhost
192.168.3.20  zone1  loghost
```

In this output, the entry `192.168.3.20 zone1 loghost` shows the address that is assigned to the VNIC for `zone1`. Your output should have a similar entry for the zone and VNIC.

If this file does not have an entry for the zone, refer to the appropriate step in "How to Manually Configure the VNIC and Exclusive IP Zone" on page 190.

6 Add the IP addresses of the VNICs and names of their associated zones to the `/etc/inet/hosts` file in the global zone.

Note – Be sure that you are in the `hosts` file for the global zone, not the `host` file in a subdirectory tree for a zone.

```
# cd /etc/inet
# vi hosts
# Internet host table
#
::1     localhost
127.0.0.1       localhost
192.168.3.70    myhost     loghost
```

The only non-loopback IP address in this output is `192.168.3.70`, the address associated with the system's network interface. Add entries for all VNICs associated with zones to this file, using the following format:

VNIC-IP-address zone-name- IP address

For example, you would type the following entry for `vnic1` and `zone1`:

```
192.168.3.20    zone1-192-168-3-20
```

7 Log in to the new zone and verify that you are in its home directory:

For example, for `zone1` you would type:

```
# zlogin zone1
# pwd
/
```

You are now in the root directory of `zone1`. If you cannot log in to the zone, check your zone configuration.

8 Verify that the VNIC you previously defined for the zone is now configured as an IP interface.

Your output should resemble the following:

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
vnic1: flags=201000842<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
          inet 192.168.3.20 netmask ffffff00 broadcast 192.168.3.255
          ether 2:8:20:54:f4:74
```

In the output, `vnic1` is configured with the IP address that you specified during zone configuration. `vnic1` also has an automatically generated unique MAC address `ether 2:8:20:54:f4:74` . Note that there are no entries for the system's network interfaces or for VNICs that are configured for other zones.

If you do not have an entry for the VNIC associated with the zone, you need to plumb the VNIC. In particular, you will have these results if you chose not to perform initial VNIC configuration from the zone console. For instructions for plumbing the VNIC, refer to the appropriate step in "How to Manually Configure the VNIC and Exclusive IP Zone" on page 190.

9 Exit the current zone.

Return to the global zone, where you can repeat the previous steps to confirm that all VNICs and zones are properly configured.

Next Steps You can use various tools to observe network traffic and take statistics on zone usage.

- To verify that your network is properly configured, refer to "Verifying Virtual Network Connectivity" on page 210
- To observe traffic over the virtual network, refer to "How to Verify Virtual Network Connectivity by Using the snoop Command" on page 214.
- To obtain statistics for accounting purposes, refer to "Gathering Usage Statistics for VNICs and Flows" on page 216.

If you need to disassemble the virtual network, refer to "How to Remove the Virtual Network Without Removing the Zones" on page 200.

Complete Example for Creating a Virtual Network

This section contains a complete set of commands for configuring a virtual network.

EXAMPLE 11–6 Basic Virtual Network

This example shows how to implement the virtual network scenario shown in Figure 10–1. The example elaborates on the tasks presented in "Configuring a Basic Virtual Network" on page 176. The commands do the following:

- Configure two VNICs, `vnic1` and `vnic2` on the data link `e1000g0`.
- Configure two exclusive IP zones, `zone1` and `zone2`.

Note – The example shows only the steps to configure `zone1`. Repeat the same steps to create and configure `zone2`.

- Assign automatically configured MAC addresses to each VNIC.
- Set two static IP addresses for the zones and VNICs, `192.168.3.20` and `192.168.3.22`.

Note – You must log in to the system's global zone as superuser or equivalent role to run the next commands.

```
# dladm show-phys
# dladm show-link
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
         inet 127.0.0.1 netmask ff000000
e1000g0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 192.168.3.70 netmask ffffff00 broadcast 192.168.3.255
        ether 0:14:4f:94:d0:40

# dladm create-vnic -l e1000g0 vnic1
# dladm create-vnic -l e1000g0 vnic2
# dladm show-vnic
LINK        OVER             SPEED  MACADDRESS         MACADDRTYPE
vnic1       e1000g0      1000 Mbps  2:8:20:5f:84:ff    random
vnic2       e1000g0      1000 Mbps  2:8:20:54:f4:74    random

# zonecfg -z zone1
```

EXAMPLE 11–6 Basic Virtual Network *(Continued)*

```
zonecfg:zone1> create
zonecfg:zone1> set zonepath=/export/home/zone1
zonecfg:zone1> set autoboot=true
zonecfg:zone1> set ip-type=exclusive
zonecfg:zone1> add net
zonecfg:zone1:net> set physical=vnic1
zonecfg:zone1:net> end
zonecfg:zone1> verify

zonecfg:zone1> info
zonename: zone1
zonepath: /export/home/zone1
brand: native
autoboot: true
.
.
net:
        address not specified
        physical: vnic1

zonecfg:zone1> commit
zonecfg:zone1> exit

# zoneadm -z zone1 verify
WARNING: /export/home/zone1 does not exist, so it could not be verified.
When 'zoneadm install' is run, 'install' will try to create
/export/home/zone1, and 'verify' will be tried again,
but the 'verify' may fail if:
the parent directory of /export/home/zone1 is group- or other-writable
or
/export/home/zone1 overlaps with any other installed zones.

# zoneadm -z zone1 install
Preparing to install zone <zone1>.
Creating list of files to copy from the global zone.
.
.
Zone <zone1> is initialized.


zoneadm list -iv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   - zone1            installed  /export/home/zone1             native   excl

# zoneadm -z zone1 boot
```

EXAMPLE 11–6 Basic Virtual Network *(Continued)*

```
# zoneadm list -v
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   1 zone1            running    /export/home/zone1             native   excl

# zlogin zone1
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
inet 127.0.0.1 netmask ff000000
lo0: flags=2002000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6,VIRTUAL> mtu 8252 index 1
 inet6 ::1/128

# ifconfig vnic1 plumb
# ifconfig vnic1 192.168.3.20
# ifconfig vnic1 up

# ifconfig -a
.
vnic1: flags=201000842<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 192.168.3.20 netmask ffffff00 broadcast 192.168.3.255
        ether 2:8:20:54:f4:74

# pwd
vnic1/
# cd root/etc
# vi hostname.vnic1
zone1-192.183.3.20

# vi /etc/inet/hosts
# Internet host table
#
::1     localhost
127.0.0.1         localhost
192.168.3.70      myhost     loghost
192.168.3.20      zone1-192-168-3-20
```

After you repeat the same steps to create `zone2` and to assign `vnic2` to `zone2`, the following example shows you how to verify that the two zones are properly configured with their respective VNICs.

```
# zoneadm list -v
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   1 zone1            running    /export/home/zone1             native   excl
   2 zone2            running    /export/home/zone2             native   excl
```

EXAMPLE 11–6 Basic Virtual Network *(Continued)*

```
# vi /etc/inet/hosts
# Internet host table
#
::1     localhost
127.0.0.1         localhost
192.168.3.70      myhost     loghost
192.168.3.20      zone1-192-168-3-20
192.168.3.22      zone2-192-168-3-22
```

How to Remove the Virtual Network Without Removing the Zones

The following procedure shows how to take down a virtual network while leaving its zones intact. The instructions refer to the virtual network that is configured in "Configuring a Basic Virtual Network" on page 176.

Use this procedure if you must do any of the following:

- Use the existing zones in a different configuration. For example, you might need to configure the zones as part of a private network. See "Configuring a Private Virtual Network" on page 202.
- Migrate the zones to another network.
- Move the zones to a different zone path.
- Clone the zones, as explained in "Cloning a Non-Global Zone on the Same System" in *System Administration Guide: Virtualization Using the Solaris Operating System*.

Before You Begin This task assumes that you have a running virtual network that consists of exclusive IP zones.

1 On the system with the virtual network, become superuser or assume the equivalent root role in the global zone.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2 Verify the state of the currently configured zones.

```
# zoneadm list -v
```

For example, you receive output similar to the following:

```
ID  NAME     STATUS       PATH                           BRAND           IP
 0  global   running      /                              native          shared
```

```
 1  zone1    running      /export/home/zone1            native           excl
 2  zone2    running      /export/home/zone2            native           excl
```

3 Halt the exclusive IP zones of the virtual network.

Issue the following command separately for each zone to be halted.

```
# zoneadm -z zone-name halt
```

Replace *zone-name* with the name of each zone.

When you halt the zone, you remove the zone's application environment and terminate a number of system activities, as explained in "Halting a Zone" in *System Administration Guide: Virtualization Using the Solaris Operating System*.

4 Verify that the zones have been halted.

```
# zoneadm list -iv
```

You receive output similar to the following:

```
ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   - zone1            installed  /export/home/zone1             native   excl
   - zone2            installed  /export/home/zone2             native   excl
```

Note that the zones are no longer running, although they remain installed. To reboot a halted zone, refer to "How to Boot a Zone" in *System Administration Guide: Virtualization Using the Solaris Operating System*.

5 Review the state of the VNICs that were configured for the halted zones.

```
# dladm show-vnic
```

You receive output similar to the following:

```
LINK        OVER              SPEED  MACADDRESS         MACADDRTYPE
vnic1       e1000g0      1000 Mbps  2:8:20:5f:84:ff    random
vnic2       e1000g0      1000 Mbps  2:8:20:54:f4:74    random
```

The resulting output shows that the VNICs are still configured as data links in the global zone. These VNICs were only plumbed and up in their associated exclusive IP zones, which are now halted. These VNICs are not plumbed in the global zones.

