System Administration Guide: IP Services

Part No: 821–1453–01
November 2010

ORACLE

110425@25097

Contents

Preface

Welcome to System Administration Guide: IP Services for Oracle Solaris. This book is part of a fourteen-volume set that covers a significant part of the Oracle Solaris system administration information. This book assumes that you have already installed Oracle Solaris. You should be ready to configure your network or ready to configure any networking software that is required on your network. Oracle Solaris is part of the Oracle Solaris product family, which also includes the Oracle Solaris Common Desktop Environment (CDE). Oracle Solaris is compliant with AT&T's System V, Release 4 operating system.

Note – This Oracle Solaris release supports systems that use the SPARC and x86 families of processor architectures. The supported systems appear in the Oracle Solaris OS: Hardware Compatibility Lists (`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" relates specifically to 64-bit x86 compatible CPUs.
- "32-bit x86" points out specific 32-bit information about x86 based systems.

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

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, shutting down and booting a system, and managing services
System Administration Guide: Advanced Administration	Terminals and modems, system resources, system processes, and troubleshooting Oracle Solaris software problems
System Administration Guide: Devices and File Systems	Removable media, disks and devices, file systems, and backing up and restoring data

Book Title	Topics
System Administration Guide: IP Services	TCP/IP network administration, IPv4 and IPv6 address administration, DHCP, IPsec, IKE, 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
System Administration Guide: Network Interfaces and Network Virtualization	Networking stack, NIC driver property configuration, NWAM configuration, manual network interface configuration, administration of VLANs and link aggregations, IP networking multipathing (IPMP), WiFi wireless networking configuration, virtual NICs (VNICs), and network resource management
System Administration Guide: Network Services	Web cache servers, time-related services, network file systems (NFS and autofs), mail, SLP, and PPP
System Administration Guide: Printing	Printing topics and tasks, using services, tools, protocols, and technologies to set up and administer printing services and printers
System Administration Guide: Security Services	Auditing, device management, file security, BART, Kerberos services, PAM, Oracle Solaris Cryptographic Framework, privileges, RBAC, SASL, and Oracle Solaris Secure Shell
System Administration Guide: Oracle Solaris Zones, Oracle Solaris 10 Containers, and Resource Management	Resource management features, which enable you to control how applications use available system resources; Oracle Solaris Zones software partitioning technology, which virtualizes operating system services to create an isolated environment for running applications; and Oracle Solaris 10 Containers, which host Oracle Solaris 10 environments running on the Oracle Solaris 11 Express kernel
Oracle Solaris SMB and Windows Interoperability Administration Guide	Oracle Solaris SMB service, which enables you to configure an Oracle Solaris system to make SMB shares available to SMB clients; Oracle Solaris SMB client, which enables you to access SMB shares; and native identity mapping services, which enables you to map user and group identities between Oracle Solaris systems and Windows systems
Oracle 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
Oracle Solaris Trusted Extensions Configuration and Administration	System installation, configuration, and administration that is specific to the Oracle Solaris' Trusted Extensions feature

Related Books

The following trade books are referred to in this book:

- Ferguson, Paul and Geoff Huston. *Quality of Service*. John Wiley & Sons, Inc., 1998.
- Hunt Craig.*TCP/IP Network Administration, 3rd Edition*. O'Reilly, 2002.
- Kilkki, Kalevi. *Differentiated Services for the Internet*. Macmillan Technical Publishing, 1999.
- Stevens, W. Richard. *TCP/IP Illustrated, Volume 1, The Protocols*. Addison Wesley, 1994.

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.

The IP Filter feature of Oracle Solaris is derived from open source IPFilter software. To view license terms, attribution, and copyright statements for IP Filter, the default path is `/usr/lib/ipf/IPFILTER.LICENCE`. If Oracle Solaris has been installed anywhere other than the default, modify the given path to access the file at the installed location.

Documentation, Support, and Training

See the following web sites for additional resources:

- Documentation (`http://docs.sun.com`)
- Support (`http://www.oracle.com/us/support/systems/index.html`)
- Training (`http://education.oracle.com`) – Click the Sun link in the left navigation bar.

Oracle Software Resources

Oracle Technology Network (`http://www.oracle.com/technetwork/index.html`) offers a range of resources related to Oracle software:

- Discuss technical problems and solutions on the Discussion Forums (`http://forums.oracle.com`).
- Get hands-on step-by-step tutorials with Oracle By Example (`http://www.oracle.com/technetwork/tutorials/index.html`).
- Download Sample Code (`http://www.oracle.com/technology/sample_code/index.html`).

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 shells that are included in the Oracle Solaris OS. Note that the default system prompt that is displayed in command examples varies, depending on the Oracle Solaris release.

TABLE P–2 Shell Prompts

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

PART I

TCP/IP Administration

This part contains tasks and conceptual information for configuring, administering, and troubleshooting TCP/IP networks.

◆ ◆ ◆ CHAPTER 1

Planning an IPv4 Addressing Scheme (Tasks)

This chapter describes the issues you must resolve in order to create your network in an organized, cost-effective manner. After you resolve these issues, you can devise a network plan as you configure and administer your network in the future.

Note – For an overview of the TCP/IP protocol suite and its implementation of the Open Systems Interconnection (OSI) model, see Chapter 1, "Oracle Solaris TCP/IP Protocol Suite (Overview)," in *System Administration Guide: IP Services*

This chapter contains the following information:

For tasks for configuring a network, refer to Chapter 4, "Configuring TCP/IP Network Services and IPv4 Addressing (Tasks)."

Network Planning (Task Map)

The following table lists different tasks for configuring the network. The table includes a description of what each task accomplishes and the section in the current documentation where the specific steps to perform the task are detailed.

Task	Description	For Information
1. Plan your hardware requirements and network topology	Determine the types of equipment that you need and the layout of this equipment at your site.	■ For general network topology questions, refer to "Determining the Network Hardware" on page 35. ■ For IPv6 topology planning, refer to "Preparing the Network Topology for IPv6 Support" on page 66. ■ For information about a specific type of equipment, refer to the equipment manufacturer's documentation.
2. Obtain a registered IP address for your network	Your network must have a unique IP address if you plan to communicate outside your local network, for example, over the Internet.	Refer to "Obtaining Your Network's IP Number" on page 37.
3. Devise an IP addressing scheme for your systems, based on your IPv4 network prefix or IPv6 site prefix.	Determine how addresses are to be deployed at your site.	Refer to "Designing an IPv4 Addressing Scheme" on page 38 or refer to "Preparing an IPv6 Addressing Plan" on page 70.
4. Create a list that contains the IP addresses and host names of all machines on your network.	Use the list to build network databases	Refer to "Network Databases" on page 44
5. Determine which name service to use on your network.	Decide whether to use NIS, LDAP, DNS, or the network databases in the local `/etc` directory.	Refer to "Selecting a Name Service and Directory Service" on page 43
6. Establish administrative subdivisions, if appropriate for your network	Decide if your site requires that you divide your network into administrative subdivisions	Refer to "Administrative Subdivisions" on page 45
7. Determine where to place routers in the network design.	If your network is large enough to require routers, create a network topology that supports them.	Refer to "Planning for Routers on Your Network" on page 45
8. If required, design a strategy for subnets.	You might need to create subnets for administering your IP address space or to make more IP addresses available for users.	For IPv4 subnet planning, refer to "What Is Subnetting?" on page 181 For IPv6 subnet planning, refer to "Creating a Numbering Scheme for Subnets" on page 71

Determining the Network Hardware

When you design your network, you must decide what type of network best meets the needs of your organization. Some of the planning decisions you must make involve the following network hardware:

- The network topology, the layout, and connections of the network hardware
- The number of host systems your network can support
- The types of hosts that the network supports
- The types of servers that you might need
- The type of network media to use: Ethernet, Token Ring, FDDI, and so on
- Whether you need bridges or routers extend this media or connect the local network to external networks
- Whether some systems need separately purchased interfaces in addition to their built in interfaces

Based on these factors, you can determine the size of your local area network.

Note – How you plan the network hardware is outside the scope of this manual. For assistance, refer to the manuals that come with your hardware.

Deciding on an IP Addressing Format for Your Network

The number of systems that you expect to support affects how you configure your network. Your organization might require a small network of several dozen standalone systems that are located on one floor of a single building. Alternatively, you might need to set up a network with more than 1,000 systems in several buildings. This setup can require you to further divide your network into subdivisions that are called *subnets*.

When you plan your network addressing scheme, consider the following factors:

- The type of IP address that you want to use: IPv4 or IPv6
- The number of potential systems on your network
- The number of systems that are multihomed or routers, which require an IP address for each interface
- Whether to use private addresses on your network
- Whether to have a DHCP server that manages pools of IPv4 addresses

The worldwide growth of the Internet since 1990 has resulted in a shortage of available IP addresses. To remedy this situation, the Internet Engineering Task Force (IETF) has developed a number of IP addressing alternatives. Types of IP addresses in use today include the following:

If your organization has been assigned more than one IP address for your network or uses subnets, appoint a centralized authority within your organization to assign network IP addresses. That authority should maintain control of a pool of assigned network IP addresses, and assign network, subnet, and host addresses as required. To prevent problems, ensure that duplicate or random network numbers do not exist in your organization.

IPv4 Addresses

These 32-bit addresses are the original IP addressing format that was designed for TCP/IP. Originally, IP networks have three classes, A, B, and C. The *network number* that is assigned to a network reflects this class designation plus 8 or more bits to represent a host. Class-based IPv4 addresses require you to configure a netmask for the network number. Furthermore, to make more addresses available for systems on the local network, these addresses were often divided into subnets.

Today, IP addresses are referred to as *IPv4 addresses*. Although you can no longer obtain class-based IPv4 network numbers from an ISP, many existing networks still have them. For more information about administering IPv4 addresses, refer to "Designing Your IPv4 Addressing Scheme" on page 39.

IPv4 Addresses in CIDR Format

The IETF has developed Classless Inter-Domain Routing (CIDR) addresses as a short to medium term fix for the shortage of IPv4 addresses. In addition, CIDR format was designed as a remedy to the lack of capacity of the global Internet routing tables. An IPv4 address with CIDR notation is 32 bits in length and has the same dotted decimal format. However, CIDR adds a prefix designation after the rightmost byte to define the network portion of the IPv4 address. For more information, refer to "Designing Your CIDR IPv4 Addressing Scheme" on page 40.

DHCP Addresses

The Dynamic Host Configuration Protocol (DHCP) protocol enables a system to receive configuration information from a DHCP server, including an IP address, as part of the booting process. DHCP servers maintain pools of IP address from which to assign addresses to DHCP clients. A site that uses DHCP can use a smaller pool of IP addresses than would be needed if all clients were assigned a permanent IP address. You can set up the DHCP service to manage your site's IP addresses, or a portion of the addresses. For more information, refer to Chapter 11, "About DHCP (Overview)."

IPv6 Addresses

The IETF has deployed 128–bit IPv6 addresses as the long term solution to the shortage of available IPv4 addresses. IPv6 addresses provide greater address space than is available with IPv4. Oracle Solaris supports IPv4 and IPv6 addressing on the same host, through the use of dual-stack TCP/IP. As with IPv4 addresses in CIDR format, IPv6 addresses have no notion of network classes or netmasks. As in CIDR, IPv6 addresses use prefixes to designate the portion of the address that defines the site's network. For an introduction to IPv6, refer to "IPv6 Addressing Overview" on page 54.

Private Addresses and Documentation Prefixes

The IANA has reserved a block of IPv4 addresses and an IPv6 site prefix for use on private networks. You can deploy these addresses on systems within an enterprise network but be aware that packets with private addresses cannot be routed across the Internet. For more information on private addresses, refer to "Using Private IPv4 Addresses" on page 41.

Note – Private IPv4 addresses are also reserved for documentation purposes. The examples in this book use private IPv4 addresses and the reserved IPv6 documentation prefix.

Obtaining Your Network's IP Number

An IPv4 network is defined by a combination of an IPv4 network number plus a network mask, or *netmask*. An IPv6 network is defined by its *site prefix*, and, if subnetted, its *subnet prefix*.

Unless your network plans to be private in perpetuity, your local users most likely need to communicate beyond the local network. Therefore, you must obtain a registered IP number for your network from the appropriate organization before your network can communicate externally. This address becomes the network number for your IPv4 addressing scheme or the site prefix for your IPv6 addressing scheme.

Internet Service Providers provide IP addresses for networks with pricing that is based on different levels of service. Investigate with various ISPs to determine which provides the best service for your network. ISP's typically offer dynamically allocated addresses or static IP addresses to businesses. Some ISPs offer both IPv4 and IPv6 addresses.

If your site is an ISP, you obtain IP address blocks for your customers from the Internet Registry (IR) for your locale. The Internet Assigned Numbers Authority (IANA) is ultimately responsible for delegating registered IP addresses to IRs around the world. Each IR has registration information and templates for the locale that the IR services. For information about the IANA and its IRs, refer to the IANA's IP Address Service page (`http://www.iana.org/ipaddress/ip-addresses.htm`).

Note – Do not arbitrarily assign IP addresses to your network, even if you are not currently attaching the network to external TCP/IP networks. Instead, use private addresses as described in "Using Private IPv4 Addresses" on page 41.

Designing an IPv4 Addressing Scheme

Note – For IPv6 address planning information, refer to "Preparing an IPv6 Addressing Plan" on page 70.

This section gives an overview IPv4 addressing to aid you in designing an IPv4 addressing plan. For information on IPv6 addresses, see "IPv6 Addressing Overview" on page 54. For information on DHCP addresses, see Chapter 11, "About DHCP (Overview)."

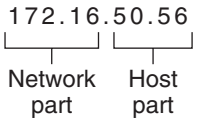
Each IPv4-based network must have the following:

- A unique network number that is assigned by either an ISP, an IR, or, for older networks, registered by the IANA. If you plan to use private addresses, the network numbers you devise must be unique within your organization.
- Unique IPv4 addresses for the interfaces of every system on the network.
- A network mask.

The IPv4 address is a 32-bit number that uniquely identifies a network interface on a system, as explained in "How IP Addresses Apply to Network Interfaces" on page 42. An IPv4 address is written in decimal digits, divided into four 8-bit fields that are separated by periods. Each 8-bit field represents a byte of the IPv4 address. This form of representing the bytes of an IPv4 address is often referred to as the *dotted-decimal format*.

The following figure shows the component parts of an IPv4 address, `172.16.50.56`.

FIGURE 1–1 IPv4 Address Format



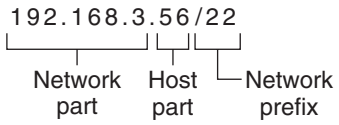
`172.16` Registered IPv4 network number. In class-based IPv4 notation, this number also defines the IP network class, Class B in this example, that would have been registered by the IANA.

`50.56` Host part of the IPv4 address. The host part uniquely identifies an interface on a system on a network. Note that for each interface on a local network, the network part of the address is the same, but the host part must be different.

If you plan to subnet a class-based IPv4 network, you need to define a subnet mask, or *netmask*, as explained in "`netmasks` Database" on page 181.