6 Delete the VNICs.

```
# dladm delete-vnic vnic-link-name
```

For example, you would type the following to delete the VNICs in the zones in Figure 10–1.

```
# dladm delete-vnic vnic1
# dladm delete-vnic vnic1
```

Next Steps You can perform further operations on the existing zones, as required.

- To restart the zones without reconfiguring them in a virtual network, refer to "How to Boot a Zone" in *System Administration Guide: Virtualization Using the Solaris Operating System*.
- To fully delete the zones, refer to "How to Remove a Non-Global Zone" in *System Administration Guide: Virtualization Using the Solaris Operating System*.
- To reconfigure the zones as part of a virtual network, create new VNICs and modify the zones, as described in "Configuring a Basic Virtual Network" on page 176.

Configuring a Private Virtual Network

The tasks in this section explain how to configure a *private virtual network* on a single system. If you need to isolate a software development environment from the external network, consider creating a private virtual network on a single host.

Note – Private virtual networks are quite different from private virtual networks (VPNs). VPN software creates a secure point-to-point link between two endpoint systems. The private network configured by the tasks in this section is a virtual network on a box that cannot be accessed by external systems.

Pseudo-network interfaces called *etherstubs* are the building blocks of private virtual networks, as shown in "Private Virtual Network on a Single System" on page 168. You create VNICs over the etherstub, and then configure the containers over the VNICs. A firewall or similar network address translation (NAT) device translates the VNIC's private IP addresses to the routable IP address of the network interface. This enables the containers of the private network to send packets beyond the host without exposing the VNICs' private IP addresses to the external network.

▼ How to Create Etherstubs and VNICs for the Private Virtual Network

This procedure uses exclusive IP zones as the containers for the private virtual network. Solaris IP Filter software performs NAT for outgoing packets from the private network.

Before You Begin For the VNICs in the private network configuration, be sure to create private IP addresses that cannot be forwarded by the default router of the external network. However, for the network interface, use an IP address that is routable on the host's external network.

1. **On the system where you create the private virtual network, become superuser or assume the equivalent root role.**

   To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2. **Create the etherstub for the private virtual network.**

   ```
   # dladm create-etherstub etherstub-link-name
   ```

   For example, to create an etherstub called `etherstub0`, you would type the following:

   ```
   # dladm create-etherstub etherstub0
   ```

3. **Verify that the etherstub was created.**

   ```
   # dladm show-etherstub
   ```

   You should receive output similar to the following:

   ```
   LINK
   etherstub0
   ```

4. **Create VNICs over the etherstub.**

   ```
   # dladm create-vnic -l etherstub-link-name vnic-link-name
   ```

   For example, you might type the following:

   ```
   # dladm create-vnic -l etherstub0 vnic0
   ```

   Reserve one VNIC for the global zone. The global zone consists of all applications and services of a system's working environment that have not been delegated to a zone or virtual machine.

   Then, create at least two more VNICs for the exclusive IP zones of the private network. The virtual switch is automatically created with the first VNIC.

5. **Verify that the VNICs are correctly created over the etherstub.**

   ```
   # dladm show-link
   ```

   You should receive output similar to the following:

   ```
   LINK        CLASS    MTU    STATE    OVER
   e1000g0     phys     1500   up       --
   vnic0       vnic     9000   up       etherstub0
   ```

   The "OVER" column contains the entry `etherstub0` in the `vnic0` row, indicating that `vnic0` is created over `etherstub0`.

6. **Create the exclusive IP zones.**

   - For instructions, refer to "How to Create an Exclusive IP Zone Over a VNIC" on page 181.

- Be sure to type the associated VNIC data link name for the zone in the `set-physical` parameter of `add-net`.

7 Install the zones.

Use Steps 1–4 in the procedure "How to Install the Exclusive IP Zone on a VNIC" on page 185

Note – Do not boot the zones at this time. You boot them as part of the next procedure, "How to Configure Routing and Network Address Translation for the Private Virtual Network" on page 204.

▼ How to Configure Routing and Network Address Translation for the Private Virtual Network

Before You Begin This procedure assumes that you have created the etherstub, VNICs, and exclusive IP zones or virtual machines for the private network, as described in "How to Create Etherstubs and VNICs for the Private Virtual Network" on page 202.

1 On the system where you create the private virtual network, become superuser or assume the equivalent root role.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2 Check the status of the host's network interface.

```
# ifconfig -a
```

You should receive output similar to the following:

```
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 192.168.3.70 netmask ffffff00 broadcast 192.168.3.255
        ether 0:14:4f:94:d0:40
```

The interface, `e1000g0` in this case, must be configured and plumbed before you can use it as part of the virtual network.

3 Assign an IP address to the VNIC that you reserved for the global zone.

Make sure that all IP addresses that you assign to the VNICs on this host are private, reserved for use on this host only. Do not use the IP address prefix of the public network to which the network interface is connected as the network portion of the VNIC's IP address.

For example, the `ifconfig -a` command above shows the IP address `192.168.3.70` for interface `e1000g0`. The output indicates that the interface is on local network `192.168.3.0/24`.

Therefore, do not assign the IP address `192.168.3.`*x* to the VNIC. A safer choice might be `192.168.0.250`, assuming that there is no `192.168.0.0/24` network that is known to the default router.

For specific instructions on assigning the IP address to the VNIC, refer to Steps 5 through 7 of "How to Create a Virtual Network Interface" on page 177.

4 Check the status of routing protocols on the system.

```
# routeadm
```

You should receive output similar to the following:

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

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

Note that routing is enabled but packet forwarding is disabled. You need to enable IPv4 forwarding in the global zone before you set up NAT or other rules through the IP Filter firewall.

5 Enable IP forwarding.

```
# routeadm -u -e ipv4-forwarding
```

6 Create the basic packet filtering file `/etc/ipf/ipnat.conf` to provide network address translation.

The next steps use Solaris IP Filter to perform NAT for outgoing packets originated from inside the private network. For an introduction to IP Filter, refer to Chapter 24, "Solaris IP Filter (Overview)," in *System Administration Guide: IP Services*

```
# cd /etc/ipf
# vi ipnat.conf
map e1000g0 192.168.0.0/24 -> 0/32 portmap tcp/udp auto
map e1000g0 192.168.0.0/24 -> 0/32
```

This rule set tells the IP Filter software how to translate the IP addresses of outgoing packets when they arrive at interface `e1000g0`. Any TCP and UDP packets that arrive from private network `192.168.0.0/24` have their IP addresses translated to the address of the global zone before exiting the system. The global zone has the same IP address as network interface `e1000g0`, `192.168.3.70`. This interface is connected to external network `192.168.3.0/24`, which is known to the network's default router.

The rule set above implements a simple NAT scenario, but you can also add packet filtering rules to `/etc/ipf/ipnat.conf`, if required. For more information, see "Configuring Solaris IP Filter" in *System Administration Guide: IP Services*.

7 Start IP Filter and verify that the rules in `/etc/ipf/ipnat.conf` are active.

```
# svcadm enable network/ipfilter
# ipnat -l
List of active MAP/Redirect filters:
map e1000g0 192.168.0.0/24 -> 0.0.0.0/32 portmap tcp/udp auto
map e1000g0 192.168.0.0/24 -> 0.0.0.0/32

List of active sessions:
```

8 Boot an already-installed exclusive IP zone.

```
# zoneadm -z zone-name boot
```

Repeat this step for all zones to be part of the private virtual network.

9 Log in to each exclusive IP zone and plumb its associated VNIC.

```
# zlogin zone-name
# ifconfig vnic-link-name plumb
#ifconfig vnic-link-name vnic-IP-address
# ifconfig vnic-link-name up
```

10 Exit the final zone that you configured and return to the global zone.

11 Add entries for all VNICs in the `/etc/inet/hosts` file, as shown in "How to Manually Configure the VNIC and Exclusive IP Zone" on page 190.

12 Edit the `/etc/hostname/`*vnic-name* files, as shown in "How to Manually Configure the VNIC and Exclusive IP Zone" on page 190.

Example 11–7 Private Virtual Network Configuration

The following example shows the commands to implement the private virtual network that is shown in Figure 10–2.

To use the commands, you must first log in to the system's global zone as superuser or equivalent role.

```
# dladm create-etherstub etherstub0
# dladm show-etherstub
LINK
etherstub0
# dladm create-vnic -l etherstub0 vnic0
```

```
# dladm create-vnic -l etherstub0 vnic1
# dladm create-vnic -l etherstub0 vnic2

# dladm show-vnic
LINK        OVER             SPEED  MACADDRESS         MACADDRTYPE
vnic0       etherstub0      0 Mbps  2:8:20:c2:39:38    random
vnic1       etherstub0      0 Mbps  2:8:20:45:8f:c9    random
vnic2       etherstub0      0 Mbps  2:8:20:6b:8:ab     random

# dladm show-link
LINK        CLASS    MTU    STATE    OVER
e1000g0     phys     1500   up       --
vnic0       vnic     9000   up       etherstub0
vnic1       vnic     9000   up       etherstub0
vnic2       vnic     9000   up       etherstub0
```

At this stage, you configure exclusive IP zones over VNICs, configure them, and assign IP addresses to them, as explained in "Configuring a Basic Virtual Network" on page 176.

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0:flags=201100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4,CoS>mtu 1500 index 2
        inet 192.168.3.70 netmask ffffff00 broadcast 192.168.3.255
        ether 0:14:4f:94:d0:40
lo0: flags=2002000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6,VIRTUAL> mtu 8252 index 1
        inet6 ::1/128

# ifconfig vnic0 plumb
# ifconfig vnic0 192.168.0.250
# ifconfig vnic0 up

# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0:flags=201100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4,CoS>mtu 1500 index 2
        inet 192.168.3.70 netmask ffffff00 broadcast 192.168.3.255
        ether 0:14:4f:94:d0:40
vnic0: flags=201100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4,CoS> mtu 9000 index 5
        inet 192.168.0.250 netmask ffffff00 broadcast 192.168.0.255
        ether 2:8:20:c2:39:38
lo0: flags=2002000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6,VIRTUAL> mtu 8252 index 1
        inet6 ::1/128

# routeadm
              Configuration   Current             Current
                     Option   Configuration       System State
---------------------------------------------------------------
```

```
                IPv4 routing   enabled              enabled
                IPv6 routing   disabled             disabled
              IPv4 forwarding  disabled             disabled
              IPv6 forwarding  disabled             disabled

            Routing services   "route:default ripng:default"

# routeadm -u -e ipv4-forwarding

# cd /etc/ipf
# vi ipnat.conf
map e1000g0 192.168.0.0/24 -> 0/32  portmap tcp/udp auto
map e1000g0 192.168.0.0/24 -> 0/32
# svcadm enable network/ipfilter

# zoneadm -z zone1 boot
# zoneadm -z zone2 boot
```

Next Steps

- Test the connectivity of the private network by using the various observability tasks in Chapter 12, "Administering Virtual Networks and Resource Controls (Tasks)."
- Create a firewall that filters outgoing packets from the private network. For more information, see "Configuring Solaris IP Filter" in *System Administration Guide: IP Services*.

◆ ◆ ◆ CHAPTER 12

Administering Virtual Networks and Resource Controls (Tasks)

This chapter contains monitoring and statistics-gathering tasks for a system that has configured interface-based flow control, or a virtual network, or both. The connectivity monitoring tasks use standard network tools to observe and evaluate activity on the virtual network. The tasks for gathering usage statistics on packet flows and, if applicable, on VNICs, use the `dladm`, `flowadm`, and `acctadm` commands. These last tasks are especially useful for provisioning, consolidation, and billing purposes. The following subjects are discussed:

Monitoring and Statistics–Gathering Task Map

Task	Description	For Instructions
Validate the configuration of the VNIC for a host's global zone.	Use standard tools to check whether the VNIC is working as expected.	"How to Verify the VNIC Configuration for the Global Zone" on page 210
Validate the configuration of the virtual network.	Check whether the virtual network is configured as expected and that the containers can pass traffic both internally and externally.	"How to Verify Configuration of a Virtual Network of Exclusive IP Zones" on page 211
Observe activity on the virtual network to validate its configuration.	Use the `snoop` command to verify that the containers of the private network can pass traffic to each other.	"How to Verify Virtual Network Connectivity by Using the `snoop` Command" on page 214

Task	Description	For Instructions
Obtain statistical information about interfaces and packet flows	View current usage of VNICs and traffic flows	"How to Obtain Statistics on VNICs" on page 216 and "How to Obtain Statistics on Flows" on page 217
Accumulate flow usage statistics for billing purposes	Gather statistical information on traffic flows for billing purposes.	"How to Configure Flow Accounting" on page 218

Verifying Virtual Network Connectivity

You can use standard network tools to verify your virtual network's connectivity. This section contains simple tasks to help you verify that the VNICs of your virtual network are correctly configured and have the expected network connectivity. Following is a list of the tools used in the tasks, along with links to their man pages.

- `dladm(1M)`
- `ifconfig(1M)`
- `netstat(1M)`
- `ping(1M)`

▼ How to Verify the VNIC Configuration for the Global Zone

Before You Begin The following task assumes that you have created a VNIC for the global zone of your system.

1 On the system with the virtual network, become superuser or assume the equivalent root role.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2 Verify the state of the data links on the system.

```
# dladm show-link
```

Your output should resemble either of the following:

- For a system that has a publicly accessible virtual network, such as the network that is configured in "How to Create a Virtual Network Interface" on page 177:

```
# dladm show-link
LINK        CLASS     MTU    STATE    OVER
bge0        phys      1500   up       bge0
vnic0       vnic      9000   up       bge10
```

In this output, both the physical network interface `bge0` and the VNIC pseudo-interface `vnic0` are configured as data links.

- For a system with a private virtual network that cannot be accessed by external users, such as the network that is configured in "How to Create Etherstubs and VNICs for the Private Virtual Network" on page 202:

```
# dladm show-link
LINK        CLASS    MTU    STATE    OVER
e1000g2     phys     1500   unknown  --
e1000g0     phys     1500   up       --
vnic0       vnic     9000   up       etherstub0
vnic1       vnic     9000   up       etherstub0
```

The network interface `e1000g0` is configured as a data link. The presence of `etherstub0` indicates this is a private network. Two VNICs, `vnic0` and `vnic1`, are successfully configured over the etherstub.

3 Verify that the VNIC is plumbed and running on the IP level of the TCP/IP protocol stack:

```
# ifconfig -a
```

You should receive output similar to the following:

```
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
bge0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.8.50 netmask ffffff00 broadcast 192.168.8.255
        ether 8:0:20:c8:f4:1d
vnic0: flags=201000842<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 192.168.8.10 netmask ffffff00 broadcast 192.168.8.255
        ether 2:8:20:54:f4:74
```

Both the network interface `bge0` and the VNIC `vnic0` are plumbed and up.

How to Verify Configuration of a Virtual Network of Exclusive IP Zones

Before You Begin The procedure assumes that you have created at least two VNICs and corresponding exclusive IP zones to form a virtual network. You also have configured and plumbed these VNICs while logged into their respective zones. The next task verifies the configuration of the virtual network created in "Basic Virtual Network on a Single System" on page 166.

1 On the system where you create the virtual network, become superuser or assume the equivalent root role in the global zone.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2 Ensure that the VNICs are configured as data links in the global zone.

```
# dladm show-vnic
```

You should receive output similar to the following:

```
LINK        OVER              SPEED  MACADDRESS         MACADDRTYPE
vnic1       e1000g0      1000 Mbps  2:8:20:5f:84:ff    random
vnic2       e1000g0      1000 Mbps  2:8:20:54:f4:74    random
```

In this example, both VNICs of the virtual network are configured as data links over network interface `e1000g0`.

3 Verify that any interfaces known to the global zone are plumbed and up.

```
# ifconfig -a
lo0: flags=2001000849<UP,UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 192.168.3.70 netmask ffffff00 broadcast 192.168.83.255
        ether 0:14:4f:94:d0:40
```

Only the network interface `e1000g0` is plumbed for the global zone. This interface has the IP address `192.168.3.70` and connects the system to the external `192.168.3.0/24` network. For the virtual network configuration, `ifconfig -a` in the global zone should not report any VNICs.

4 Check the state of the configured zones.

```
# zoneadm list -v
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   5 zone2            running    /export/home/zone2             native   excl
   7 zone1            running    /export/home/zone1             native   excl
```

The STATUS column indicates that the zones are up and running. If the status of the zones indicates a condition other than "running," you need to reboot the zone. For instructions, refer to Chapter 20, "Installing, Booting, Halting, Uninstalling, and Cloning Non-Global Zones (Tasks)," in *System Administration Guide: Virtualization Using the Solaris Operating System*.

5 Check the global zone's known routes.

```
# netstat -rn
```

You should receive output similar to the following:

```
Routing Table: IPv4
  Destination           Gateway           Flags  Ref     Use     Interface
-------------------- -------------------- ----- ----- ---------- ---------
default              192.168.3.1          UG        1        8    e1000g0
192.168.3.0          192.168.3.70         U         1      143    e1000g0
127.0.0.1            127.0.0.1            UH        1       13    lo0
```

```
Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use    If
--------------------------- --------------------------- ----- --- ------- -----
::1                         ::1                         UH      1      22 lo0
```

The global zone's default route to external networks is through the gateway `192.168.3.1`. This is the IP address of the default router for network `192.168.3.0/24`. The global zone also reports that the route to the gateway is through `192.168.3.70`, the IP address of the system's `e1000g0` interface.

6 Log in to one of the zones of the virtual network, for example, `zone1`, and ensure that the zone's VNIC is plumbed and up.

```
# zlogin zone1
# ifconfig -a vnic1
vnic1: flags=201000842<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
          inet 192.168.3.20 netmask ffffff00 broadcast 192.168.3.255
          ether 2:8:20:54:f4:74
```

7 Check the known routes between the local zone and the external network.

```
#  netstat -rn
```

You should receive output similar to the following:

```
Routing Table: IPv4
  Destination           Gateway           Flags  Ref     Use     Interface
-------------------- -------------------- ----- ----- ---------- ---------
default              192.168.3.1          UG       1        0     vnic1
192.168.3.0          192.168.3.20         U        1        2     vnic1
127.0.0.1            127.0.0.1            UH       1       23     lo0
```

The output verifies that the default route for `zone1` is to the default router, `192.168.3.1`. `zone1` also knows to route packets through `vnic1`, `192.168.3.20`. This traffic is then passed to the global zone, where the packets travel through the network interface `e1000g0`.

8 Verify the VNICs' connectivity.

Perform these steps while logged into a local zone. The following steps assume that you are logged into `zone1`.

a. Check the connectivity between the local zone's VNIC and the system's network interface.

`# ping` *network-interface-address*

For example, check that `vnic1` can pass traffic to network interface `e1000g0`, IP address `192.168.3.70`.