The next example shows of the CIDR format address `192.168.3.56/22`

FIGURE 1–2 CIDR Format IPv4 Address



`192.168.3` Network part, which consists of the IPv4 network number that is received from an ISP or IR.

`56` Host part, which you assign to an interface on a system.

`/22` Network prefix, which defines how many bits of the address comprise the network number. The network prefix also provides the subnet mask for the IP address. Network prefixes are also assigned by the ISP or IR.

An Oracle Solaris-based network can combine standard IPv4 addresses, CIDR format IPv4 addresses, DHCP addresses, IPv6 addresses, and private IPv4 addresses.

Designing Your IPv4 Addressing Scheme

This section describes the classes into which standard IPv4 address are organized. Though the IANA no longer gives out class-based network numbers, these network numbers are still in use on many networks. You might need to administer the address space for a site with class-based network numbers. For a complete discussion of IPv4 network classes, refer to "Network Classes" on page 193.

The following table shows the division of the standard IPv4 address into network and host address spaces. For each class, "Range" specifies the range of decimal values for the first byte of the network number. "Network Address" indicates the number of bytes of the IPv4 address that are dedicated to the network part of the address. Each byte is represented by *xxx*. "Host Address" indicates the number of bytes that are dedicated to the host part of the address. For example, in a class A network address, the first byte is dedicated to the network, and the last three bytes are dedicated to the host. The opposite designation is true for a class C network.

TABLE 1–1 Division of the IPv4 Classes

Class	Byte Range	Network Number	Host Address
A	0–127	*xxx*	*xxx.xxx.xxx*
B	128–191	*xxx.xxx*	*xxx.xxx*
C	192–223	*xxx.xxx.xxx*	*xxx*

The numbers in the first byte of the IPv4 address define whether the network is class A, B, or C. The remaining three bytes have a range from 0–255. The two numbers 0 and 255 are reserved. You can assign the numbers 1–254 to each byte, depending on the network class that was assigned to your network by the IANA.

The following table shows which bytes of the IPv4 address are assigned to you. The table also shows the range of numbers within each byte that are available for you to assign to your hosts.

TABLE 1–2 Range of Available IPv4 Classes

Network Class	Byte 1 Range	Byte 2 Range	Byte 3 Range	Byte 4 Range
A	0–127	1–254	1–254	1–254
B	128–191	Preassigned by IANA	1–254	1–254
C	192–223	Preassigned by IANA	Preassigned by IANA	1–254

IPv4 Subnet Number

Local networks with large numbers of hosts are sometimes divided into subnets. If you divide your IPv4 network number into subnets, you need to assign a network identifier to each subnet. You can maximize the efficiency of the IPv4 address space by using some of the bits from the host part of the IPv4 address as a network identifier. When used as a network identifier, the specified part of the address becomes the subnet number. You create a subnet number by using a netmask, which is a bitmask that selects the network and subnet parts of an IPv4 address. Refer to "Creating the Network Mask for IPv4 Addresses" on page 182 for details.

Designing Your CIDR IPv4 Addressing Scheme

The network classes that originally constituted IPv4 are no longer in use on the global Internet. Today, the IANA distributes classless CIDR format addresses to its registries around the world. Any IPv4 address that you obtain from an ISP is in CIDR format, as shown in Figure 1–2.

The network prefix of the CIDR address indicates how many IPv4 addresses are available for hosts on your network. Note that these host addresses are assigned to interfaces on a host. If a host has more than one physical interface, you need to assign a host address for every physical interface that is in use.

The network prefix of a CIDR address also defines the length of the subnet mask. Most Oracle Solaris commands recognize the CIDR prefix designation of a network's subnet mask. However, the Oracle Solaris installation program and `/etc/netmask file` require you to set the subnet mask by using dotted decimal representation. In these two cases, use the dotted decimal representation of the CIDR network prefix, as shown in the next table.

TABLE 1–3 CIDR Prefixes and Their Decimal Equivalent

CIDR Network Prefix	Available IP Addresses	Dotted Decimal Subnet Equivalent
/19	8,192	255.255.224.0
/20	4,096	255.255.240.0
/21	2,048	255.255.248.0
/22	1024	255.255.252.0
/23	512	255.255.254.0
/24	256	255.255.255.0
/25	128	255.255.255.128
/26	64	255.255.255.192
/27	32	255.255.255.224

For more information on CIDR addresses, refer to the following sources:

- For technical details on CIDR, refer to RFC 1519, Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy (`http://www.ietf.org/rfc/rfc1519.txt?number=1519`).
- More general information about CIDR is available from Pacific Bell Internet at Classless Inter-Domain Routing (CIDR) Overview (`http://www.wirelesstek.com/cidr.htm`).
- Another CIDR overview can be found in the Wikipedia article,"Classless inter-domain routing" (`http://en.wikipedia.org/wiki/Classless_inter-domain_routing`).

Using Private IPv4 Addresses

The IANA has reserved three blocks of IPv4 addresses for companies to use on their private networks. These addresses are defined in RFC 1918, Address Allocation for Private Internets (`http://www.ietf.org/rfc/rfc1918.txt?number=1918`). You can use these *private addresses*,

also known as 1918 addresses, for systems on local networks within a corporate intranet. However, private addresses are not valid on the Internet. Do not use them on systems that must communicate outside the local network.

The following table lists the private IPv4 address ranges and their corresponding netmasks.

IPv4 Address Range	netmask
10.0.0.0 - 10.255.255.255	10.0.0.0
172.16.0.0 - 172.31.255.255	172.16.0.0
192.168.0.0 - 192.168.255.255	192.168.0.0

How IP Addresses Apply to Network Interfaces

To connect to the network, a system must have at least one *physical network interface*. Each network interface must have its own unique IP address. During Oracle Solaris installation, you must supply the IP address for the first interface that the installation program finds. Usually that interface has the name *device-name*0, for example `bge0` or `e1000g0`. This interface is considered the *primary network interface*.

If you add a second network interface to a host, that interface also must have its own unique IP address. When you add the second network interface, the host then becomes *multihomed*. By contrast, when you add a second network interface to a host and enable IP forwarding, that host becomes a router. See "Configuring an IPv4 Router" on page 90 for an explanation.

Each network interface has a device name, a device driver, and an associated device file in the `/devices` directory. The network interface might have a device name such as `eri` or `smc0`, which are device names for two commonly used Ethernet interfaces.

For information and tasks related to interfaces, refer to Part II, "Administering Single Interfaces," in *System Administration Guide: Network Interfaces and Network Virtualization*.

Note – This book assumes that your systems have Ethernet network interfaces. If you plan to use different network media, refer to the manuals that come with the network interface for configuration information.

Naming Entities on Your Network

After you receive your assigned network IP address and you have given the IP addresses to your systems, the next task is to assign names to the hosts. Then you must determine how to handle name services on your network. You use these names initially when you set up your network and later when you expand your network through routers, bridges, or PPP.

The TCP/IP protocols locate a system on a network by using its IP address. However, if you use a recognizable name, then you can easily identify the system. Therefore, the TCP/IP protocols (and Oracle Solaris) require both the IP address and the host name to uniquely identify a system.

From a TCP/IP perspective, a network is a set of named entities. A host is an entity with a name. A router is an entity with a name. The network is an entity with a name. A group or department in which the network is installed can also be given a name, as can a division, a region, or a company. In theory, the hierarchy of names that can be used to identify a network has virtually no limit. The domain name identifies a *domain*.

Administering Host Names

Many sites let users pick host names for their machines. Servers also require at least one host name, which is associated with the IP address of its primary network interface.

As a system administrator, you must ensure that each host name in your domain is unique. In other words, no two machines on your network can both have the name "fred." However, the machine "fred" might have multiple IP addresses.

When planning your network, make a list of IP addresses and their associated host names for easy access during the setup process. The list can help you verify that all host names are unique.

Selecting a Name Service and Directory Service

Oracle Solaris enables you to use three types of name services: local files, NIS, and DNS. Name services maintain critical information about the machines on a network, such as the host names, IP addresses, Ethernet addresses, and so forth. Oracle Solaris also gives you the option of using the LDAP directory service in addition to or instead of a name service. For an introduction to name services on Oracle Solaris, refer to Part I, "About Naming and Directory Services," in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

Network Databases

When you install the operating system, you supply the host name and IP address of your server, clients, or standalone system as part of the procedure. The Oracle Solaris installation program adds this information into the `hosts` This database is part of a set of network databases that contain information necessary for TCP/IP operation on your network. The name service that you select for your network reads these databases.

The configuration of the network databases is critical. Therefore, you need to decide which name service to use as part of the network planning process. Moreover, the decision to use name services also affects whether you organize your network into an administrative domain. "Network Databases and the `nsswitch.conf` File" on page 185 has detailed information on the set of network databases.

Using NIS or DNS as the Name Service

The NIS and DNS name services maintain network databases on several servers on the network. *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* describes these name services and explains how to configure the databases. In addition, the guide explain the "namespace" and "administrative domain" concepts in detail.

Using Local Files as the Name Service

If you do not implement NIS, LDAP, or DNS, the network uses *local files* to provide the name service. The term "local files" refers to the series of files in the `/etc` directory that the network databases use. The procedures in this book assume you are using local files for your name service, unless otherwise indicated.

Note – If you decide to use local files as the name service for your network, you can set up another name service at a later date.

Domain Names

Many networks organize their hosts and routers into a hierarchy of administrative domains. If you are using the NIS or DNS name service, you must select a domain name for your organization that is unique worldwide. To ensure that your domain name is unique, you should register the domain name with the InterNIC. If you plan to use DNS, you also need to register your domain name with the InterNIC.

The domain name structure is hierarchical. A new domain typically is located below an existing, related domain. For example, the domain name for a subsidiary company can be located below the domain of the parent company. If the domain name has no other relationship, an organization can place its domain name directly under one of the existing top-level domains.

The following are a few examples of top-level domains:

- `.com` – Commercial companies (international in scope)
- `.edu` – Educational institutions (international in scope)
- `.gov` – U.S. government agencies
- `.fr` – France

You select the name that identifies your organization, with the provision that the name must be unique.

Administrative Subdivisions

The question of administrative subdivisions deals with matters of size and control. The more hosts and servers that you have in a network, the more complex your management task. You might want to handle such situations by setting up additional administrative divisions. Add networks of a particular class. Divide existing networks into subnets. The decision about setting up administrative subdivisions for your network is determined by the following factors:

- **How large is the network?**

  A single administrative division can handle a single network of several hundred hosts, all in the same physical location and requiring the same administrative services. However, sometimes you should establish several administrative subdivisions. Subdivisions are particularly useful if you have a small network with subnets and the network is scattered over an extensive geographical area.

- **Do users on the network have similar needs?**

  For example, you might have a network that is confined to a single building and supports a relatively small number of machines. These machines are divided among a number of subnetworks. Each subnetwork supports groups of users with different needs. In this example, you might use an administrative subdivision for each subnet.

Planning for Routers on Your Network

Recall that in TCP/IP, two types of entities exist on a network: hosts and routers. All networks must have hosts, while not all networks require routers. The physical topology of the network determines if you need routers. This section introduces the concepts of network topology and routing. These concepts are important when you decide to add another network to your existing network environment.

Note – For complete details and tasks for router configuration on IPv4 networks, refer to "Packet Forwarding and Routing on IPv4 Networks" on page 84. For complete details and tasks for router configuration on IPv6 networks, refer to "Configuring an IPv6 Router" on page 112.

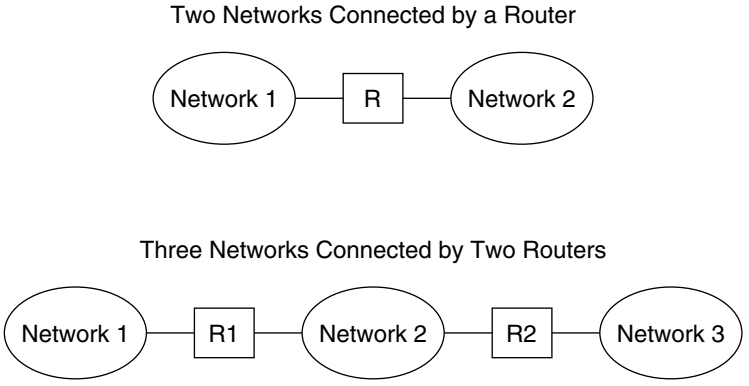
Network Topology Overview

Network topology describes how networks fit together. Routers are the entities that connect networks to each other. A router is any machine that has two or more network interfaces and implements IP forwarding. However, the system cannot function as a router until properly configured, as described in "Configuring an IPv4 Router" on page 90.

Routers connect two or more networks to form larger internetworks. The routers must be configured to pass packets between two adjacent networks. The routers also should be able to pass packets to networks that lie beyond the adjacent networks.

The following figure shows the basic parts of a network topology. The first illustration shows a simple configuration of two networks that are connected by a single router. The second illustration shows a configuration of three networks, interconnected by two routers. In the first example, Router R joins Network 1 and Network 2 into a larger internetwork. In the second example, Router R1 connects Networks 1 and 2. Router R2 connects Networks 2 and 3. The connections form a network that includes Networks 1, 2, and 3.

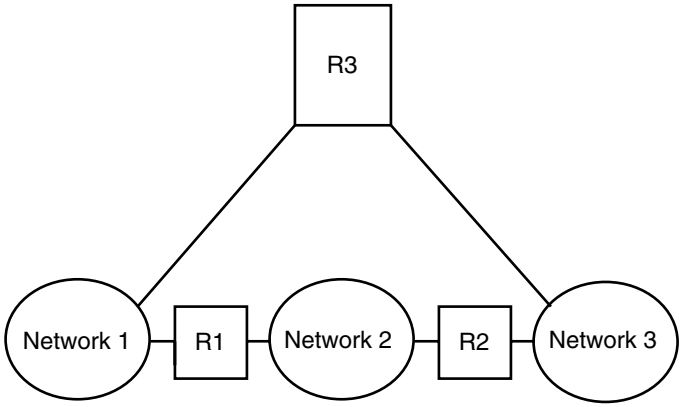
FIGURE 1–3 Basic Network Topology



In addition to joining networks into internetworks, routers route packets between networks that are based on the addresses of the destination network. As internetworks grow more complex, each router must make more and more decisions about the packet destinations.

The following figure shows a more complex case. Router R3 directly connects networks 1 and 3. The redundancy improves reliability. If network 2 goes down, router R3 still provides a route between networks 1 and 3. You can interconnect many networks. However, the networks must use the same network protocols.