```
# ping 192.168.3.70
192.168.3.70 is alive
```

b. Check that the VNIC can pass traffic through the default router, IP address `192.168.3.1`.

```
# ping 192.168.3.1
192.168.3.1 is alive
```

c. Check that the VNIC can pass traffic to another VNIC in the virtual network.

```
# ping vnic-IP-address
```

For example, to check that `vnic1` can pass traffic to `vnic2` (IP address`192.168.3.22`), run the following command.

```
# ping 192.168.3.22
192.168.3.22 is alive
```

Next Steps

- If Steps 5–7 complete successfully in one exclusive IP zone, then repeat them for each exclusive IP zone in the virtual network.
- To observe packet flows and take statistics, go on to the procedure "Observing Traffic on Virtual Networks" on page 214.

Observing Traffic on Virtual Networks

Use the standard `snoop` command to observe and analyze the status of the virtual network. `snoop` gathers packets and displays their output, enabling you to observe and analyze their content. You can use `snoop` output to verify the connectivity by observing the "conversation" among the VNICs on a virtual network. For full details on `snoop` usage, refer to the `snoop(1M)` man page.

▼ How to Verify Virtual Network Connectivity by Using the `snoop` Command

The following task observes traffic on the private network configured in Example 11–7. However, you can use `snoop` to observe traffic over a publicly-accessible virtual network, as well.

1 On the system where you create the private virtual network, become superuser or assume the equivalent root role in the global zone.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2 Gather information about network traffic on the private virtual network.

```
# snoop -d etherstub0
```

By "snooping" on the private network's etherstub, you can obtain information about activities on all the VNICs configured over that etherstub. You can also snoop the individual VNICs.

3 Check the `snoop` output to verify connectivity among the VNICs of the etherstub.

```
Using device etherstub0 (promiscuous mode)
192.168.0.250 -> 192.168.0.200 RIP R (10 destinations)
192.168.0.250 -> 192.168.0.200 RIP R (10 destinations)
192.168.0.250 -> 224.0.0.1    ICMP Router advertisement
  (Lifetime 1800s [1]: {192.168.0.250 0})
192.168.0.250 -> 192.168.0.200 RIP R (10 destinations)
192.168.0.250 -> 192.168.0.200 RIP R (10 destinations)
192.168.0.220 -> (broadcast)  ARP C Who is 192.168.0.250, 192.168.0.250 ?
192.168.0.200-> (broadcast)  ARP C Who is 192.168.0.220, 192.168.0.220 ?
192.168.0.250 -> (broadcast)  ARP C Who is 192.168.0.200, 192.168.0.200 ?
192.168.0.200 -> 192.168.0.220 ARP R 192.168.0.200, 192.168.0.200 is 2:8:20:45:8f:c9
192.168.0.250 -> 192.168.0.200 ICMP Echo request (ID: 20291 Sequence number: 0)
192.168.0.200 -> (broadcast)  ARP C Who is 192.168.0.250, 192.168.0.250 ?
192.168.0.250 -> 192.168.0.200 ARP R 192.168.0.250, 192.168.0.250 is 2:8:20:c2:39:38
192.168.0.200 -> 192.168.0.250 ICMP Echo reply (ID: 20291 Sequence number: 0)
192.168.0.250 -> 192.168.0.250 RIP R (10 destinations)
```

This output shows the contents of packets that are exchanged among the three VNICs in Figure 10–2 as they contact each other.

- `etherstub0`, over `vnic0` (IP address `192.168.0.250`) sends out RIP routing protocol packets to `vnic1` (`192.168.0.200`).
- `vnic2` (`192.168.0.220`) sends out an ARP broadcast message ("Who is"), to `vnic0` (`192.168.0.250`). Then, `vnic0` sends out an ARP request to `vnic1`:

  ```
  192.168.0.220 -> (broadcast)  ARP C Who is 192.168.0.250, 192.168.0.250 ?
  192.168.0.250 -> (broadcast)  ARP C Who is 192.168.0.200, 192.168.0.200 ?
  ```

- `vnic1` (`192.168.0.200`) sends out an ARP broadcast message ("Who is") to `vnic2` (`192.168.0.220`).
- Eventually `vnic1` responds to the `vnic2`'s "Who is" broadcast by sending its MAC address, as follows:

  ```
  192.168.0.200 -> 192.168.0.220
  ARP R 192.168.0.200, 192.168.0.200 is 2:8:20:45:8f:c9
  ```

  This response proves that the two VNICs of the virtual network, `vnic1` and `vnic2`, can send packet traffic to each other.
- Then `vnic0` responds to `vnic1`'s ARP request with its MAC address:

  ```
  192.168.0.250 -> 192.168.0.200
   ARP R 192.168.0.250, 192.168.0.250 is 2:8:20:c2:39:38
  ```

  This response proves that `etherstub0` on `vnic0` and `vnic1` on the virtual network can send traffic to each other.

The previous output is but a sample of the many ways to use `snoop` for diagnostic purposes. For more information, refer to the `snoop`(1M) man page.

Gathering Usage Statistics for VNICs and Flows

Network virtualization and resource control introduces tools for observing traffic on a per-interface or per-flow basis. The next tasks show how to use options of the `dladm` and `flowadm` commands to obtain statistics on packet traffic. For full details on each command, refer to the following man pages:

- `dladm`(1M)
- `flowadm`(1M)

▼ How to Obtain Statistics on VNICs

VNIC statistics are useful for provisioning purposes, such as determining whether a system needs additional VNICs or possibly additional interfaces. You can also use VNIC traffic statistics to determine how well network consolidation onto a virtual network is working.

Before You Begin The task assumes that you have a working virtual network, such as the network configured in "Configuring a Basic Virtual Network" on page 176.

1 On the system where you create the virtual network, become superuser or assume the equivalent root role in the global zone.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2 Observe traffic flow over network interfaces

Before checking the flow usage on individual VNICs, you might want to view overall usage of the VNIC's underlying interface.

where *link-name* is the name of a currently plumbed interface, for example `internal0`.

```
# dladm show-link -s -i 5 internal0
LINK        IPACKETS  RBYTES      IERRORS  OPACKETS  OBYTES      OERRORS
internal0   5315127   400738164         0   169526   29260024         0
LINK        IPACKETS  RBYTES      IERRORS  OPACKETS  OBYTES      OERRORS
internal0         1          60         0         2        340         0
LINK        IPACKETS  RBYTES      IERRORS  OPACKETS  OBYTES      OERRORS
internal0        17        1020         0         4        456         0
^C
```

To halt the display, press `Ctrl-C`.

3 Observe traffic flow over configured VNICs.

Use the following syntax for each VNIC in the virtual network. You must run this command in the global zone for all VNICs configured on the system.

```
#  dladm show-link -s -i 5 vnic-link-name
```

where *vnic-name* is the name of the VNIC whose traffic you want to observe.

You should receive output similar to the following:

```
# dladm show-link -s -i 5 vnic0
            ipackets  rbytes      ierrors   opackets      obytes      oerrors
vnic0       537001    48104701    0         5             210         0
            ipackets  rbytes      ierrors   opackets      obytes      oerrors
vnic0       3         270         0         0             0           0
^C
```

The output indicates that `vnic0` has had both incoming packet (`ipackets`) and outgoing packet (`opackets`) traffic.

To halt the display, press `Ctrl-C`.

How to Obtain Statistics on Flows

Flow statistics help you evaluate packet traffic on your network before assigning bandwidth and priorities to already configured flows. Like VNIC statistical information, flow statistics are also useful for provisioning and, possibly, for billing purposes.

Before You Begin This procedure assumes that you have configured flows, as described in Chapter 13, "Configuring Resource Management on an Interface."

1 On the system where you configure flow control, become superuser or assume the equivalent root role in the global zone.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2 Observe packet flow statistics for a system that is configured with interface-based flow control.

The following example shows output for a system with two network interfaces that uses traditional bandwidth control. The system does not have a virtual network configured.

```
# flowadm show-flow -s
FLOW        IPACKETS  RBYTES      IERRORS OPACKETS  OBYTES      OERRORS
net20       0         0           0       1723      72366       0
httpsflow   0         0           0       32932     3225851     0
httpflow    29        3982        0       42        20799       0
```

where

`net20`	is a flow for all UDP traffic across interface `10.10.3.20/24` on the internal network.
`httpsflow`	is a flow for all secure HTTP traffic over interface `192.168.3.25` , which is connected to the external network
`httpflow`	is a flow for all HTTP traffic over interface `192.168.3.25`.

3 To halt the display, press Ctrl-C.

Setting Up Accounting for Packet Flows

You can use the extended accounting feature to capture these statistics into a file, which can then be printed for usage reports.

▼ How to Configure Flow Accounting

Use the `flowadm` command and extended accounting feature of the Solaris OS to collect statistical data on flow usage. In this manner, you can maintain records of flow traffic for tracking, provisioning, or billing purposes. The following task shows how to gather statistics for a system on a traditional network, which has implemented flow control on the system's interfaces. For an example, refer to Figure 10–3. For the tasks that implement flow control on this network, see "How to Set Up and Configure Flow Control for a System on a Traditional Network" on page 225.

Before You Begin You must have configured flows on the system's interfaces, as explained in "Interface-Based Flow Control for Traditional Networks" on page 225.

1 Assume the Primary Administrator role or become superuser on the system with the interfaces whose usage you want to track.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 On the system with the interfaces whose usage you want to track, become superuser or assume the equivalent root role.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

3 View the system's currently configured flows.

```
# flowadm show-flow
NAME            LINK            ATTR            VALUE
app-flow        internal0
                                ip_version      4
                                local_ip        10.10.12.45/
httpsflow       nge0
                                transport       tcp
                                local_port      443
httpflow        nge0
                                transport       tcp
                                local_port      80
```

The output shows three flows. `app-flow` is a flow configured on interface `internal0`. This interface isolates traffic coming from the application server on the network, by using the IP address `10.10.12.45/` of the server as the key. The flows `httpflow` and `httpsflow` are configured on interface `nge0`. These flows isolate HTTP and secure HTTP packets by using their port numbers, 80 and 443, respectively.

4 View usage statistics for the system's three flows.

```
# flowadm show-flow -s

FLOW        IPACKETS  RBYTES      IERRORS OPACKETS  OBYTES      OERRORS
app-flow    65        0           0       1723      72366       0
httpsflow   0         0           0       32932     3225851     0
httpflow    29        3982        0       42        20799       0
```

The `-s` option of `flowadm` gathers usage statistics at the point when the command is issued.

5 View the accounting options that are available from extended accounting.

```
# acctadm
           Task accounting: inactive
      Task accounting file: none
    Tracked task resources: none
  Untracked task resources: extended
         Process accounting: inactive
   Process accounting file: none
 Tracked process resources: none
Untracked process resources: extended,host
           Flow accounting: inactive
      Flow accounting file: none
    Tracked flow resources: none
  Untracked flow resources: extended
           Network accounting: inactive
      Network accounting file: none
    Tracked Network resources: none
  Untracked Network resources: extended
```

The last four options implement extended accounting for flows. The previous output verifies that network accounting is not turned on and an accounting file is unknown.

6 Confirm that an extended accounting file does not already exist.

```
# cd /var/adm/exacct ; ls
```

If no accounting file is listed, this confirms that you can now create the file.

7 Create an extended accounting file for the flows on the system.

Use the syntax:

```
# acctadm -e extended -f /var/adm/exacct/file-name net
```

The `net` argument turns on extended accounting for network flows.

Suppose you wanted to track usage of the flows you have defined for a system. You would type the following:

```
# acctadm -e extended -f /var/adm/exacct/httpflow net
```

8 Verify that extended accounting is now turned on.

```
# acctadm
           Task accounting: inactive
           .
           .
           Network accounting: active
           Network accounting file: /var/adm/exacct/httpflow
           Tracked Network resources: extended
           Untracked Network resources: none
```

9 Show usage of all the system's flows.

```
# flowadm show-usage -f /var/adm/exacct/httpflow

Bytes          Packets   Errors    Duration       Bandwidth      Link/Flow
_____________________________________________________________________________
68912          76        0         140              3.94 Kbps        httpflow
3449           32        0         140            197.09 bps       httpsflow
0              0         0         140              0.00 bps        app-flow
```

10 Show the contents of the extended accounting file that have to do with a particular flow.

Use the following syntax:

```
# flowadm show-usage -f /var/adm/exacct/filename flow
```

For example, use the following command to obtain data on HTTP usage on the system.

```
# flowadm show-usage -f /var/adm/exacct/httpflow httpflow
14:35:51    14:36:11    0               0                 0.00 bps  httpflow
14:36:11    14:36:31    0               0                 0.00 bps  httpflow
14:36:31    14:36:51    3789            64895            27.47 Kbps  httpflow
14:36:51    14:37:11    120             108              91.20 bps  httpflow
14:37:11    14:37:31    0               0                 0.00 bps  httpflow
```

This output gathers statistics on `httpflow` since the command was issued.

11 Obtain a count of all packet flows on the system.

Gathering statistics on interface usage can indicate over time whether you should configure a virtual network on the interface or possibly add another interface.

```
# dladm show-usage -f /var/adm/exacct/httpflow
Bytes           Packets   Errors    Duration       Bandwidth      Link/Flow

_______________________________________________________________________________
159080          1188      0         400             3.18 Kbps            nge0
1600            17        0         400            32.00 bps             internal0
```

The output shows packet count for the `nge0` interface, which has flows for HTTP and HTTPS traffic, as well as unfiltered traffic. The `internal0` interface has a flow for traffic received from IP address `10.10.12.45`, the address of the application server.

12 Obtain a count of all packet traffic for a specific interface.

```
# dladm show-usage -f /var/adm/exacct/httpflow nge0
Start Time  End Time    In Bytes       Out Bytes      Bandwidth     Device

_______________________________________________________________________________
14:34:51    14:35:11    2512           0                1.00 Kbps     nge0
14:35:11    14:35:31    986            0              394.40 bps      nge0
14:35:31    14:35:51    2108           0              843.20 bps      nge0
14:35:51    14:36:11    2632           0                1.05 Kbps     nge0
14:36:11    14:36:31    1653           91             697.60 bps      nge0
14:36:31    14:36:51    5237           64937           28.07 Kbps     nge0
14:36:51    14:37:11    8902           3424             4.93 Kbps     nge0
14:37:11    14:37:31    9648           4958             5.84 Kbps     nge0
14:37:31    14:37:51    1766           0              706.40 bps      nge0
14:37:51    14:38:11    3334           0                1.33 Kbps     nge0
14:38:11    14:38:31    1834           578            964.80 bps      nge0
```

◆ ◆ ◆ CHAPTER 13

Configuring Resource Management on an Interface

This chapter explains how to set up resource management on a per-interface basis. This capability improves system and network performance for many types of network configurations, from corporate intranetworks, to local area networks, to virtual networks on a single system.

The chapter covers the following topics:

Resource Management Task Map

Task	Description	For Instructions
Manage traffic flows on a system to improve performance, efficiency, and observability	Isolate network traffic into individual flows. Then assign the flows a set amount of interface bandwidth and a priority among other flows.	"How to Set Up and Configure Flow Control for a System on a Traditional Network" on page 225
Manage traffic flows on a virtual network for system efficiency and for providing different types of services to each container of the virtual network.	Isolate the traffic on each container into individual flows. Assign traffic flows on each container a differing amount of bandwidth and priority, possibly to establish service level agreements.	To come

How Resource Management Works

With interface-based resource management, you can isolate, prioritize, track, and control data traffic on an individual system. These features also enable you to improve the efficiency and performance of a network. Isolating types of data traffic is especially helpful for network provisioning, establishing service level agreements, billing clients, and diagnosing security problems. The concepts in this section pertain to traffic on an internal virtual network as well to traffic on systems configured in traditional external networks.

Resource control helps to isolate processes to improve a system's efficiency and to make the processes easier to observe and track for accounting purposes. Configuring resource control involves organizing packet traffic on the interface into *flows* that have the same characteristics. These characteristics are derived from the information contained in the fields of an individual packet's header. Therefore, you can organize packet traffic into flows by one of the following characteristics:

- IP address
- Transport protocol name (UDP, TCP, or SCTP)
- Application port number, for example, port 21 for FTP
- DS field attribute, which is used for quality-of-service in IPv6 packets only. (For more information about the DS field, refer to "DS Codepoint" in *System Administration Guide: IP Services*.)

Note that a flow can be based only on one of the previously listed characteristics.

For example, you can create a flow for only FTP packets or only for all packets received from a particular source IP address. You cannot create a flow for packets from port number 21 (FTP) that come only from a specified IP address. Or you cannot create a flow for all traffic from IP address `192.168.1.10`, and then create flows for transport layer traffic on `192.168.1.10`.

Bandwidth management involves assigning a portion of the interface's bandwidth to each flow. Modern network interfaces, such as GLD.v3 interfaces `e1000g`, `bge`, `nge`, and others, have large amounts of bandwidth available for assignment to flows. When you create a flow, you can allocate bandwidth to it and then give the flow a relative priority among all flows on the interface. Furthermore, if the system has processor sets, you can assign a CPU processor set to a flow.

The resulting set of rules that define the characteristics of all flows on a system make up the system's *flow control policy*. You implement these rules by using the `flowadm` command and its `set-flowprop` subcommand. For complete technical information, refer to the `flowadm(1M)` man page.

Note – This chapter uses the term *flow control* to generically refer to both resource control and bandwidth management, unless the text specifically states otherwise.

Interface-Based Flow Control for Traditional Networks

This section shows how to set up flow control for a heavily used system on a traditional network. A proxy server in a small network is used as an example. However, the same generic procedures apply for configuring flow control on the interface of any system on your network.

The steps use the scenario shown in "Interface-based Resource Control for a Traditional Network" on page 171. This topology is typical of the networks in use at colleges or small businesses that host their own services and do not outsource their applications. The scenario is illustrated in Figure 10–3.

During the next task, you create a flow control policy to control traffic over both of the proxy server's interfaces. The task shows how to improve the proxy server's efficiency on its public side by doing the following for traffic over `DMZ0`:

- Creating two flows to isolate web traffic, one flow for HTTP packets and a second flow for secure HTTPS packets.
- Assigning a specific amount of the interface's bandwidth to each flow
- Prioritizing the two flows in comparison to other types of packet traffic.

The task then shows how to improve proxy server performance on the internal network by doing the following for traffic over `internal1`:

- Creating separate flows for the application server, database server, and backup server.
- Using the IP address of each server as the identifier for packet traffic from the proxy to the application server.
- Not creating flows for traffic from the individual systems on subnet `10.10.12.0/24`

You use the `flowadm` and `dladm` commands throughout the procedure. For detailed technical information about these commands, refer to the `dladm`(1M) and the `flowadm`(1M) man pages.

▼ How to Set Up and Configure Flow Control for a System on a Traditional Network

Before You Begin This procedure assumes that you are using a system with at least two interfaces. However, you can use the `flowadm` command as shown in the next steps to configure flow control for a single-interface host.

The procedure assumes that the names of the interface `nge0` and `nge1` have already been changed to `DMZ0` and`internal0`, respectively.

Note – You do not have to change the interface device names to use the steps in this procedure. However, many sites might prefer to create their own link names to provide more clarity for their configurations. To change the device name of an interface, refer to "How to Rename a Data Link" on page 47.

1 On the system where you set up flow control, become superuser or assume the equivalent root role.

To create and assign the root role, see "How to Make root User Into a Role" in *System Administration Guide: Security Services*.

2 Check the status of the links.