FIGURE 1–4 A Network Topology That Provides an Additional Path Between Networks



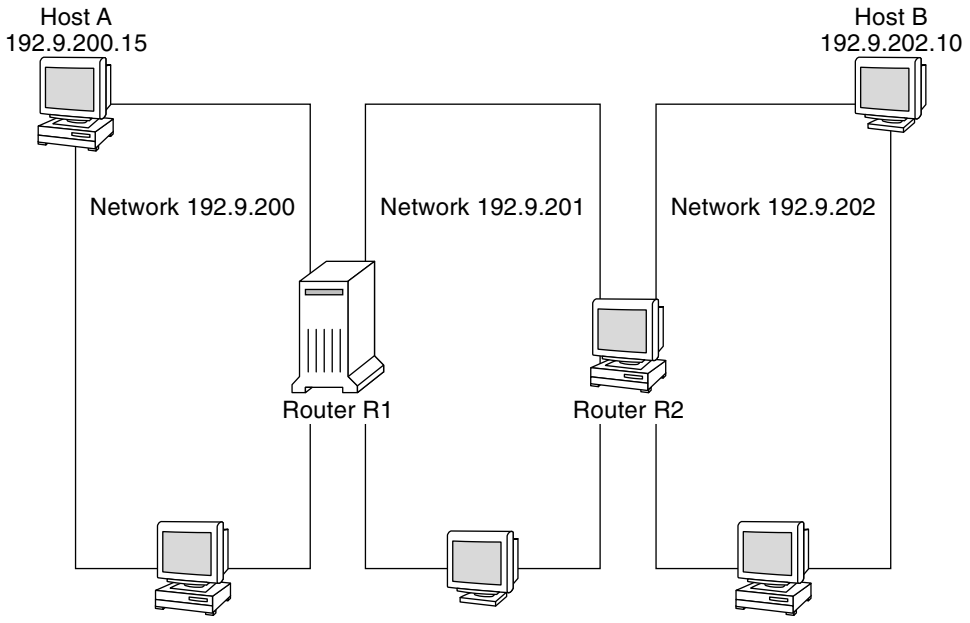
How Routers Transfer Packets

The IP address of the recipient, which is a part of the packet header, determines how the packet is routed. If this address includes the network number of the local network, the packet goes directly to the host with that IP address. If the network number is not the local network, the packet goes to the router on the local network.

Routers maintain routing information in *routing tables*. These tables contain the IP address of the hosts and routers on the networks to which the router is connected. The tables also contain pointers to these networks. When a router receives a packet, the router checks its routing table to determine if the table lists the destination address in the header. If the table does not contain the destination address, the router forwards the packet to another router that is listed in its routing table. Refer to "Configuring an IPv4 Router" on page 90 for detailed information on routers.

The following figure shows a network topology with three networks that are connected by two routers.

FIGURE 1–5 A Network Topology With Three Interconnected Networks



Router R1 connects networks `192.9.200` and `192.9.201`. Router R2 connects networks `192.9.201` and `192.9.202`. If Host A on network `192.9.200` sends a message to Host B on network `192.9.202`, the following events occur:

1. Host A sends a packet out over network `192.9.200`. The packet header contains the IPv4 address of the recipient Host B, `192.9.202.10`.
2. None of the machines on network `192.9.200` has the IPv4 address `192.9.202.10`. Therefore, Router R1 accepts the packet.
3. Router R1 examines its routing tables. No machine on network `192.9.201` has the address `192.9.202.10`. However, the routing tables do list Router R2.
4. R1 then selects R2 as the "next hop" Router. R1 sends the packet to R2.
5. Because R2 connects network `192.9.201` to `192.9.202`, R2 has routing information for Host B. Router R2 then forwards the packet to network `192.9.202`, where Host B accepts the packet.

◆ ◆ ◆ CHAPTER 2

Planning an IPv6 Addressing Scheme (Overview)

This chapter presents an overview of the Oracle Solaris Internet Protocol version 6 (IPv6) implementation. This implementation includes the associated daemon and utilities that support the IPv6 address space.

IPv6 and IPv4 addresses coexist in the Oracle Solaris networking environment. Systems that are configured with IPv6 addresses retain their IPv4 addresses, if these addresses already exist. Operations that involve IPv6 addresses do not adversely affect IPv4 operations, and vice versa.

The following major topics are discussed:

For more detailed information about IPv6, consult the following chapters.

- IPv6 network planning – Chapter 3, "Planning an IPv6 Network (Tasks)"
- IPv6-related tasks – Chapter 5, "Enabling IPv6 on a Network (Tasks)," andChapter 6, "Administering a TCP/IP Network (Tasks)."
- IPv6 details – Chapter 10, "IPv6 in Depth (Reference)"

Major Features of IPv6

The defining feature of IPv6 is increased address space in comparison to IPv4. IPv6 also improves Internet capabilities in numerous areas, as outlined in this section.

Expanded Addressing

IP address size increases from 32 bits in IPv4 to 128 bits in IPv6, to support more levels of addressing hierarchy. In addition, IPv6 provides many more addressable IPv6 systems. For more information, see "IPv6 Addressing Overview" on page 54.

Address Autoconfiguration and Neighbor Discovery

The IPv6 *Neighbor Discovery (ND)* protocol facilitates the autoconfiguration of IPv6 addresses. *Autoconfiguration* is the ability of an IPv6 host to automatically generate its own IPv6 address, which makes address administration easier and less time-consuming. For more information, see "IPv6 Address Autoconfiguration" on page 61.

The Neighbor Discovery protocol corresponds to a combination of these IPv4 protocols: Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP), Router Discovery (RDISC), and ICMP Redirect. IPv6 routers use Neighbor Discovery to advertise the IPv6 site prefix. IPv6 hosts use Neighbor Discovery for various purposes, which include soliciting the prefix from an IPv6 router. For more information, see "IPv6 Neighbor Discovery Protocol Overview" on page 59.

Header Format Simplification

The IPv6 header format either drops or makes optional certain IPv4 header fields. This change keeps the bandwidth cost of the IPv6 header as low as possible, despite the increased address size. Even though IPv6 addresses are four times longer than IPv4 addresses, the IPv6 header is only twice the size of the IPv4 header.

Improved Support for IP Header Options

Changes in the way IP header options are encoded allow for more efficient forwarding. Also, IPv6 options have less stringent limits on their length. The changes provide greater flexibility for introducing new options in the future.

Application Support for IPv6 Addressing

Many critical Oracle Solaris network services recognize and support IPv6 addresses, for example:

- Name services, such as DNS, LDAP, and NIS. For more information on IPv6 support by these name services, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

- Authentication and privacy applications, such as IP Security Architecture (IPsec) and Internet Key Exchange (IKE). For more information, see Part III, "IP Security."
- Differentiated services, as provided by IP Quality of Service (IPQoS). For more information, see Part V, "IP Quality of Service (IPQoS)."
- Failover detection, as provided by IP network multipathing (IPMP). For more information, see "Failure and Repair Detection in IPMP" in *System Administration Guide: Network Interfaces and Network Virtualization*.

Additional IPv6 Resources

In addition to this Part, you can obtain information about IPv6 from the sources that are listed in the following sections.

IPv6 Requests for Comments and Internet Drafts

Many RFCs are available regarding IPv6. The following table lists the major IPv6 articles and their Internet Engineering Task Force (IETF) web locations as of this writing.

TABLE 2–1 IPv6–Related RFCs and Internet Drafts

RFC or Internet Draft	Subject	Location
RFC 2461, *Neighbor Discovery for IP Version 6 (IPv6)*	Describes the features and functions of IPv6 Neighbor Discovery protocol	http://www.ietf.org/rfc/rfc2461.txt$number=2461 (`http://www.ietf.org/rfc/rfc2461.txt?number-2461`)
RFC 3306, *Unicast—Prefix—Based IPv6 Multicast Addresses*	Describes the format and types of IPv6 multicast addresses	ftp://ftp.rfc-editor.org/in-notes/rfc3306.txt (`ftp://ftp.rfc-editor.org/in-notes/rfc3306.txt`)
RFC 3484: *Default Address Selection for Internet Protocol version 6 (IPv6)*	Describes the algorithms used in IPv6 default address selection	http://www.ietf.org/rfc/rfc3484?number=3484 (`http://www.ietf.org/rfc/rfc3484.txt?number=3484`)
RFC 3513, *Internet Protocol version 6 (IPv6) Addressing Architecture*	Contains complete details about the types of IPv6 addresses and includes many examples	http://www.ietf.org/rfc/rfc3513.txt?number=3513 (`http://www.ietf.org/rfc/rfc3513.txt?number=3513`)
RFC 3587, *IPv6 Global Unicast Address Format*	Defines the standard format for IPv6 unicast addresses	http://www.ietf.org/rfc/rfc3587.txt?number=3587 (`http://www.ietf.org/rfc/rfc3587.txt?number=3587`)

Web Sites

The following web sites provide useful information about IPv6.

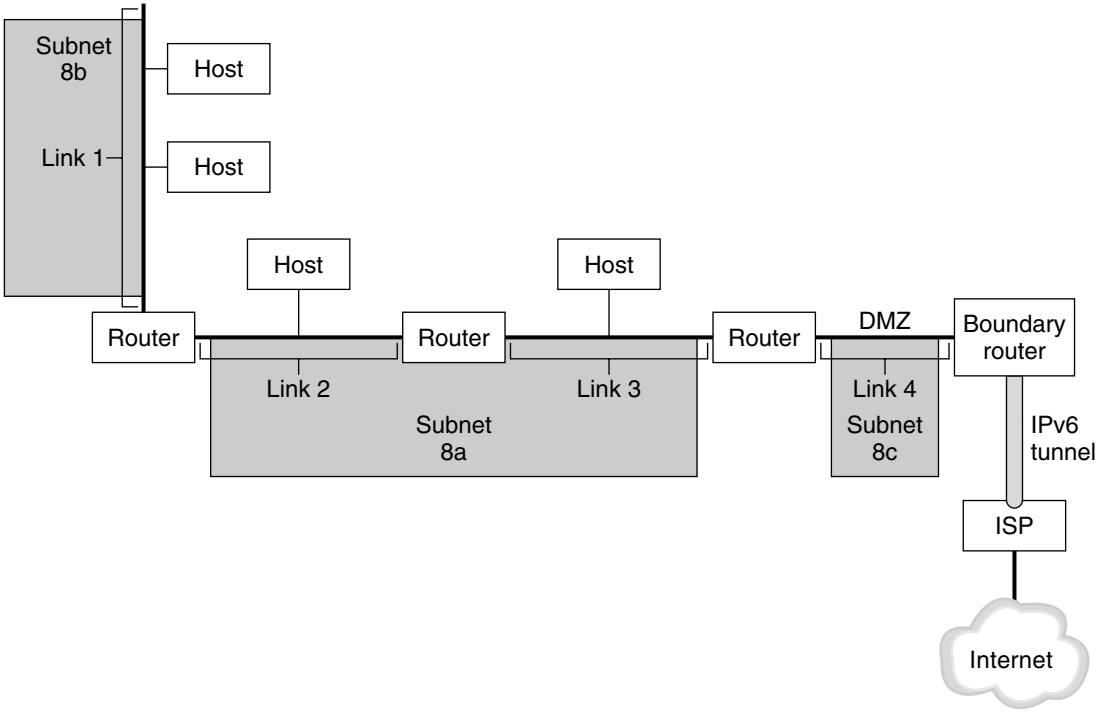
TABLE 2–2 IPv6–Related Web Sites

Web Site	Description	Location
IPv6 Forum	Links to IPv6–related presentations, events, classes, and implementations worldwide are available from this society's web site	http://www.ipv6forum.com
Internet Educational Task Force IPv6 Working Group	Links to all relevant IPv6 RFCs and Internet Drafts are on the home page of this IETF working group	http://www.ietf.org/html.charters/ipv6-charter.html

IPv6 Network Overview

This section introduces terms that are fundamental to the IPv6 network topology. The following figure shows the basic parts of an IPv6 network.

FIGURE 2–1 Basic Components of an IPv6 Network



The figure depicts an IPv6 network and its connection to an ISP. The internal network consists of Links 1, 2, 3, and 4. Each link is populated by hosts and terminated by a router. Link 4, which is the network's DMZ, is terminated on one end by the boundary router. The boundary router runs an IPv6 tunnel to an ISP, which provides Internet connectivity for the network. Links 2 and 3 are administered as Subnet 8a. Subnet 8b consists only of systems on Link 1. Subnet 8c is contiguous with the DMZ on Link 4.

As illustrated in Figure 2–1, an IPv6 network has essentially the same components as an IPv4 network. However, IPv6 terminology differs slightly from IPv4 terminology. Here is a list of familiar terms for network components as they are used in an IPv6 context.

node — Any system with an IPv6 address and interface that is configured for IPv6 support. This generic term applies to both hosts and routers.

IPv6 router — A node that forwards IPv6 packets. At least one of the router's interfaces must be configured for IPv6 support. An IPv6 router can also advertise the registered IPv6 site prefix for the enterprise over the internal network.

IPv6 host — A node with an IPv6 address. An IPv6 host can have more than one interface that is configured for IPv6 support. As in IPv4, IPv6 hosts do not forward packets.

link — A single, contiguous network medium that is bounded on either end by a router.

neighbor — An IPv6 node that is on the same link as the local node.

IPv6 subnet — The administrative segment of an IPv6 network. Components of an IPv6 subnet can directly correspond to all nodes on a link, as in IPv4. Nodes on a link can be administered in separate subnets, if required. Additionally, IPv6 does support multilink subnets, where nodes on more than one link can be components of a single subnet. Links 2 and 3 in Figure 2–1 are components of multilink Subnet 8a.

IPv6 tunnel — A tunnel that provides a virtual point-to-point path between an IPv6 node and another IPv6 node endpoint. IPv6 supports manually configurable tunnels and automatic 6to4 tunnels.

boundary router — The router at the edge of a network that provides one end of the IPv6 tunnel to an endpoint outside the local network. This router must have at least one IPv6 interface to the internal network. For the external network, the router can have an IPv6 interface or an IPv4 interface.

IPv6 Addressing Overview

IPv6 addresses are assigned to interfaces, rather than to nodes, in recognition that a node can have more than one interface. Moreover, you can assign more than one IPv6 address to an interface.

Note – For complete technical information about the IPv6 address format, go to RFC 2374, IPv6 Global Unicast Address Format (`http://www.ietf.org/rfc/rfc2374.txt?number=2374`)

IPv6 defines three address types:

unicast Identifies an interface of an individual node.

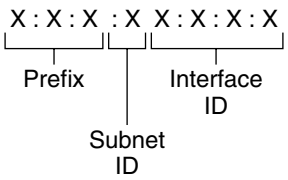
multicast Identifies a group of interfaces, usually on different nodes. Packets that are sent to the multicast address go to all members of the *multicast group*.

anycast Identifies a group of interfaces, usually on different nodes. Packets that are sent to the anycast address go to the *anycast group* member node that is physically closest to the sender.

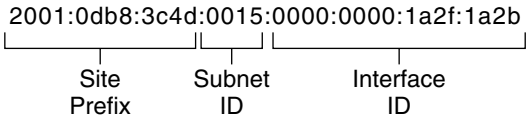
Parts of the IPv6 Address

An IPv6 address is 128 bits in length and consists of eight, 16-bit fields, with each field bounded by a colon. Each field must contain a hexadecimal number, in contrast to the dotted-decimal notation of IPv4 addresses. In the next figure, the x's represent hexadecimal numbers.

FIGURE 2–2 Basic IPv6 Address Format



Example:



The leftmost three fields (48 bits) contain the *site prefix*. The prefix describes the *public topology* that is usually allocated to your site by an ISP or Regional Internet Registry (RIR).

The next field is the 16-bit *subnet ID*, which you (or another administrator) allocate for your site. The subnet ID describes the *private topology*, also known as the *site topology*, because it is internal to your site.