```
# dladm show-link
LINK        CLASS     MTU    STATE    OVER
internal0   phys      1500   up       --
e1000g0     phys      1500   unknown  --
e1000g1     phys      1500   unknown  --
DMZ0        phys      1500   up       --
```

Note that the nge interfaces are now listed by their new link names.

3 Verify that the interfaces are plumbed and up on the IP layer.

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
         inet 127.0.0.1 netmask ff000000
DMZ0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 10.10.6.5 netmask ff000000 broadcast 10.255.255.255
        ether 0:14:4f:94:d0:60
internal0: flags=201000849<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 10.10.12.42 netmask ffffff00 broadcast 10.10.10.255
        ether 0:10:20:30:40:aa
```

The output verifies that the interfaces are plumbed and listed by their link names.

4 Create a flow to isolate traffic by the port number of the application.

Specify the port number and the associated transport layer protocol of the application to the `flowadm` command. Use the following syntax:

```
# flowadm add-flow -l link-name -a transport=name,port-number flow-name
```

The `-l` option is followed by the link name. The `-a` option is followed by the attributes of the flow that you want to configure. Use the following syntax to specify the attributes of a flow that is defined by the port number of the application:

`transport=`*name* — Name of the appropriate transport layer protocol that is used in conjunction with the application's

	port number. Possible values include TCP, UDP, SCTP, ICMP, and ICMPv6.
local_port | remote_port=*port-number*	Port number of the application whose packets you want to isolate into a flow. You also must indicate whether the packets are flowing through the system's local port or arriving from a remote system's port . To find out the port number of an application, consult the `/etc/services` file, as explained in the `services(4)` man page.
flow-name	Name that you create for the flow.

For example, use the following commands to create flows for the `DMZ0` link that is shown in Figure 10–3:

```
# flowadm add-flow -l DMZ0 -a transport=tcp,local_port=80 httpflow
# flowadm add-flow -l DMZ0 -a transport=tcp,local_port=443 httpsflow
```

The flow `httpflow` isolates traffic for the HTTP application, which runs over the standard port 80 on the proxy server. The flow `httpsflow` isolates traffic for the HTTPS application, which runs over the standard port 443.

5 Verify the status of the flows on a link.

Use the following syntax for the `show-flow` subcommand of `flowadm`:

`flowadm show-flow -l` *link-name*

The next example shows how to display the status of the `DMZ0` link.

```
# flowadm show-flow -l DMZ0
NAME            LINK            ATTR            VALUE
httpsflow       DMZ0
                                ip_version      4
                                transport       tcp
                                local_port      443
httpflow        DMZ0
                                ip_version      4
                                transport       tcp
                                local_port      80
```

6 Add bandwidth controls to a flow.

Use the following syntax of the `show-flowprop` subcommand of `flowadm`:

`flowadm set-flowprop -p` *flow-properties flow-name*

You can specify any of the following for *flow-properties*:

`maxbw`	Maximum amount of bandwidth that packets in this flow can use on the link.

`priority` Amount of priority to give to packets in this flow, in relation to other flows on the link. The possible values are `high`, `normal`, or `low`.

`cpus` Allocate packets of the flow to a processor set, for systems that have multiple processor sets.

For example, for the flows on the `DMZ0` link, you might set the following bandwidths:

```
# flowadm set-flowprop -p maxbw=100 httpflow
# flowadm set-flowprop -p maxbw=100 httpsflow
```

These flows set a bandwidth limit on both the open and secure HTTP services on link `DMZ0` of the proxy server shown in Figure 10–3.

7 Create flows to isolate traffic from a host by IP address.

Use the following syntax:

flowadm add-flow -l *link-name* **-a local_ip | remote_ip**=*IP address flow-name*

For example, in Figure 10–3, the proxy server acts as a proxy for three servers on internal network `10.10.12.0/24`. This system also forwards packets for users on network `10.10.12.0/24`. To create flows to separate the traffic from the three servers, you would do the following:

```
flowadm add-flow -l internal0 -a local_ip=10.10.12.45 app-flow
flowadm add-flow -l internal0 -a local_ip=10.10.12.46 db-flow
flowadm add-flow -l internal0 -a local_ip=10.10.12.47 backup-flow
```

You do not need to create a separate flow for user packet traffic.

8 Set priorities for a flow.

When you set priorities, flows are assigned importance in comparison to each other. The three priority settings are `high`, `normal`, and `low`. You use the `set-flowprop` subcommand of `flowadm`, as shown in Step 7, to set flow priority.

For example, you might set priorities for the flows from the three servers on network `10.10.12.0/24` in Figure 10–3 as follows:

- Give the application server a medium amount of bandwidth and high priority.
- Give the database server slightly less bandwidth and high priority.
- Give the backup server a narrow bandwidth and low priority.

On a system with processor sets, you can also assign cpus to flows, as shown in the syntax in Step 7. You can isolate a flow further by assigning to it a specified amount of cpus. For example, the proxy server in Figure 10–3 has 16 processor sets, which you can assign to particular flows.

Here are the `set-flowprop` policies for the internal servers shown in Figure 10–3

```
# flowadm set-flowprop -p maxbw=100,priority=high,cpus=2 app-flow
# flowadm set-flowprop -p maxbw=100,priority=high,cpus=3 db-flow
# flowadm set-flowprop -p maxbw=10,priority=low,cpus=4 backup-flow
```

9 Verify that the flows now exist on the system's interfaces.

```
# flowadm show-flow
NAME            LINK            ATTR            VALUE
app-flow        internal0
                                ip_version      4
                                local_ip        10.10.12.45/
db-flow         internal0
                                ip_version      4
                                local_ip        10.10.12.46/
backup-flow    internal0
                                ip_version      4
                                local_ip        10.10.12.47/

httpsflow       DMZ0
                                ip_version      4
                                transport       tcp
                                local_port      443
httpflow        DMZ0
                                ip_version      4
                                transport       tcp
                                local_port      80
```

Example 13–1 Setting Up Traditional Flow Control on an Interface

This example contains the steps for setting flow control on the interfaces of the proxy server that is shown in "Interface-based Resource Control for a Traditional Network" on page 171. Five flows are created with the following characteristics:

`httpflow` Created on link `DMZ0` over interface `nge0`. This flow contains HTTP packets, which flow through the local port 80.

`httpsflow` Created on link `DMZ0` over interface `nge0`. This flow contains secure HTTP packets, which flow through the local port 443.

`app-flow` Created on link `internal0` over interface `nge1`. This flow contains all traffic from IP address `10.10.12.45`, which is the address of an application server on the network.

`db-flow` Created on link `internal0` over interface `nge1`. This flow contains all traffic from IP address `10.10.12.46`, which is the address of a database server on the network.

`backup-flow` Created on link `internal0` over interface `nge1`. This flow contains all traffic from IP address `10.10.12.47`, which is the address of a backup server on the network.

```
# dladm show-phys
LINK         MEDIA                STATE      SPEED DUPLEX   DEVICE
nge1         Ethernet             up         1000 full      nge1
e1000g0      n                    unknown    0    half      e1000g0
e1000g1      n                    unknown    0    half      e1000g1
nge0         Ethernet             up         1000 full      nge0
# dladm show-link
LINK        CLASS    MTU    STATE    OVER
internal0   phys     1500   up       --
e1000g0     phys     1500   unknown  --
e1000g1     phys     1500   unknown  --
DMZ0        phys     1500   up       --
#ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
         inet 127.0.0.1 netmask ff000000
DMZ0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 10.10.6.5 netmask ff000000 broadcast 10.255.255.255
        ether 0:14:4f:94:d0:60
internal0: flags=201000849<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 10.10.12.42 netmask ffffff00 broadcast 10.10.10.255
        ether 0:10:20:30:40:aa
# flowadm add-flow -l DMZ0 -a transport=tcp,local_port=80 httpflow
# flowadm add-flow -l DMZ0 -a transport=tcp,local_port=443 httpsflow
# flowadm set-flowprop -p maxbw=100 httpflow
# flowadm set-flowprop -p maxbw=100 httpsflow
# flowadm add-flow -l internal0 -a local_ip=10.10.12.45 app-flow
# flowadm add-flow -l internal0 -a local_ip=10.10.12.46 db-flow
# flowadm add-flow -l internal0 -a local_ip=10.10.12.47 backup-flow
# flowadm set-flowprop -p maxbw=100,priority=high,cpus=2 app-flow
# flowadm set-flowprop -p maxbw=100,priority=high,cpus=3 db-flow
# flowadm set-flowprop -p maxbw=10,priority=low,cpus=4 backup-flow
# flowadm show-flow
NAME            LINK            ATTR            VALUE
app-flow        internal0
                                ip_version      4
                                local_ip        10.10.12.45/
db-flow         internal0
                                ip_version      4
                                local_ip        10.10.12.46/
backup-flow    internal0
                                ip_version      4
                                local_ip        10.10.12.47/
```

```
httpsflow       DMZ0
                                ip_version      4
                                transport       tcp
                                local_port      443
httpflow        DMZ0
                                ip_version      4
                                transport       tcp
                                local_port      80
```

See Also

- To observe flow traffic on a network, refer to "How to Verify Virtual Network Connectivity by Using the snoop Command" on page 214 (flow control specific information to come)
- To gathered statistics for flows for accounting purposes, refer to "Gathering Usage Statistics for VNICs and Flows" on page 216.