The rightmost four fields (64 bits) contain the *interface ID*, also referred to as a *token*. The interface ID is either automatically configured from the interface's MAC address or manually configured in EUI-64 format.

Consider again the address in Figure 2–2:

```
2001:0db8:3c4d:0015:0000:0000:1a2f:1a2b
```

This example shows all 128 bits of an IPv6 address. The first 48 bits, `2001:0db8:3c4d`, contain the site prefix, representing the public topology. The next 16 bits, `0015`, contain the subnet ID, representing the private topology for the site. The lower order, rightmost 64 bits, `0000:0000:1a2f:1a2b`, contain the interface ID.

Abbreviating IPv6 Addresses

Most IPv6 addresses do not occupy all of their possible 128 bits. This condition results in fields that are padded with zeros or contain only zeros.

The IPv6 addressing architecture allows you use the two-colon (::) notation to represent contiguous 16-bit fields of zeros. For example, you might abbreviate the IPv6 address in Figure 2–2 by replacing the two contiguous fields of zeros in the interface ID with two colons. The resulting address is `2001:0db8:3c4d:0015::1a2f:1a2b`. Other fields of zeros can be represented as a single 0. You can also omit any leading zeros in a field, such as changing `0db8` to `db8`.

So the address `2001:0db8:3c4d:0015:0000:0000:1a2f:1a2b` can be abbreviated as `2001:db8:3c4d:15::1a2f:1a2b`.

You can use the two colon notation to replace any contiguous fields of all zeros in the IPv6 address. For example, the IPv6 address `2001:0db8:3c4d:0015:0000:d234::3eee:0000` can be collapsed into `2001:db8:3c4d:15:0:d234:3eee::`.

Prefixes in IPv6

The leftmost fields of the IPv6 address contain the prefix, which is used for routing IPv6 packets. IPv6 prefixes have the following format:

prefix/length in bits

Prefix length is stated in classless inter-domain routing (CIDR) notation. CIDR notation is a slash at the end of the address that is followed by the prefix length in bits. For information on CIDR format IP addresses, refer to "Designing Your CIDR IPv4 Addressing Scheme" on page 40.

The *site prefix* of an IPv6 address occupies up to 48 of the leftmost bits of the IPv6 address. For example, the site prefix of the IPv6 address `2001:db8:3c4d:0015:0000:0000:1a2f:1a2b/48` is contained in the leftmost 48 bits, `2001:db8:3c4d`. You use the following representation, with zeros compressed, to represent this prefix:

```
2001:db8:3c4d::/48
```

Note – The prefix `2001:db8::/32` is a special IPv6 prefix that is used specifically for documentation examples.

You can also specify a *subnet prefix*, which defines the internal topology of the network to a router. The example IPv6 address has the following subnet prefix.

```
2001:db8:3c4d:15::/64
```

The subnet prefix always contains 64 bits. These bits include 48 bits for the site prefix, in addition to 16 bits for the subnet ID.

The following prefixes have been reserved for special use:

`2002::/16` Indicates that a 6to4 routing prefix follows.

`fe80::/10` Indicates that a link-local address follows.

`ff00::/8` Indicates that a multicast address follows.

Unicast Addresses

IPv6 includes two different unicast address assignments:

- Global unicast address
- Link-local address

The type of unicast address is determined by the leftmost (high order) contiguous bits in the address, which contain the prefix.

The unicast address format is organized in the following hierarchy:

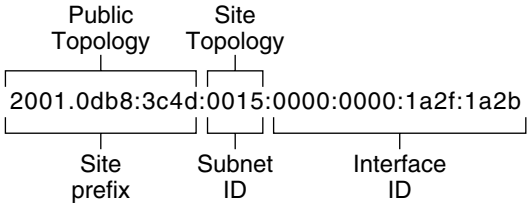
- Public topology

- Site (private) topology
- Interface ID

Global Unicast Address

The global unicast address is globally unique in the Internet. The example IPv6 address that is shown in "Prefixes in IPv6" on page 55 is a global unicast address. The next figure shows the scope of the global unicast address, as compared to the parts of the IPv6 address.

FIGURE 2–3 Parts of the Global Unicast Address



Public Topology

The site prefix defines the *public topology* of your network to a router. You obtain the site prefix for your enterprise from an ISP or Regional Internet Registry (RIR).

Site Topology and IPv6 Subnets

IN IPv6, the *subnet ID* defines an administrative subnet of the network and is up to 16 bits in length. You assign a subnet ID as part of IPv6 network configuration. The *subnet prefix* defines the site topology to a router by specifying the specific link to which the subnet has been assigned.

IPv6 subnets are conceptually the same as IPv4 subnets, in that each subnet is usually associated with a single hardware link. However, IPv6 subnet IDs are expressed in hexadecimal notation, rather than in dotted decimal notation.

Interface ID

The *interface ID* identifies an interface of a particular node. An interface ID must be unique within the subnet. IPv6 hosts can use the Neighbor Discovery protocol to automatically generate their own interface IDs. Neighbor Discovery automatically generates the interface ID, based on the MAC or EUI-64 address of the host's interface. You can also manually assign interface IDs, which is recommended for IPv6 routers and IPv6-enabled servers. For instructions on how to create a manual EUI-64 address, refer to RFC 3513 Internet Protocol Version 6 (IPv6) Addressing Architecture.

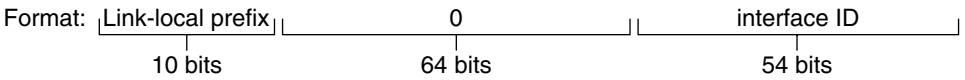
Transitional Global Unicast Addresses

For transition purposes, the IPv6 protocol includes the ability to embed an IPv4 address within an IPv6 address. This type of IPv4 address facilitates the tunneling of IPv6 packets over existing IPv4 networks. One example of a transitional global unicast address is the 6to4 address. For more information on 6to4 addressing, refer to "6to4 Tunnels" on page 153.

Link-Local Unicast Address

The link-local unicast address can be used only on the local network link. Link-local addresses are not valid nor recognized outside the enterprise. The following example shows the format of the link-local address.

EXAMPLE 2–1 Parts of the Link-Local Unicast Address



Example: fe80::123e:456d

A *link-local prefix* has the following format:

`fe80::`*interface-ID*`/10`

The following is an example of a link-local address:

```
fe80::23a1:b152
```

`fe80`	Hexadecimal representation of the 10-bit binary prefix 1111111010. This prefix identifies the type of IPv6 address as link local.
interface-ID	Hexadecimal address of the interface, which is usually derived from the 48-bit MAC address.

When you enable IPv6 during Oracle Solaris installation, the lowest numbered interface on the local machine is configured with a link-local address. Each interface requires at least one link-local address to identify the node to other nodes on the local link. Therefore, you need to manually configure link-local addresses for additional interfaces of a node. After configuration, the node uses its link-local addresses for automatic address configuration and neighbor discovery.

Multicast Addresses

IPv6 supports the use of multicast addresses. The multicast address identifies a *multicast group*, which is a group of interfaces, usually on different nodes. An interface can belong to any number of multicast groups. If the first 16 bits of an IPv6 address is `ff00`*n*, the address is a multicast address.

Multicast addresses are used for sending information or services to all interfaces that are defined as members of the multicast group. For example, one use of multicast addresses is to communicate with all IPv6 nodes on the local link.

When an interface's IPv6 unicast address is created, the kernel automatically makes the interface a member of certain multicast groups. For example, the kernel makes each node a member of the Solicited Node multicast group, which is used by the Neighbor Discovery protocol to detect reachability. The kernel also automatically makes a node a member of the All-Nodes or All Routers multicast groups.

For detailed information about multicast addresses, refer to "IPv6 Multicast Addresses in Depth" on page 199. For technical information, see RFC 3306, Unicast-Prefix-based IPv6 Multicast Addresses (`ftp://ftp.rfc-editor.org/in-notes/rfc3306.txt`), which explains the multicast address format. For more information about the proper use of multicast addresses and groups, RFC 3307, Allocation Guidelines for IPv6 Multicast Addresses (`ftp://ftp.rfc-editor.org/in-notes/rfc3307.txt`).

Anycast Addresses and Groups

IPv6 anycast addresses identify a group of interfaces on different IPv6 nodes. Each group of interfaces is known as an *anycast group*. When a packet is sent to the anycast address, the anycast group member that is physically closest to the sender receives the packet.

Note – The Oracle Solaris implementation of IPv6 does not support the creation of anycast addresses and groups. However, Oracle Solaris IPv6 nodes can send packets to anycast addresses. For more information, see "Considerations for Tunnels to a 6to4 Relay Router" on page 156.

IPv6 Neighbor Discovery Protocol Overview

IPv6 introduces the Neighbor Discovery protocol, which uses messaging as the means to handle the interaction between neighbor nodes. *Neighbor nodes* are IPv6 nodes that are on the same

link. For example, by issuing neighbor discovery-related messages, a node can learn a neighbor's link-local address. Neighbor Discovery controls the following major activities on the IPv6 local link:

- **Router discovery** – Aids hosts in locating routers on the local link.
- **Address autoconfiguration** – Enables a node to automatically configure IPv6 addresses for its interfaces.
- **Prefix discovery** – Enables nodes to discover the known subnet prefixes that have been allocated to a link. Nodes use prefixes to distinguish destinations that are on the local link from those destinations that are only reachable through a router.
- **Address resolution** – Helps nodes to determine the link-local address of a neighbor, given only the destinations's IP address.
- **Next-hop determination** – Uses an algorithm to determine the IP address of a packet recipient one hop that is beyond the local link. The next-hop can be a router or the destination node.
- **Neighbor unreachability detection** – Aids nodes to determine if a neighbor is no longer reachable. For both routers and hosts, address resolution can be repeated.
- **Duplicate address detection** – Enables a node to determine if an address that the node wants to use is not already in use.
- **Redirection** – Enables a router to inform a host of a better first-hop node to use to reach a particular destination.

Neighbor Discovery uses the following ICMP message types for communication among nodes on a link:

- Router solicitation
- Router advertisement
- Neighbor solicitation
- Neighbor advertisement
- Redirection

For detailed information on Neighbor Discovery messages and other Neighbor Discovery protocol topics, refer to "IPv6 Neighbor Discovery Protocol" on page 214. For technical information on Neighbor Discovery, see RFC 2461, Neighbor Discovery for IP Version 6 (IPv6) (`http://www.ietf.org/rfc/rfc2461.txt?number=2461`).

IPv6 Address Autoconfiguration

A major feature of IPv6 is a host's ability to autoconfigure an interface. Through Neighbor Discovery, the host locates an IPv6 router on the local link and requests a site prefix. The host does the following, as part of the autoconfiguration process:

- Creates a link-local address for each interface, which does not require a router on the link.
- Verifies the address's uniqueness on a link, which does not require a router on the link.
- Determines if the global addresses should be obtained through the stateless mechanism, the stateful mechanism, or both mechanisms. (Requires a router on the link.)

Stateless Autoconfiguration Overview

Stateless autoconfiguration requires no manual configuration of hosts, minimal (if any) configuration of routers, and no additional servers. The stateless mechanism enables a host to generate its own addresses. The stateless mechanism uses local information as well as nonlocal information that is advertised by routers to generate the addresses.

You can implement temporary addresses for an interface, which are also autoconfigured. You enable a temporary address token for one or more interfaces on a host. However, unlike standard, autoconfigured IPv6 addresses, a temporary address consists of the site prefix and a randomly generated 64 bit number. This random number becomes the interface ID portion of the IPv6 address. A link-local address is not generated with the temporary address as the interface ID.

Routers advertise all prefixes that have been assigned on the link. IPv6 hosts use Neighbor Discovery to obtain a subnet prefix from a local router. Hosts automatically create IPv6 addresses by combining the subnet prefix with an interface ID that is generated from an interface's MAC address. In the absence of routers, a host can generate only link-local addresses. Link-local addresses can only be used for communication with nodes on the same link.

Note – Do not use stateless autoconfiguration to create the IPv6 addresses of servers. Hosts automatically generate interface IDs that are based on hardware-specific information during autoconfiguration. The current interface ID could become invalid if the existing interface is swapped for a new interface.

◆ ◆ ◆ CHAPTER 3

Planning an IPv6 Network (Tasks)

Deploying IPv6 on a new network or an existing network requires a major planning effort. This chapter contains the planning tasks that are necessary before you can configure IPv6 at your site. For existing networks, IPv6 deployment should be phased in gradually. The topics in this chapter help you phase in IPv6 onto an otherwise IPv4-only network.

The following topics are discussed in this chapter:

For an introduction to IPv6 concepts, refer to Chapter 2, "Planning an IPv6 Addressing Scheme (Overview)." For detailed information, refer to Chapter 10, "IPv6 in Depth (Reference)."

IPv6 Planning (Task Maps)

Complete the tasks in the next task map in sequential order to accomplish the planning tasks necessary for IPv6 deployment.

The following table lists different tasks for configuring the IPv6 network. The table includes a description of what each task accomplishes and the section in the current documentation where the specific steps to perform the task are detailed.

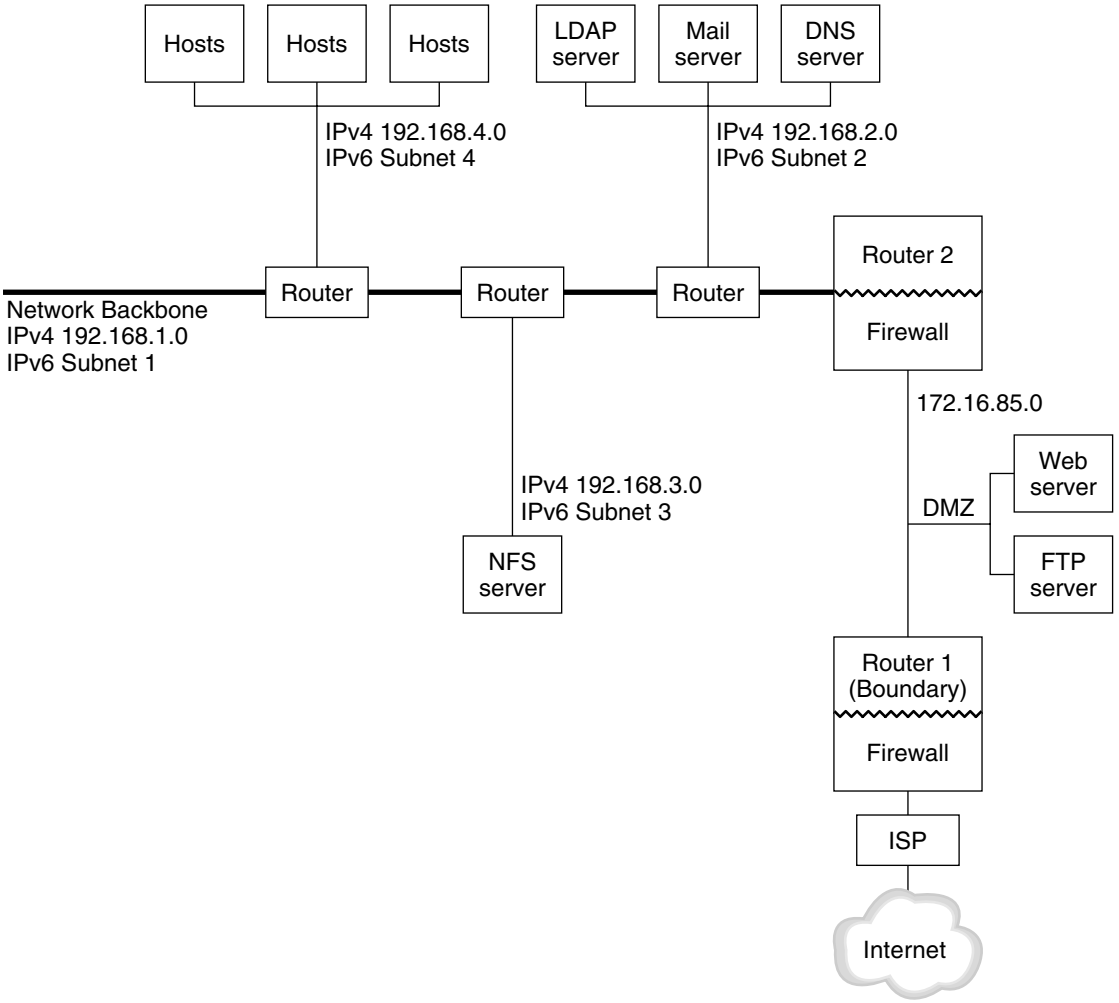
Task	Description	For Instructions
1. Prepare your hardware to support IPv6.	Ensure that your hardware can be upgraded to IPv6.	"Preparing the Network Topology for IPv6 Support" on page 66

Task	Description	For Instructions
2. Get an ISP that supports IPv6.	Ensure that your current ISP supports IPv6. Otherwise, find an ISP who can support IPv6. You can use two ISPs, one ISP for IPv6 and one for ISP IPv4 communications.	
3. Ensure that your applications are IPv6 ready.	Verify that your applications can run in an IPv6 environment.	"How to Prepare Network Services for IPv6 Support" on page 68
4. Get a site prefix.	Obtain a 48-bit site prefix for your site from your ISP or from the nearest RIR.	"Obtaining a Site Prefix" on page 70
5. Create a subnet addressing plan.	You need to plan the overall IPv6 network topology and addressing scheme before you can configure IPv6 on the various nodes in your network.	"Creating a Numbering Scheme for Subnets" on page 71
6. Design a plan for tunnel usage.	Determine which routers should run tunnels to other subnets or external networks.	"Planning for Tunnels in the Network Topology" on page 69
7. Create an addressing plan for entities on the network.	Your plan for addressing servers, routers, and hosts should be in place before IPv6 configuration.	"Creating an IPv6 Addressing Plan for Nodes" on page 71
8. Develop an IPv6 security policy.	Investigate IP Filter, IP security architecture (IPsec), Internet Key Exchange (IKE), and other Oracle Solaris security features as you develop an IPv6 security policy.	Part III, "IP Security"
9. (Optional) Set up a DMZ.	For security purposes, you need an addressing plan for the DMZ and its entities before you configure IPv6.	"Security Considerations for the IPv6 Implementation" on page 70
10. Enable the nodes to support IPv6.	Configure IPv6 on all routers and hosts.	"IPv6 Router Configuration (Task Map)" on page 113
11. Turn on network services.	Make sure that existing servers can support IPv6.	"Major TCP/IP Administrative Tasks (Task Map)" on page 128
12. Update name servers for IPv6 support.	Make sure that DNS, NIS, and LDAP servers are updated with the new IPv6 addresses.	"Configuring Name Service Support for IPv6" on page 122

IPv6 Network Topology Scenario

The tasks throughout this chapter explain how to plan for IPv6 services on a typical enterprise network. The following figure shows the network that is referred to throughout the chapter. Your proposed IPv6 network might include some or all of the network links that are illustrated in this figure.

FIGURE 3–1 IPv6 Network Topology Scenario



The enterprise network scenario consists of five subnets with existing IPv4 addresses. The links of the network correspond directly to the administrative subnets. The four internal networks are shown with RFC 1918-style private IPv4 addresses, which is a common solution for the lack of IPv4 addresses. The addressing scheme of these internal networks follows:

- Subnet 1 is the internal network backbone `192.168.1`.
- Subnet 2 is the internal network `192.168.2`, with LDAP, `sendmail`, and DNS servers.
- Subnet 3 is the internal network `192.168.3`, with the enterprise's NFS servers.
- Subnet 4 is the internal network `192.168.4`, which contains hosts for the enterprise's employees.

The external, public network `172.16.85` functions as the corporation's DMZ. This network contains web servers, anonymous FTP servers, and other resources that the enterprise offers to the outside world. Router 2 runs a firewall and separates public network `172.16.85` from the internal backbone. On the other end of the DMZ, Router 1 runs a firewall and serves as the enterprise's boundary server.

In Figure 3–1, the public DMZ has the RFC 1918 private address `172.16.85`. In the real world, the public DMZ must have a registered IPv4 address. Most IPv4 sites use a combination of public addresses and RFC 1918 private addresses. However, when you introduce IPv6, the concept of public addresses and private addresses changes. Because IPv6 has a much larger address space, you use public IPv6 addresses on both private networks and public networks.

Preparing the Existing Network to Support IPv6

Note – The Oracle Solaris dual protocol stack supports concurrent IPv4 and IPv6 operations. You can successfully run IPv4–related operations during and after deployment of IPv6 on your network.

IPv6 introduces additional features to an existing network. Therefore, when you first deploy IPv6, you must ensure that you do not disrupt any operations that are working with IPv4. The subjects covered in this section describe how to introduce IPv6 to an existing network in a step-by-step fashion.

Preparing the Network Topology for IPv6 Support

The first step in IPv6 deployment is to assess which existing entities on your network can support IPv6. In most cases, the network topology-wires, routers, and hosts-can remain unchanged as you implement IPv6. However, you might have to prepare existing hardware and applications for IPv6 before actually configuring IPv6 addresses on network interfaces.

Verify which hardware on your network can be upgraded to IPv6. For example, check the manufacturers' documentation for IPv6 readiness regarding the following classes of hardware:

- Routers
- Firewalls
- Servers
- Switches

Note – All procedures in the this Part assume that your equipment, particularly routers, can be upgraded to IPv6.

Some router models cannot be upgraded to IPv6. For more information and a workaround, refer to "IPv4 Router Cannot Be Upgraded to IPv6" on page 174.

Preparing Network Services for IPv6 Support

The following typical IPv4 network services in the current Oracle Solaris release are IPv6 ready:

- `sendmail`
- NFS
- HTTP (Apache 2.x or Orion)
- DNS
- LDAP

The IMAP mail service is for IPv4 only.

Nodes that are configured for IPv6 can run IPv4 services. When you turn on IPv6, not all services accept IPv6 connections. Services that have been ported to IPv6 will accept a connection. Services that have not been ported to IPv6 continue to work with the IPv4 half of the protocol stack.

Some issues can arise after you upgrade services to IPv6. For details, see "Problems After Upgrading Services to IPv6" on page 175.

Preparing Servers for IPv6 Support

Because servers are considered IPv6 hosts, by default their IPv6 addresses are automatically configured by the Neighbor Discovery protocol. However, many servers have multiple network interface cards (NICs) that you might want to swap out for maintenance or replacement. When you replace one NIC, Neighbor Discovery automatically generates a new interface ID for that NIC. This behavior might not be acceptable for a particular server.

Therefore, consider manually configuring the interface ID portion of the IPv6 addresses for each interface of the server. For instructions, refer to "How to Configure a User-Specified IPv6 Token" on page 120. Later, when you need to replace an existing NIC, the already configured IPv6 address is applied to the replacement NIC.

▼ How to Prepare Network Services for IPv6 Support

1 **Update the following network services to support IPv6:**

- Mail servers
- NIS servers
- NFS

Note – LDAP supports IPv6 without requiring IPv6-specific configuration tasks.

2 **Verify that your firewall hardware is IPv6 ready.**

Refer to the appropriate firewall-related documentation for instructions.

3 **Verify that other services on your network have been ported to IPv6.**

For more information, refer to marketing collateral and associated documentation for the software.

4 **If your site deploys the following services, make sure that you have taken the appropriate measures for these services:**

- Firewalls

  Consider strengthening the policies that are in place for IPv4 to support IPv6. For more security considerations, see "Security Considerations for the IPv6 Implementation" on page 70.
- Mail

  In the MX records for DNS, consider adding the IPv6 address of your mail server.
- DNS

  For DNS-specific considerations, see "How to Prepare DNS for IPv6 Support" on page 68.
- IPQoS

  Use the same Diffserv policies on a host that were used for IPv4. For more information, see "Classifier Module" on page 681.

5 **Audit any network services that are offered by a node prior to converting that node to IPv6.**

▼ How to Prepare DNS for IPv6 Support

The current Oracle Solaris release supports DNS resolution on both the client side and the server side. Do the following to prepare DNS services for IPv6.

For more information that is related to DNS support for IPv6, refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

1. **Ensure that the DNS server that performs recursive name resolution is dual-stacked (IPv4 and IPv6) or for IPv4 only.**
2. **On the DNS server, populate the DNS database with relevant IPv6 database AAAA records in the forward zone.**

Note – Servers that run multiple critical services require special attention. Ensure that the network is working properly. Also ensure that all critical services are ported to IPv6. Then, add the server's IPv6 address to the DNS database.

---

3. **Add the associated PTR records for the AAAA records into the reverse zone.**
4. **Add either IPv4 only data, or both IPv6 and IPv4 data into the NS record that describes zones.**

Planning for Tunnels in the Network Topology

The IPv6 implementation supports a number of tunnel configurations to serve as transition mechanisms as your network migrates to a mix of IPv4 and IPv6. Tunnels enable isolated IPv6 networks to communicate. Because most of the Internet runs IPv4, IPv6 packets from your site need to travel across the Internet through tunnels to destination IPv6 networks.

Here are some major scenarios for using tunnels in the IPv6 network topology:

- The ISP from which you purchase IPv6 service allows you to create a tunnel from your site's boundary router to the ISP network. Figure 3–1 shows such a tunnel. In such a case, you would run a manual, IPv6 over IPv4 tunnel.
- You manage a large, distributed network with IPv4 connectivity. To connect the distributed sites that use IPv6, you can run an automatic 6to4 tunnel from the edge router of each subnet.
- Sometimes, a router in your infrastructure cannot be upgraded to IPv6. In this case, you can create a manual tunnel over the IPv4 router, with two IPv6 routers as endpoints.

For procedures for configuring tunnels, refer to "Configuring Tunnels (Task Map)" on page 160. For conceptual information regarding tunnels, refer to "Overview of IP Tunnels" on page 151.

Security Considerations for the IPv6 Implementation

When you introduce IPv6 into an existing network, you must take care not to compromise the security of the site. Be aware of the following security issues as you phase in your IPv6 implementation:

- The same amount of filtering is required for both IPv6 packets and IPv4 packets.
- IPv6 packets are often tunneled through a firewall. Therefore, you should implement either of the following scenarios:
  - Have the firewall do content inspection inside the tunnel.
  - Put an IPv6 firewall with similar rules at the opposite tunnel endpoint.
- Some transition mechanisms exist that use IPv6 over UDP over IPv4 tunnels. These mechanisms might prove dangerous by short-circuiting the firewall.
- IPv6 nodes are globally reachable from outside the enterprise network. If your security policy prohibits public access, you must establish stricter rules for the firewall. For example, consider configuring a stateful firewall.

This book includes security features that can be used within an IPv6 implementation.

- The IP security architecture (IPsec) feature enables you to provide cryptographic protection for IPv6 packets. For more information, refer to Chapter 18, "IP Security Architecture (Overview)."
- The Internet Key Exchange (IKE) feature enables you to use public key authentication for IPv6 packets. For more information, refer to Chapter 21, "Internet Key Exchange (Overview)."

Preparing an IPv6 Addressing Plan

A major part of the transition from IPv4 to IPv6 includes the development of an addressing plan. This task involves the following preparations:

- "Obtaining a Site Prefix" on page 70
- "Creating the IPv6 Numbering Scheme" on page 71

Obtaining a Site Prefix

Before you configure IPv6, you must obtain a site prefix. The site prefix is used to derive IPv6 addresses for all the nodes in your IPv6 implementation. For an introduction to site prefixes, refer to "Prefixes in IPv6" on page 55.

Any ISP that supports IPv6 can provide your organization with a 48-bit IPv6 site prefix. If your current ISP only supports IPv4, you can use another ISP for IPv6 support while retaining your

current ISP for IPv4 support. In such an instance, you can use one of several workarounds. For more information, see "Current ISP Does Not Support IPv6" on page 175.

If your organization is an ISP, then you obtain site prefixes for your customers from the appropriate Internet registry. For more information, see the Internet Assigned Numbers Authority (IANA) (`http://www.iana.org`).

Creating the IPv6 Numbering Scheme

Unless your proposed IPv6 network is entirely new, use your existing IPv4 topology as the basis for the IPv6 numbering scheme.

Creating a Numbering Scheme for Subnets

Begin your numbering scheme by mapping your existing IPv4 subnets into equivalent IPv6 subnets. For example, consider the subnets illustrated in Figure 3–1. Subnets 1–4 use the RFC 1918 IPv4 private address designation for the first 16 bits of their addresses, in addition to the digits 1–4 to indicate the subnet. For illustrative purposes, assume that the IPv6 prefix `2001:db8:3c4d/48` has been assigned to the site.

The following table shows how the private IPv4 prefixes map into IPv6 prefixes.

IPv4 Subnet Prefix	Equivalent IPv6 Subnet Prefix
`192.168.1.0/24`	`2001:db8:3c4d:1::/64`
`192.168.2.0/24`	`2001:db8:3c4d:2::/64`
`192.168.3.0/24`	`2001:db8:3c4d:3::/64`
`192.168.4.0/24`	`2001:db8:3c4d:4::/64`

Creating an IPv6 Addressing Plan for Nodes

For most hosts, stateless autoconfiguration of IPv6 addresses for their interfaces is an appropriate, time saving strategy. When the host receives the site prefix from the nearest router, Neighbor Discovery automatically generates IPv6 addresses for each interface on the host.

Servers need to have stable IPv6 addresses. If you do not manually configure a server's IPv6 addresses, a new IPv6 address is autoconfigured whenever a NIC card is replaced on the server. Keep the following tips in mind when you create addresses for servers:

- Give servers meaningful and stable interface IDs. One strategy is to use a sequential numbering scheme for interface IDs. For example, the internal interface of the LDAP server in Figure 3–1 might become `2001:db8:3c4d:2::2`.

- Alternatively, if you do not regularly renumber your IPv4 network, consider using the existing IPv4 addresses of the routers and servers as their interface IDs. In Figure 3–1, suppose Router 1's interface to the DMZ has the IPv4 address `123.456.789.111`. You can convert the IPv4 address to hexadecimal and use the result as the interface ID. The new interface ID would be `::7bc8:156F`.

  Only use this approach if you own the registered IPv4 address, rather than having obtained the address from an ISP. If you use an IPv4 address that was given to you by an ISP, you create a dependency that would create problems if you change ISPs.