Glossary

This glossary contains definitions of new terms in this book that are not in the *Sun Global Glossary* available from the `docs.sun.com` web site.

3DES See Triple-DES.

AES Advanced Encryption Standard. A symmetric 128-bit block data encryption technique. The U.S. government adopted the Rijndael variant of the algorithm as its encryption standard in October 2000. AES replaces DES encryption as the government standard.

anycast address An IPv6 address that is assigned to a group of interfaces (typically belonging to different nodes). A packet that is sent to an anycast address is routed to the *nearest* interface having that address. The packet's route is in compliance with the routing protocol's measure of distance.

anycast group A group of interfaces with the same anycast IPv6 address. The Solaris OS implementation of IPv6 does not support the creation of anycast addresses and groups. However, Solaris IPv6 nodes can send traffic to anycast groups.

asymmetric key cryptography An encryption system in which the sender and receiver of a message use different keys to encrypt and decrypt the message. Asymmetric keys are used to establish a secure channel for symmetric key encryption. The Diffie-Hellman protocol is an example of an asymmetric key protocol. Contrast with symmetric key cryptography.

authentication header An extension header that provides authentication and integrity, without confidentiality, to IP datagrams.

autoconfiguration The process where a host automatically configures its IPv6 address from the site prefix and the local MAC address.

bidirectional tunnel A tunnel that can transmit datagrams in both directions.

Blowfish A symmetric block cipher algorithm that takes a variable-length key from 32 bits to 448 bits. Its author, Bruce Schneier, claims that Blowfish is optimized for applications where the key does not change often.

broadcast address IPv4 network addresses with the host portion of the address having all zeroes (10.50.0.0) or all one bits (10.50.255.255). A packet that is sent to a broadcast address from a machine on the local network is delivered to all machines on that network.

CA See certificate authority (CA).

certificate authority (CA) A trusted third-party organization or company that issues digital certificates used to create digital signatures and public-private key pairs. The CA guarantees the identity of the individual who is granted the unique certificate.

certificate revocation list (CRL) A list of public key certificates that have been revoked by a CA. CRLs are stored in the CRL database that is maintained through IKE.

class In IPQoS, a group of network flows that share similar characteristics. You define classes in the IPQoS configuration file.

classless inter-domain routing (CIDR) address An IPv4 address format that is not based on network classes (Class A, B, and C). CIDR addresses are 32 bits in length. They use the standard IPv4 dotted decimal notation format, with the addition of a network prefix. This prefix defines the network number and the network mask.

data address An IP address which can be used as the source or destination address for data. Data addresses are part of an IPMP group and can be used to send and receive traffic on any interface in the group. Moreover, the set of data addresses in an IPMP group can be used continuously provided that one interface in the group is functioning.

datagram See IP datagram.

`DEPRECATED` **address** An IP address that cannot be used as the source address for data in an IPMP group. Typically, IPMP test addresses are `DEPRECATED`. However, any address can be marked `DEPRECATED` to prevent the address from being used as a source address.

DES Data Encryption Standard. A symmetric-key encryption method developed in 1975 and standardized by ANSI in 1981 as ANSI X.3.92. DES uses a 56-bit key.

Diffie-Hellman protocol Also known as public key cryptography. An asymmetric cryptographic key agreement protocol that was developed by Diffie and Hellman in 1976. The protocol enables two users to exchange a secret key over an insecure medium without any prior secrets. Diffie-Hellman is used by the IKE protocol.

diffserv model Internet Engineering Task Force architectural standard for implementing differentiated services on IP networks. The major modules are classifier, meter, marker, scheduler, and dropper. IPQoS implements the classifier, meter, and marker modules. The diffserv model is described in RFC 2475, *An Architecture for Differentiated Services*.

digital signature A digital code that is attached to an electronically transmitted message that uniquely identifies the sender.

domain of interpretation (DOI) A DOI defines data formats, network traffic exchange types, and conventions for naming security-relevant information. Security policies, cryptographic algorithms, and cryptographic modes are examples of security-relevant information.

DS codepoint (DSCP) A 6-bit value that, when included in the DS field of an IP header, indicates how a packet must be forwarded.

DSA Digital Signature Algorithm. A public key algorithm with a variable key size from 512 to 4096 bits. The U.S. Government standard, DSS, goes up to 1024 bits. DSA relies on SHA-1 for input.

dual stack A TCP/IP protocol stack with both IPv4 and IPv6 at the network layer, with the rest of the stack being identical. When you enable IPv6 during Solaris OS installation, the host receives the dual-stack version of TCP/IP.

dynamic packet filter See stateful packet filter.

dynamic reconfiguration (DR) A feature that allows you to reconfigure a system while the system is running, with little or no impact on ongoing operations. Not all Sun platforms support DR. Some Sun platforms might only support DR of certain types of hardware such as NICs.

encapsulating security payload (ESP) An extension header that provides integrity and confidentiality to datagrams. ESP is one of the five components of the IP Security Architecture (IPsec).

encapsulation The process of a header and payload being placed in the first packet, which is subsequently placed in the second packet's payload.

failure detection The process of detecting when an interface or the path from an interface to an Internet layer device no longer works. IP network multipathing (IPMP) includes two types of failure detection: link based (default) and probe based (optional).

filter A set of rules that define the characteristics of a class in the IPQoS configuration file. The IPQoS system selects for processing any traffic flows that conform to the filters in its IPQoS configuration file. See packet filter.

firewall Any device or software that isolates an organization's private network or intranet from the Internet, thus protecting it from external intrusions. A firewall can include packet filtering, proxy servers, and NAT (network address translation).

flow accounting In IPQoS, the process of accumulating and recording information about traffic flows. You establish flow accounting by defining parameters for the `flowacct` module in the IPQoS configuration file.

hash value A number that is generated from a string of text. Hash functions are used to ensure that transmitted messages have not been tampered with. MD5 and SHA-1 are examples of one-way hash functions.

header See IP header.

HMAC Keyed hashing method for message authentication. HMAC is a secret key authentication algorithm. HMAC is used with an iterative cryptographic hash function, such as MD5 or SHA-1, in combination with a secret shared key. The cryptographic strength of HMAC depends on the properties of the underlying hash function.

hop A measure that is used to identify the number of routers that separate two hosts. If three routers separate a source and destination, the hosts are four hops away from each other.

host A system that does not perform packet forwarding. Upon installation of the Solaris OS, a system becomes a host by default, that is, the system cannot forward packets. A host typically has one physical interface, although it can have multiple interfaces.

ICMP Internet Control Message Protocol. Used to handle errors and exchange control messages.

ICMP echo request packet A packet sent to a machine on the Internet to solicit a response. Such packets are commonly known as "ping" packets.

IKE Internet Key Exchange. IKE automates the provision of authenticated keying material for IPsec security association (SA)s.

Internet Protocol (IP) The method or protocol by which data is sent from one computer to another on the Internet.

IP See Internet Protocol (IP), IPv4, IPv6.

IP datagram A packet of information that is carried over IP. An IP datagram contains a header and data. The header includes the addresses of the source and the destination of the datagram. Other fields in the header help identify and recombine the data with accompanying datagrams at the destination.

IP header Twenty bytes of data that uniquely identify an Internet packet. The header includes source and destination addresses for the packet. An option exists within the header to allow further bytes to be added.

IP in IP encapsulation The mechanism for tunneling IP packets within IP packets.

IP link A communication facility or medium over which nodes can communicate at the link layer. The link layer is the layer immediately below IPv4/IPv6. Examples include Ethernets (simple or bridged) or ATM networks. One or more IPv4 subnet numbers or prefixes are assigned to an IP link. A subnet number or prefix cannot be assigned to more than one IP link. In ATM LANE, an IP link is a single emulated LAN. When you use ARP, the scope of the ARP protocol is a single IP link.

IP stack TCP/IP is frequently referred to as a "stack." This refers to the layers (TCP, IP, and sometimes others) through which all data passes at both client and server ends of a data exchange.

IPMP group IP multipathing group, composed of a set of network interfaces with a set of data addresses that are treated as interchangeable by the system to improve network availability and utilization. The IPMP group, including all its underlying IP interfaces and data addresses, is represented by an IPMP interface.

IPQoS A software feature that provides an implementation of the diffserv model standard, plus flow accounting and 802.1 D marking for virtual LANs. Using IPQoS, you can provide different levels of network services to customers and applications, as defined in the IPQoS configuration file.

IPsec IP security. The security architecture that provides protection for IP datagrams.

IPv4 Internet Protocol, version 4. IPv4 is sometimes referred to as IP. This version supports a 32-bit address space.