Due to the limited number of IPv4 addresses, in the past a network designer had to consider where to use global, registered addresses and private, RFC 1918 addresses. However, the notion of global and private IPv4 addresses does not apply to IPv6 addresses. You can use global unicast addresses, which include the site prefix, on all links of the network, including the public DMZ.

◆ ◆ ◆ CHAPTER 4

Configuring TCP/IP Network Services and IPv4 Addressing (Tasks)

TCP/IP network administration evolves in two stages. The first stage is to assemble the hardware. Then, you configure the daemons, files, and services that implement the TCP/IP protocol.

This chapter explains how to configure TCP/IP on a network that implements IPv4 addressing and services.

Note – Many of the tasks in this chapter apply to both IPv4-only and IPv6-enabled networks. Where configuration tasks differ between the two addressing formats, the IPv4 configuration steps are in this chapter. The tasks in this chapter then cross reference the equivalent IPv6 tasks in Chapter 5, "Enabling IPv6 on a Network (Tasks)."

This chapter contains the following information:

Before You Configure an IPv4 Network (Task Map)

Before you configure TCP/IP, complete the tasks that are listed in the following table. The table includes a description of what each task accomplishes and the section in the current documentation where the specific steps to perform the task are detailed.

Task	Description	For Instructions
1. Design the network topology.	Determine the physical layout of the network.	"Network Topology Overview" on page 46 and "IPv4 Autonomous System Topology" on page 88
2. Obtain a network number from your ISP or Regional Internet Registry (RIR).	Get a registered network number, which enables systems at your site to communicate externally.	"Designing Your IPv4 Addressing Scheme" on page 39.
3. Plan the IPv4 addressing scheme for the network. If applicable, include subnet addressing.	Use the network number as the basis for your addressing plan.	"Designing Your IPv4 Addressing Scheme" on page 39.
4. Assemble the network hardware depending on the network topology. Assure that the hardware is functioning properly.	Set up the systems, network media, routers, switches, hubs and bridges that you outlined in the network topology design.	The hardware manuals and "Network Topology Overview" on page 46.
5. Assign IPv4 addresses and host names to all systems in the network.	Assign the IPv4 addresses during Oracle Solaris installation or post installation, in the appropriate files.	"Designing Your IPv4 Addressing Scheme" on page 39 and "How to Change the IPv4 Address and Other Network Configuration Parameters" on page 83
6. Run configuration software that is required by network interfaces and routers, if applicable.	Configure routers and multihomed hosts.	"Planning for Routers on Your Network" on page 45 and "Configuring an IPv4 Router" on page 90 for information on routers.
7. Determine which name service or directory service your network uses: NIS, LDAP, DNS, or local files.	Configure your selected name service and/or directory service.	*System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.
8. Select domain names for your network, if applicable.	Choose a domain name for your network and register it with the InterNIC.	*System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*

Determining Host Configuration Modes

As a network administrator, you configure TCP/IP to run on hosts and routers (if applicable). You can configure these systems to obtain configuration information from files on the local system or from files that are located on other systems on the network. You need the following configuration information:

- Host name of each system
- IP address of each system
- Domain name to which each system belongs

- Default router
- IPv4 netmask in use on each system's network

A system that obtains TCP/IP configuration information from local files operates in *local files mode*. A system that obtains TCP/IP configuration information from a remote network server operates in *network client mode*.

Systems That Should Run in Local Files Mode

To run in local files mode, a system must have local copies of the TCP/IP configuration files. These files are described in "TCP/IP Configuration Files" on page 177. The system should have its own disk, though this recommendation is not strictly necessary.

Most servers should run in local files mode. This requirement includes the following servers:

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

Additionally, routers should run in local files mode.

Systems that function exclusively as print servers do not need to run in local files mode. Whether individual hosts should run in local files mode depends on the size of your network.

If you are running a very small network, the amount of work that is involved in maintaining these files on individual hosts is manageable. If your network serves hundreds of hosts, the task becomes difficult, even with the network divided into a number of administrative subdomains. Thus, for large networks, using local files mode is usually less efficient. However, because routers and servers must be self-sufficient, they should be configured in local files mode.

Network Configuration Servers

Network configuration servers are the servers that supply the TCP/IP configuration information to hosts that are configured in network client mode. These servers support three booting protocols:

- RARP – Reverse Address Resolution Protocol (RARP) maps Ethernet addresses (48 bits) to IPv4 addresses (32 bits), which is the reverse of ARP. When you run RARP on a network configuration server, hosts that are running in network client mode obtain their IP addresses and TCP/IP configuration files from the server. The `in.rarpd` daemon enables RARP services. Refer to the `in.rarpd(1M)` man page for details.
- TFTP – The Trivial File Transfer Protocol (TFTP) is an application that transfers files between remote systems. The `in.tftpd` daemon executes TFTP services, enabling file transfer between network configuration servers and their network clients. Refer to the `in.tftpd(1M)` man page for details.

- Bootparams – The Bootparams protocol supplies parameters for booting that are required by clients that boot off the network. The `rpc.bootparamd` daemon executes these services. Refer to the `bootparamd(1M)` man page for details.

Network configuration servers can also function as NFS file servers.

If you are configuring any hosts as network clients, then you must also configure at least one system on your network as a network configuration server. If your network is subnetted, then you must have at least one network configuration server for each subnet with network clients.

Systems That Are Network Clients

Any host that obtains its configuration information from a network configuration server operates in network client mode. Systems that are configured as network clients do not require local copies of the TCP/IP configuration files.

Network client mode simplifies administration of large networks. Network client mode minimizes the number of configuration tasks that you perform on individual hosts. Network client mode assures that all systems on the network adhere to the same configuration standards.

You can configure network client mode on all types of computers. For example, you can configure network client mode on standalone systems.

Mixed Configurations

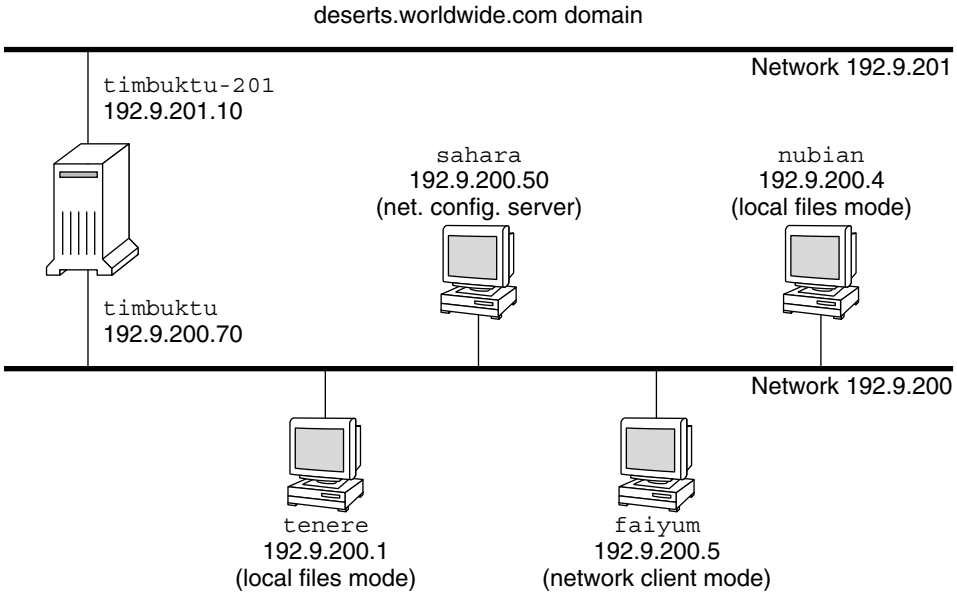
Configurations are not limited to either an all-local-files mode or an all-network-client mode. Routers and servers should always be configured in local mode. For hosts, you can use any combination of local files and network client mode.

IPv4 Network Topology Scenario

Figure 4–1 shows the hosts of a fictitious network with the network number `192.9.200`. The network has one network configuration server, which is called `sahara`. Hosts `tenere` and `nubian` have their own disks and run in local files mode. Host `faiyum` also has a disk, but this system operates in network client mode.

Finally, the system `timbuktu` is configured as a router. The system includes two network interfaces. The first interface is named `timbuktu`. This interface belongs to network `192.9.200`. The second interface is named `timbuktu-201`. This interface belongs to network `192.9.201`. Both networks are in the organizational domain `deserts.worldwide.com`. The domain uses local files as its name service.

FIGURE 4–1 Hosts in an IPv4 Network Topology Scenario



Adding a Subnet to a Network (Task Map)

If you are changing from a network that does not use a subnet to a network that does use a subnet, perform the tasks in the following task map.

Note – The information in this section applies to IPv4 subnets only. For information on planning IPv6 subnets, refer to "Preparing the Network Topology for IPv6 Support" on page 66 and "Creating a Numbering Scheme for Subnets" on page 71.

The following table lists different tasks for adding a subnet to the current network. The table includes a description of what each task accomplishes and the section in the current documentation where the specific steps to perform the task are detailed.

Task	Description	For Instructions
1. Determine if your network topology requires subnets.	Decide on the new subnet topology, including where to locate routers and hosts on the subnets.	"Planning for Routers on Your Network" on page 45, "What Is Subnetting?" on page 181, and "Network Classes" on page 193

Task	Description	For Instructions
2. Assign the IP addresses with the new subnet number to the systems to become members of the subnet.	Configure IP addresses that use the new subnet number, either during Oracle Solaris installation or later, in the `/etc/hostname.`*interface* file.	"Deciding on an IP Addressing Format for Your Network" on page 35
3. Configure the network mask of the subnet on all prospective systems in the subnet.	Modify the `/etc/inet/netmasks` file, if you are manually configuring network clients. Or, supply the netmask to the Oracle Solaris installation program.	"`netmasks` Database" on page 181 and "Creating the Network Mask for IPv4 Addresses" on page 182
4. Edit the network databases with the new IP addresses of all systems in the subnet.	Modify `/etc/inet/hosts` on all hosts to reflect the new host addresses.	"`hosts` Database" on page 178
5. Reboot all systems.		

Network Configuration Task Map

The following table lists additional tasks to perform after changing from a network configuration without subnets to a network that uses subnets. The table includes a description of what each task accomplishes and the section in the current documentation where the specific steps to perform the task are detailed.

Task	Description	For Instructions
Configure a host for local files mode	Involves editing the `nodename`, `hostname`, `hosts`, `defaultdomain`, `defaultrouter`, and `netmasks` files	"How to Configure a Host for Local Files Mode" on page 79
Set up a network configuration server	Involves turning on the `in.tftp` daemon, and editing the `hosts`, `ethers`, and `bootparams` files	"How to Set Up a Network Configuration Server" on page 81
Configure a host for network client mode	Involves creating the `hostname` file, editing the `hosts` file, and deleting the `nodename` and `defaultdomain` files, if they exist	"How to Configure Hosts for Network Client Mode" on page 82
Specify a routing strategy for the network client	Involves determining whether to use static routing or dynamic routing on the host.	"How to Enable Static Routing on a Single-Interface Host" on page 99 and "How to Enable Dynamic Routing on a Single-Interface Host" on page 101.

Task	Description	For Instructions
Modify the existing network configuration	Involves changing the host name, IP address, and other parameters that were set at installation or configured at a later time.	"How to Change the IPv4 Address and Other Network Configuration Parameters" on page 83

Configuring Systems on the Local Network

Network software installation occurs along with the installation of the operating system software. At that time, certain IP configuration parameters must be stored in appropriate files so that they can be read at boot time.

The network configuration process involves creating or editing the network configuration files. How configuration information is made available to a system's kernel is conditional. The availability depends on whether these files are stored locally (local files mode) or acquired from the network configuration server (network client mode).

The parameters that are supplied during network configuration follow:

- The IP address of each network interface on every system.
- The host names of each system on the network. You can type the host name in a local file or a name service database.
- The NIS, LDAP, or DNS domain name in which the system resides, if applicable.
- The default router addresses. You supply this information if you have a simple network topology with only one router attached to each network. You also supply this information if your routers do not run routing protocols such as the Router Discovery Server Protocol (RDISC) or the Router Information Protocol (RIP). For more information on default routers, refer to "Packet Forwarding and Routing on IPv4 Networks" on page 84 See Table 4–1 for a list of routing protocols supported in Oracle Solaris.
- Subnet mask (required only for networks with subnets).

If the Oracle Solaris installation program detects more than one interface on the system, you can optionally configure the additional interfaces during installation. For complete instructions, see *Getting Started With Oracle Solaris 11 Express*.

This chapter contains information on creating and editing local configuration files. See *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on working with name service databases.

▼ How to Configure a Host for Local Files Mode

Use this procedure for configuring TCP/IP on a host that runs in local files mode.

For steps for manually configuring interfaces in Oracle Solaris, refer to "How to Configure an IP Interface" in *System Administration Guide: Network Interfaces and Network Virtualization*.

1. **Change to the `/etc` directory.**

2. **Verify that the correct host name is set in the `/etc/nodename` file.**

   When you specify the host name of a system during Oracle Solaris installation, that host name is entered into the `/etc/nodename` file. Make sure that the node name entry is the correct host name for the system.

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

   The Oracle Solaris installation program creates entries for the primary network interface, loopback address, and, if applicable, any additional interfaces that were configured during installation.

   a. **Make sure that the existing entries in `/etc/inet/hosts` are current.**

   b. **(Optional) Add the IP addresses and corresponding names for any network interfaces that were added to the local host after installation.**

   c. **(Optional) Add the IP address or addresses of the file server, if the `/usr` file system is NFS mounted.**

4. **Type the host's fully qualified domain name in the `/etc/defaultdomain` file.**

   For example, suppose host `tenere` was part of the domain `deserts.worldwide.com`. Therefore, you would type `deserts.worldwide.com` in `/etc/defaultdomain`. See "`/etc/defaultdomain` File" on page 178 for more information.

5. **Type the router's name in the `/etc/defaultrouter` file.**

   See "`/etc/defaultrouter` File" on page 178 for information about this file.

6. **Type the name of the default router and its IP addresses in the `/etc/inet/hosts` file.**

   Additional routing options are available, as discussed in "How to Configure Hosts for Network Client Mode" on page 82. You can apply these options to a local files mode configuration.

7. **Add the network mask for your network, if applicable:**

   - If the host gets its IP address from a DHCP server, you do not have to specify the network mask.
   - If you have set up a NIS server on the same network as this client, you can add `netmask` information into the appropriate database on the server.

- For all other conditions, do the following:

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

Use the following format:

```
network-number netmask
```

For example, for the Class C network number `192.168.83`, you would type:

```
192.168.83.0    255.255.255.0
```

For CIDR addresses, convert the network prefix into the equivalent dotted decimal representation. Network prefixes and their dotted decimal equivalents can be found in Table 1–3. For example, use the following to express the CIDR network prefix `192.168.3.0/22`.

```
192.168.3.0 255.255.252.0
```

b. Change the lookup order for netmasks in `/etc/nsswitch.conf`, so that local files are searched first:

```
netmasks:   files nis
```

8 Reboot the system.

How to Set Up a Network Configuration Server

Information for setting up installation servers and boot servers is found in *Getting Started With Oracle Solaris 11 Express*.

1 Change to the root (`/`) directory of the prospective network configuration server.

2 Turn on the `in.tftpd` daemon by creating the directory `/tftpboot`:

```
# mkdir /tftpboot
```

This command configures the system as a TFTP, bootparams, and RARP server.

3 Create a symbolic link to the directory.

```
# ln -s /tftpboot/. /tftpboot/tftpboot
```

4 Enable the `tftp` line in the `/etc/inetd.conf` file.

Check that the entry reads as follows:

```
tftp dgram udp6 wait root /usr/sbin/in.tftpd in.tftpd -s /tftpboot
```

This line prevents `in.tftpd` from retrieving any file other than the files that are located in `/tftpboot`.

5 **Edit the `hosts` database.**

Add the host names and IP addresses for every client on the network.

6 **Edit the `ethers` database.**

Create entries for every host on the network that runs in network client mode.

7 **Edit the `bootparams` database.**

See "`bootparams` Database" on page 189. Use the wildcard entry or create an entry for every host that runs in network client mode.

8 **Convert the `/etc/inetd.conf` entry into a Service Management Facility (SMF) service manifest, and enable the resulting service:**

```
# /usr/sbin/inetconv
```

9 **Verify that `in.tftpd` is working correctly.**

```
# svcs network/tftp/udp6
```

You should receive output resembling the following:

```
STATE          STIME    FMRI
online         18:22:21 svc:/network/tftp/udp6:default
```

More Information

Administering the `in.tftpd` Daemon

The `in.tftpd` daemon is managed by the Service Management Facility. Administrative actions on `in.tftpd`, such as enabling, disabling, or restarting, can be performed using the `svcadm` command. Responsibility for initiating and restarting this service is delegated to `inetd`. Use the `inetadm` command to make configuration changes and to view configuration information for `in.tftpd`. You can query the service's status by using the `svcs` command. For an overview of the Service Management Facility, refer to Chapter 11, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*.

Configuring Network Clients

Network clients receive their configuration information from network configuration servers. Therefore, before you configure a host as a network client you must ensure that at least one network configuration server is set up for the network.

▼ How to Configure Hosts for Network Client Mode

Do the following procedure on each host to be configured in network client mode.

1. **Ensure that the `/etc/inet/hosts` file contains only the `localhost` name and IP address of the loopback network interface.**

   ```
   # cat /etc/inet/hosts
   # Internet host table
   #
   127.0.0.1       localhost
   ```

   The IPv4 loopback interface has the IP address `127.0.0.1`.

   For more information, see "Loopback Address" on page 179. The file should not contain the IP address and host name for the local host (primary network interface).

2. **Check for the existence of an `/etc/defaultdomain` file.**

   If such a file exists, delete it.

   The `hostconfig` program automatically sets the domain name. To override the domain name that is set by `hostconfig`, type the substitute domain name in the `/etc/defaultdomain` file.

3. **Ensure that the search paths in the client's `/etc/nsswitch.conf` file reflect the name service requirements for your network.**

▼ How to Change the IPv4 Address and Other Network Configuration Parameters

This procedure explains how to modify the IPv4 address, host name, and other network parameters on a previously installed system. Use the procedure for modifying the IP address of a server or networked standalone system. The procedure does not apply to network clients or appliances. The steps create a configuration that persists across reboots.

Note – The instructions apply specifically to changing the IPv4 address of the primary network interface. To add another interface to the system, refer to "How to Configure an IP Interface" in *System Administration Guide: Network Interfaces and Network Virtualization*.

In almost all cases, the following steps use traditional IPv4 dotted decimal notation to specify the IPv4 address and subnet mask. Alternatively, you can use CIDR notation to specify the IPv4 address in all the applicable files in this procedure. For an introduction to CIDR notation, see "IPv4 Addresses in CIDR Format" on page 36.

1. **If the system's host name must change, modify the host name entry by using the `svccfg` command, as follows:.**

   ```
   # svccfg -s svc:/system/identity:node setprop config/nodename = astring: hostname
   ```

2. **Modify the IP address and, if applicable, the host name in the `/etc/inet/hosts` file or equivalent `hosts` database.**

3. **Modify the IP address by using the `ipadm` command.**

   ```
   # ipadm
   ```

4. **If the subnet mask has changed, modify the subnet entries in the following files:**

   - `/etc/netmasks`
   - (Optional) `/etc/hostname.`*interface*

5. **If the subnet address has changed, change the IP address of the default router in `/etc/defaultrouter` to that of the new subnet's default router.**

6. **Reboot the system.**

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

Example 4–1 Changing the IP Address and Host Name

This example shows how to change a host's name, IP address of the primary network interface, and subnet mask. The IP address for the primary network interface `bge0` changes from `10.0.0.14` to `192.168.34.100`.

```
# ipadm show-addr
ADDROBJ      TYPE     STATE   ADDR
lo0/v4       static   ok      127.0.0.1/8
bge0/v4      static   ok      10.0.0.14/24

# ipadm create-addr -T static -a 192.168.34.100/24 bge0/v4new
# svccfg -s svc:/system/identity:node setprop config/nodename = astring: mynewhostname

# ipadm show-addr
ADDROBJ         TYPE     STATE   ADDR
lo0/v4          static   ok      127.0.0.1/8
bge0/v4new      static   ok      192.168.34.100/24

# hostname
mynewhostname
```

See Also To change the IP address of an interface other than the primary network interface, refer to *System Administration Guide: Basic Administration* and "How to Configure an IP Interface" in *System Administration Guide: Network Interfaces and Network Virtualization*.

Packet Forwarding and Routing on IPv4 Networks

This section contains procedures and examples that show how to configure forwarding and routing for routers and hosts on IPv4 networks.

Packet forwarding is the basic method for sharing information across systems on a network. Packets are transferred between a source interface and a destination interface, usually on two different systems. When you issue a command or send a message to a nonlocal interface, your

system forwards those packets onto the local network. The interface with the destination IP address that is specified in the packet headers then retrieves the packets from the local network. If the destination address is not on the local network, the packets are then forwarded to the next adjacent network, or *hop*. By default, packet forwarding is automatically configured when you install Oracle Solaris.

Routing is the process by which systems decide where to send a packet. Routing protocols on a system "discover" the other systems on the local network. When the source system and the destination system are on the same local network, the path that packets travel between them is called a *direct route*. If a packet must travel at least one hop beyond its source system, the path between the source system and destination system is called an *indirect route*. The routing protocols learn the path to a destination interface and retain data about known routes in the system's *routing table*.

Routers are specially configured systems with multiple physical interfaces that connect the router to more than one local network. Therefore, the router can forward packets beyond the home LAN, regardless of whether the router runs a routing protocol. For more information about how routers forward packets, refer to "Planning for Routers on Your Network" on page 45.

Routing protocols handle routing activity on a system and, by exchanging routing information with other hosts, maintain known routes to remote networks. Both routers and hosts can run routing protocols. The routing protocols on the host communicate with routing daemons on other routers and hosts. These protocols assist the host in determining where to forward packets. When network interfaces are enabled, the system automatically communicates with the routing daemons. These daemons monitor routers on the network and advertise the routers' addresses to the hosts on the local network. Some routing protocols, though not all, also maintain statistics that you can use to measure routing performance. Unlike packet forwarding, you must explicitly configure routing on an Oracle Solaris system.

This section contains tasks for administering packet forwarding and routing on IPv4 routers and hosts. For information about routing on an IPv6-enabled network, refer to "Configuring an IPv6 Router" on page 112.

Routing Protocols Supported by Oracle Solaris

Routing protocols are classified as interior gateway protocols (IGPs), exterior gateway protocols (EGPs), or a combination of both. *Interior gateway protocols* exchange routing information between routers on networks under common administrative control. In the network topology shown in Figure 4–3, the routers run an IGP for exchanging routing information. *Exterior gateway protocols* enable the router that connects the local internetwork to an external network to exchange information with another router on the external network. For example, the router that connects a corporate network to an ISP runs an EGP to exchange routing information with its router counterpart at the ISP. Border Gateway Protocol (BGP) is a popular EGP that is used for carrying routing information between different organizations and IGPs.

The following table provides information about the Oracle Solaris routing protocols and the location of each protocol's associated documentation.

TABLE 4–1 Oracle Solaris Routing Protocols

Protocol	Associated Daemon	Description	For Instructions
Routing Information Protocol (RIP)	`in.routed`	IGP that routes IPv4 packets and maintains a routing table	"How to Configure an IPv4 Router" on page 91
Internet Control Message Protocol (ICMP) Router Discovery	`in.routed`	Used by hosts to discover the presence of a router on the network	"How to Enable Static Routing on a Single-Interface Host" on page 99 and "How to Enable Dynamic Routing on a Single-Interface Host" on page 101
Routing Information Protocol, next generation (RIPng) Protocol	`in.ripngd`	IGP that routes IPv6 packets and maintains a routing table	"How to Configure an IPv6-Enabled Router" on page 113
Neighbor Discovery (ND) Protocol	`in.ndpd`	Advertises the presence of an IPv6 router and discovers the presence of IPv6 hosts on a network	"Configuring an IPv6 Interface" on page 109

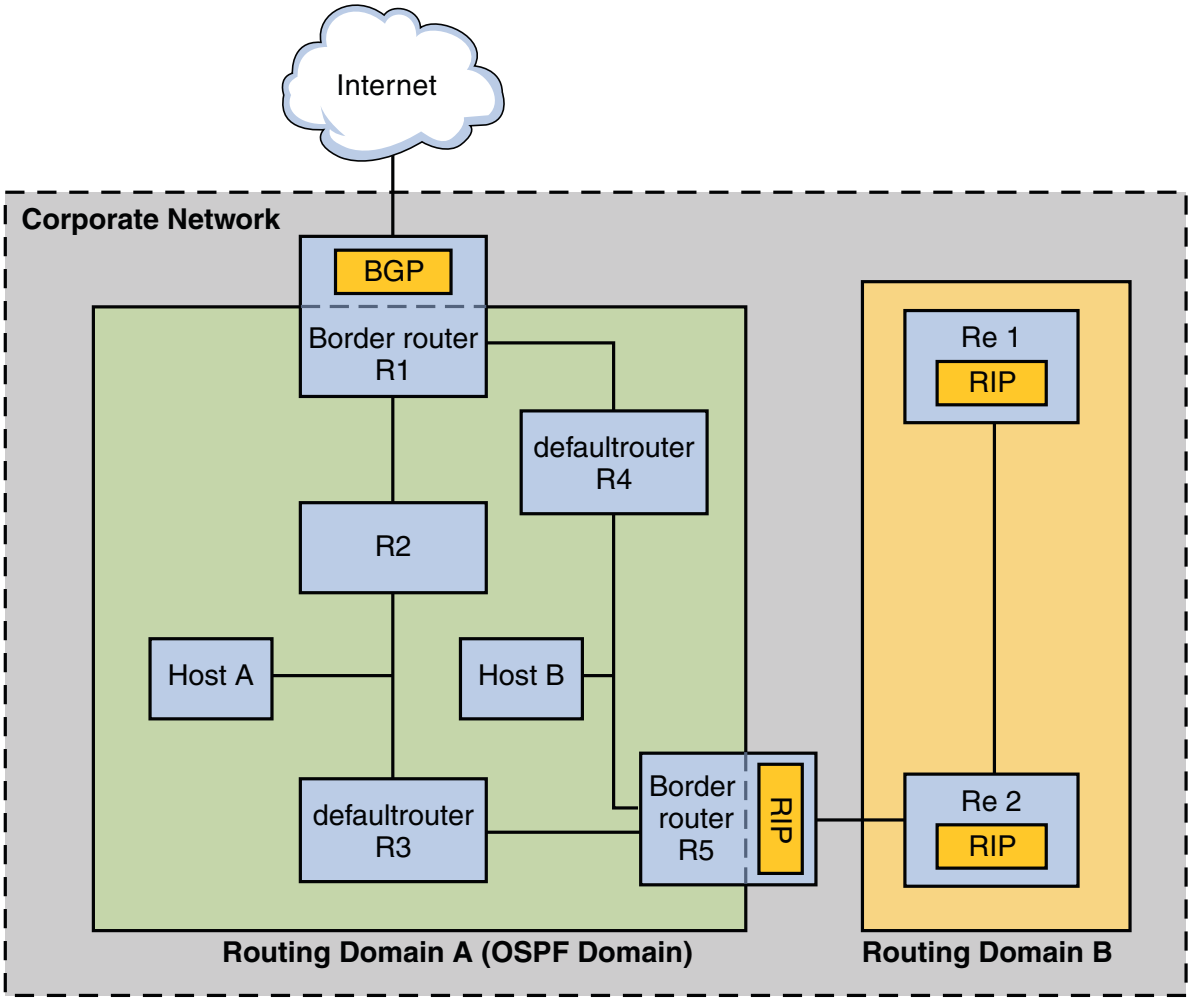
Oracle Solaris also supports the Open Source Quagga routing protocol suite. These protocols are available from the SFW consolidation disk, though they are not part of the mainOracle Solaris distribution. The following table lists the Quagga protocols:

TABLE 4–2 OpenSolaris Quagga Protocols

Protocol	Daemon	Description
RIP protocol	`ripd`	IPv4 distance vectoring IGP that routes IPv4 packets and advertises its routing table to neighbors.
RIPng	`ripngd`	IPv6 distance vectoring IGP. Routes IPv6 packets and maintains a routing table.
Open Shortest Path First (OSPF) protocol	`ospfd`	IPv4 link state IGP for packet routing and high availability networking
Border Gateway Protocol (BGP)	`bgpd`	IPv4 and IPv6 EGP for routing across administrative domains.

The following figure shows an autonomous system that uses the Quagga routing protocols:

FIGURE 4–2 Corporate Network That Runs Quagga Protocols



The figure shows a corporate network autonomous system that is subdivided into two routing domains, A and B. A*routing domain* is an internetwork with a cohesive routing policy, either for administrative purposes or because the domain uses a single routing protocol. Both domains in the figure run routing protocols from the Quagga protocol suite.

Routing Domain A is an OSPF domain, which is administered under a single OSPF domain ID. All systems within this domain run OSPF as their interior gateway protocol. In addition to internal hosts and routers, Domain A includes two border routers.