IPv6 Internet Protocol, version 6. IPv6 supports a 128-bit address space.

key management The way in which you manage security association (SA)s.

keystore name The name that an administrator gives to the storage area, or keystore, on a network interface card (NIC). The keystore name is also called the token or the token ID.

link layer The layer immediately below IPv4/IPv6.

link-local address In IPv6, a designation that is used for addressing on a single link for purposes such as automatic address configuration. By default, the link-local address is created from the system's MAC address.

load spreading The process of distributing inbound or outbound traffic over a set of interfaces. With load spreading, higher throughput is achieved. Load spreading occurs only when the network traffic is flowing to multiple destinations that use multiple connections. Two types of load spreading exists: inbound load spreading for inbound traffic and outbound load spreading for outbound traffic.

local-use address A unicast address that has only local routability scope (within the subnet or within a subscriber network). This address also can have a local or global uniqueness scope.

marker 1. A module in the diffserv architecture and IPQoS that marks the DS field of an IP packet with a value that indicates how the packet is to be forwarded. In the IPQoS implementation, the marker module is `dscpmk`.

2. A module in the IPQoS implementation that marks the virtual LAN tag of an Ethernet datagram with a user priority value. The user priority value indicates how datagrams are to be forwarded on a network with VLAN devices. This module is called `dlcosmk`.

MD5 An iterative cryptographic hash function that is used for message authentication, including digital signatures. The function was developed in 1991 by Rivest.

message authentication code (MAC) MAC provides assurance of data integrity and authenticates data origin. MAC does not protect against eavesdropping.

meter A module in the diffserv architecture that measures the rate of traffic flow for a particular class. The IPQoS implementation includes two meters, `tokenmt` and `tswtclmt`.

minimal encapsulation An optional form of IPv4 in IPv4 tunneling that can be supported by home agents, foreign agents, and mobile nodes. Minimal encapsulation has 8 or 12 bytes less of overhead than does IP in IP encapsulation.

MTU Maximum Transmission Unit. The size, given in octets, that can be transmitted over a link. For example, the MTU of an Ethernet is 1500 octets.

multicast address An IPv6 address that identifies a group of interfaces in a particular way. A packet that is sent to a multicast address is delivered to all of the interfaces in the group. The IPv6 multicast address has similar functionality to the IPv4 broadcast address.

multihomed host A system that has more than one physical interface and that does not perform packet forwarding. A multihomed host can run routing protocols.

NAT See network address translation.

neighbor advertisement A response to a neighbor solicitation message or the process of a node sending unsolicited neighbor advertisements to announce a link-layer address change.

neighbor discovery An IP mechanism that enables hosts to locate other hosts that reside on an attached link.

neighbor solicitation A solicitation that is sent by a node to determine the link-layer address of a neighbor. A neighbor solicitation also verifies that a neighbor is still reachable by a cached link-layer address.

network address translation NAT. The translation of an IP address used within one network to a different IP address known within another network. Used to limit the number of global IP addresses that are needed.

network interface card (NIC) Network adapter card that is an interface to a network. Some NICs can have multiple physical interfaces, such as the `qfe` card.

node In IPv6, any system that is IPv6-enabled, whether a host or a router.

outcome The action to take as a result of metering traffic. The IPQoS meters have three outcomes, red, yellow, and green, which you define in the IPQoS configuration file.

packet A group of information that is transmitted as a unit over communications lines. Contains an IP header plus a payload.

packet filter A firewall function that can be configured to allow or disallow specified packets through a firewall.

packet header See IP header.

payload The data that is carried in a packet. The payload does not include the header information that is required to get the packet to its destination.

per-hop behavior (PHB) A priority that is assigned to a traffic class. The PHB indicates the precedence which flows of that class have in relation to other traffic classes.

perfect forward secrecy (PFS) In PFS, the key that is used to protect transmission of data is not used to derive additional keys. Also, the source of the key that is used to protect data transmission is never used to derive additional keys.

PFS applies to authenticated key exchange only. See also Diffie-Hellman protocol.

physical interface A system's attachment to a link. This attachment is often implemented as a device driver plus a network interface card (NIC). Some NICs can have multiple points of attachment, for example, `qfe`.

PKI Public Key Infrastructure. A system of digital certificates, Certificate Authorities, and other registration authorities that verify and authenticate the validity of each party involved in an Internet transaction.

plumb The act of opening a device that is associated with a physical interface name. When an interface is plumbed, streams are set up so that the IP protocol can use the device. You use the `ifconfig` command to plumb an interface during a system's current session.

private address An IP address that is not routable through the Internet. Private addresses can used by internal networks on hosts that do not require Internet connectivity. These addresses are defined in Address Allocation for Private Internets (`http://www.ietf.org/rfc/rfc1918.txt?number=1918`) and often referred to as "1918" addresses.

protocol stack See IP stack.

proxy server A server that sits between a client application, such as a Web browser, and another server. Used to filter requests – to prevent access to certain web sites, for instance.

public key cryptography A cryptographic system that uses two different keys. The public key is known to everyone. The private key is known only to the recipient of the message. IKE provides public keys for IPsec.

redirect In a router, to inform a host of a better first-hop node to reach a particular destination.

repair detection The process of detecting when a NIC or the path from the NIC to some layer-3 device starts operating correctly after a failure.

replay attack In IPsec, an attack in which a packet is captured by an intruder. The stored packet then replaces or repeats the original at a later time. To protect against such attacks, a packet can contain a field that increments during the lifetime of the secret key that is protecting the packet.

reverse tunnel A tunnel that starts at the mobile node's care-of address and terminates at the home agent.

router A system that usually has more than one interface, runs routing protocols, and forwards packets. You can configure a system with only one interface as a router if the system is the endpoint of a PPP link.

router advertisement The process of routers advertising their presence together with various link and Internet parameters, either periodically or in response to a router solicitation message.

router discovery The process of hosts locating routers that reside on an attached link.

router solicitation The process of hosts requesting routers to generate router advertisements immediately, rather than at their next scheduled time.

RSA A method for obtaining digital signatures and public key cryptosystems. The method was first described in 1978 by its developers, Rivest, Shamir, and Adleman.

SA See security association (SA).

SADB Security Associations Database. A table that specifies cryptographic keys and cryptographic algorithms. The keys and algorithms are used in the secure transmission of data.

SCTP See streams control transport protocol.

security association (SA) An association that specifies security properties from one host to a second host.

security parameter index (SPI) An integer that specifies the row in the security associations database (SADB) that a receiver should use to decrypt a received packet.

security policy database (SPD) Database that specifies the level of protection to apply to a packet. The SPD filters IP traffic to determine whether a packet should be discarded, should be passed in the clear, or should be protected with IPsec.

selector The element that specifically defines the criteria to be applied to packets of a particular class in order to select that traffic from the network stream. You define selectors in the filter clause of the IPQoS configuration file.

SHA-1 Secure Hashing Algorithm. The algorithm operates on any input length less than 2^{64} to produce a message digest. The SHA-1 algorithm is input to DSA.

site-local-use address A designation that is used for addressing on a single site.

smurf attack To use ICMP echo request packets directed to an IP broadcast address or multiple broadcast addresses from remote locations to create severe network congestion or outages.

sniff To eavesdrop on computer networks – frequently used as part of automated programs to sift information, such as clear-text passwords, off the wire.

SPD See security policy database (SPD).

SPI See security parameter index (SPI).

spoof To gain unauthorized access to a computer by sending a message to it with an IP address indicating that the message is coming from a trusted host. To engage in IP spoofing, a hacker must first use a variety of techniques to find an IP address of a trusted host and then modify the packet headers so that it appears that the packets are coming from that host.

stack See IP stack.

standby A physical interface that is not used to carry data traffic unless some other physical interface has failed.

stateful packet filter A packet filter that can monitor the state of active connections and use the information obtained to determine which network packets to allow through the firewall. By tracking and matching requests and replies, a stateful packet filter can screen for a reply that doesn't match a request.

stateless autoconfiguration The process of a host generating its own IPv6 addresses by combining its MAC address and an IPv6 prefix that is advertised by a local IPv6 router.

stream control transport protocol A transport layer protocol that provides connection-oriented communications in a manner similar to TCP. Additionally, SCTP supports multihoming, in which one of the endpoints of the connection can have more than one IP address.

symmetric key cryptography An encryption system in which the sender and receiver of a message share a single, common key. This common key is used to encrypt and decrypt the message. Symmetric keys are used to encrypt the bulk of data transmission in IPsec. DES is one example of a symmetric key system.

TCP/IP TCP/IP (Transmission Control Protocol/Internet Protocol) is the basic communication language or protocol of the Internet. It can also be used as a communications protocol in a private network (either an intranet or an extranet).

test address An IP address in an IPMP group which must be used as the source or destination address for probes, and must not be used as a source or destination address for data traffic.

Triple-DES Triple-Data Encryption Standard. A symmetric-key encryption method. Triple-DES requires a key length of 168 bits. Triple-DES is also written as 3DES.

tunnel The path that is followed by a datagram while it is encapsulated. See encapsulation.

unicast address An IPv6 address that identifies a single interface of an IPv6-enabled node. The parts of the unicast address are site prefix, subnet ID, and interface ID.

user-priority A 3-bit value that implements class-of-service marks, which define how Ethernet datagrams are forwarded on a network of VLAN devices.

virtual LAN (VLAN) device Network interfaces that provide traffic forwarding at the Ethernet (data link) level of the IP protocol stack.

virtual network A combination of software and hardware network resources and functionality that are administered together as a single software entity. An *internal* virtual network consolidates network resources onto a single system, sometimes referred to as a "network in a box."

virtual network interface (VNIC) A pseudo-interface that provides virtual network connectivity whether or not it is configured on a physical network interface. Containers such as exclusive IP zones or xVM domains are configured above VNICs to form a virtual network.

virtual private network (VPN) A single, secure, logical network that uses tunnels across a public network such as the Internet.

Index

E

F

G

H

I