Border router R1 connects the Corporate Network to an ISP and ultimately the Internet. To facilitate communications between the Corporate Network and the outside world, R1 runs BGP

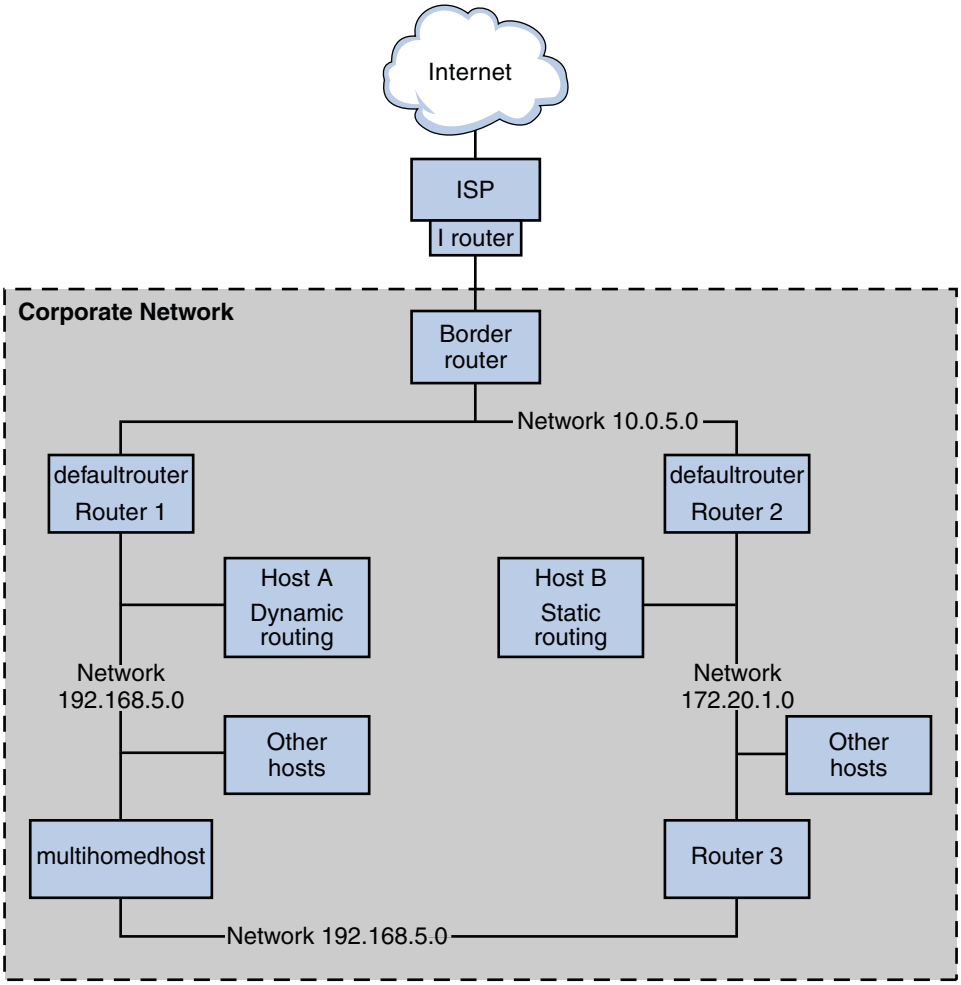
over its externally facing network interface. The border router R5 connects Domain A with Domain B. All systems on Domain B are administered with RIP as their interior gateway protocol. Therefore, border router R5 must run OSPF on the Domain A facing interface and RIP on the Domain B facing interface.

For more information on the Quagga protocols, refer to the Open Solaris Quagga (`http://hub.opensolaris.org/bin/view/Project+quagga/`). For configuration procedures for these protocols, go to the documentation for quagga (`http://quagga.net/docs/docs-info.php`).

IPv4 Autonomous System Topology

Sites with multiple routers and networks typically administer their network topology as a single routing domain, or *autonomous system (AS)* . The following figure shows a typical network topology that would be considered a small AS. This topology is referenced in the examples throughout this section.

FIGURE 4–3 Autonomous System With Multiple IPv4 Routers



The figure shows an AS that is divided into three local networks, `10.0.5.0`, `172.20.1.0`, and `192.168.5`. Four routers share packet-forwarding and routing responsibilities. The AS includes the following types of systems:

- *Border routers* connect an AS to an external network, such as the Internet. Border routers interconnect with networks external to the IGP running on the local AS. A border router can run an EGP, such as Border Gateway Protocol (BGP), to exchange information with external routers, for example, the routers at the ISP. In Figure 4–3, the border router's interfaces connect to internal network `10.0.5.0` and to a high-speed router to a service provider.

For information on configuring a border router, refer to the Open Source Quagga documentation (`http://www.quagga.net/docs/docs-info.php#SEC72`)for BGP information.

If you plan to use BGP to connect your AS to the Internet, you should obtain an autonomous system number (ASN) from the Internet Registry for your locale. Regional registries, such as the American Registry for Internet Numbers (ARIN), offer guidelines on how to obtain an ASN. For example, the ARIN Number Resource Policy Manual (`https://www.arin.net/policy/nrpm.html#five`) contains instructions for getting an ASN for autonomous systems in the United States and Canada. Alternatively, your ISP might be able to obtain an ASN for you.

- *Default routers* maintain routing information about all the systems on the local network. These routers typically run IGPs such as RIP. In Figure 4–3, Router 1s interfaces are connected to internal network `10.0.5.0` and internal network `192.168.5`. Router 1 also serves as the default router for `192.168.5`. Router 1 maintains routing information for all systems on `192.168.5` and routes to other routers, such as the border router. Router 2s interfaces connect to internal network `10.0.5.0` and internal network `172.20.1`.

  For an example of configuring a default router, refer to Example 4–2.
- *Packet-forwarding routers* forward packets but do not run routing protocols. This type of router receives packets from one of its interfaces that is connected to a single network. These packets are then forwarded through another interface on the router to another local network. In Figure 4–3, Router 3 is a packet-forwarding router with connections to networks `172.20.1` and `192.168.5`.
- *Multihomed hosts* have two or more interfaces that are connected to the same network segment. A multihomed host can forward packets, which is the default for all systems that run Oracle Solaris. Figure 4–3 shows a multihomed host with both interfaces connected to network `192.168.5`. For an example of configuring a multihomed host, refer to Example 4–4.
- *Single interface hosts* rely on the local routers, not only for packet forwarding but also for receiving valuable configuration information. Figure 4–3 includes Host A on the `192.168.5` network, which implements dynamic routing, and Host B on the `172.20.1` network, which implements static routing. To configure a host to run dynamic routing, refer to "How to Enable Dynamic Routing on a Single-Interface Host" on page 101. To configure a host to run static routing, refer to "How to Enable Static Routing on a Single-Interface Host" on page 99.

Configuring an IPv4 Router

This section contains a procedure and example for configuring an IPv4 router. To configure an IPv6-enabled router, refer to "How to Configure an IPv6-Enabled Router" on page 113.

Because a router provides the interface between two or more networks, you must assign a unique name and IP address to each of the router's physical network interfaces. Thus, each

router has a host name and an IP address that are associated with its primary network interface, in addition to a minimum of one more unique name and IP address for each additional network interface.

You can also use the following procedure to configure a system with only one physical interface (by default, a host) to be a router. You might configure a single interface system as a router if the system serves as one endpoint on a PPP link, as explained in "Planning a Dial-up PPP Link" in *System Administration Guide: Network Services*.

Note – You can configure all interfaces of a router during Oracle Solaris system installation. For instructions, see *Getting Started With Oracle Solaris 11 Express*.

How to Configure an IPv4 Router

The following instructions assume that you are configuring interfaces for the router after installation.

Before You Begin After the router is physically installed on the network, configure the router to operate in local files mode, as described in "How to Configure a Host for Local Files Mode" on page 79. This configuration ensures that routers boot if the network configuration server is down.

1 Use the `dladm show-link` command to determine which interfaces are physically installed on the router.

```
# dladm show-link
```

The following example output from `dladm show-link` indicates that a `qfe` NIC with four interfaces and two `bge` interfaces are physically available on the system.

```
LINK      CLASS     MTU      STATE   BRIDGE   OVER
e1000g0   phys      1500     up      --       --
e1000g1   phys      1500     up      --       --
e1000g2   phys      1500     up      --       --
nge0      phys      1500     up      --       --
bge0      phys      1500     up      --       --
bge1      phys      1500     up      --       --
```

2 Review which interfaces on the router were configured and plumbed during installation.

```
# ipadm show-addr
```

The following example output from `ipadm show-addr` shows that the interface `e1000g0` was configured during installation. This interface is on the `172.16.0.0` network. The `nge` interface and the `bge` interfaces have not been configured.

```
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
e1000g0/v4        static   ok           172.16.26.232/24
```

3 Configure another interface with a valid IP address.

```
# ipadm create-addr -T static -a address addrobj
```

where *addrobj* is an identifier of the interface and its corresponding address. The *addrobj* must use the naming format *interface/user-defined-string*.

For example, to assign the IP address `192.168.84.3` to `e1000g1`, type the following:

```
# ipadm create-addr -T static -a 192.168.84.3/24 e1000g1/v4
```

Caution – You can configure an IPv4 routers to receive its IP address through DHCP, but this is recommended only for very experienced DHCP system administrators.

For more information about configuring single interfaces, refer to "How to Configure an IP Interface" in *System Administration Guide: Network Interfaces and Network Virtualization*. For more information about the `ipadm` command, see the `ipadm`(1M) man page.

4 Add the host name and IP address of each interface to the `/etc/inet/hosts` file.

For example:

```
172.16.26.232      deadsea        #interface for network 172.16.0.0
192.168.200.20     timbuktu       #interface for network 192.168.200
192.168.201.20     timbuktu-201   #interface for network 192.168.201
192.168.200.9      gobi
192.168.200.10     mojave
192.168.200.110    saltlake
192.168.200.12     chilean
```

The interfaces `timbuktu` and `timbuktu-201` are on the same system. Notice that the network address for `timbuktu-201` is different from the network interface for `timbuktu`. The difference exists because the physical network media for network `192.168.201` is connected to the `timbuktu-201` network interface while the media for network `192.168.200` is connected to the `timbuktu` interface.

5 Enable IPv4 packet forwarding on the router.

Use either of the following commands to enable packet forwarding:

```
# ipadm set-prop -p forwarding=on ipv4
```

At this point, the router can forward packets beyond the local network. The router also supports *static routing*, a process where you can manually add routes to the routing table. If you plan to use static routing on this system, then router configuration is complete. However, you need to maintain routes in the system routing table. For information on adding routes, see "Configuring Routes" on page 95 and the `route`(1M) man page.

6 (Optional) Start a routing protocol.

The routing daemon `/usr/sbin/in.routed` automatically updates the routing table, a process that is known as *dynamic routing*. Turn on the default IPv4 routing protocols in either of the following ways:

- Use the `routeadm` command, as follows:

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

- Use the following SMF command to start a routing protocol such as RIP.

  ```
  # svcadm enable route:default
  ```

  The SMF FMRI associated with the `in.routed` daemon is `svc:/network/routing/route`.

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

Example 4–2 Configuring the Default Router for a Network

This example shows how to upgrade a system with more than one interface to become a default router. The goal is to make Router 2, which is shown in Figure 4–3, the default router for network `172.20.1.0`. Router 2 contains two wired network connections, one connection to network `172.20.1.0` and one to network `10.0.5.0`. The example assumes that the router operates in local files mode, as described in "How to Configure a Host for Local Files Mode" on page 79.

After becoming superuser or assuming an equivalent role, you would determine out the status of the system's interfaces.

```
# dladm show-link
LINK     CLASS     MTU     STATE   BRIDGE   OVER
e1000g0  phys      1500    up      --       --
bge0     phys      1500    up      --       --
bge1     phys      1500    up      --       --
# ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
e1000g0/v4        static   ok           172.20.1.10/24
```

The output of `dladm show-link` indicates that three links are available on the system. Only the `e1000g0` interface has been configured with an IP address. You would begin default router configuration by physically connecting the `bge0` interface to the `10.0.5.0` network.

```
# ipadm create-addr -T static -a 10.0.5.10/24 bge0/v4
# ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
e1000g0/v4        static   ok           172.20.1.10/24
bge0/v4           static   ok           10.0.5.10/24
```

Continue by configuring the following network databases with information about the newly plumbed interface and the network to which it is connected:

```
# vi /etc/inet/hosts
127.0.0.1       localhost
172.20.1.10        router2        #interface for network 172.20.1
10.0.5.10          router2-out    #interface for network 10.0.5
```

Finally, use SMF to enable packet forwarding and then enable the `in.routed` routing daemon.

```
# ipadm set-prop -p forwarding=on ipv4
# svcadm enable route:default
```

Now IPv4 packet forwarding and dynamic routing through RIP are enabled on Router 2. However, the default router configuration for network `172.20.1.0` is not yet complete. You would need to do the following:

- Modify each host on `172.10.1.10` so that the host gets its routing information from the new default router. For more information, refer to "How to Enable Static Routing on a Single-Interface Host" on page 99.
- Define a static route to the border router in the routing table of Router 2. For more details, refer to "Routing Tables and Routing Types" on page 94.

Routing Tables and Routing Types

Both routers and hosts maintain a *routing table*. The routing daemon on each system updates the table with all known routes. The system's kernel reads the routing table before forwarding packets to the local network. The routing table lists the IP addresses of networks that the system knows about, including the system's local, default network. The table also lists the IP address of a gateway system for each known network. The *gateway* is a system that can receive outgoing packets and forward them one hop beyond the local network. The following is a simple routing table for a system on an IPv4-only network:

```
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
default              172.20.1.10          UG        1    532   ce0
224.0.0.0            10.0.5.100           U         1      0   bge0
10.0.0.0             10.0.5.100           U         1      0   bge0
127.0.0.1            127.0.0.1            UH        1     57   lo0
```

You can configure two types of routing on an Oracle Solaris system: static and dynamic. You can configure either or both routing types on a single system. A system that implements *dynamic routing* relies on routing protocols, such as RIP for IPv4 networks, and RIPng for IPv6 networks, to maintain its routing tables. A system that runs only *static routing* does not rely on a routing protocol for routing information and for updating the routing table. Instead, you must maintain the system's known routes manually through the `route` command. For complete details, refer to the `route(1M)` man page.

When you configure routing for the local network or autonomous system, consider which type of routing to support on particular routers and hosts.

The following table shows the different types of routing and the networking scenarios to which each routing type is best applied.

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

The AS that is shown is Figure 4–3 combines both static and dynamic routing.

Configuring Routes

To implement dynamic routing for an IPv4 network, use the `routeadm` or `svcadm` command to start the `in.routed` routing daemon. For instructions, see "How to Configure an IPv4 Router" on page 91. Dynamic routing is the preferred strategy for most networks and autonomous systems. However, your network topology or a particular system on your network might require static routing. In that case, you must manually edit the system routing table to reflect the known route to the gateway. The next procedure shows how to add a static route.

Note – Two routes to the same destination does not automatically cause the system to do load balancing or failover. If you need these capabilities, use IPMP, as explained in Chapter 13, "Introducing IPMP," in *System Administration Guide: Network Interfaces and Network Virtualization*.

▼ How to Add a Static Route to the Routing Table

1 View the current state of the routing table.

Use your regular user account to run the following form of the `netstat` command:

```
% netstat -rn
```

Your output would resemble the following:

```
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.5.125        192.168.5.10          U      1   5879   ipge0
224.0.0.0            198.168.5.10          U      1  0       ipge0
default              192.168.5.10          UG     1  91908
127.0.0.1            127.0.0.1             UH     1  811302   lo0
```

2 (Optional) Flush the existing entries in the routing table.

```
# route flush
```

3 Add a route that persists across system reboots.

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

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

Example 4–3 Adding a Static Route to the Routing Table

The following example shows how to add a static route to a system. The system is Router 2, the default router for the `172.20.1.0` network that is shown in Figure 4–3. In Example 4–2, Router 2 is configured for dynamic routing. To better serve as the default router for the hosts on network `172.20.1.0`, Router 2 additionally needs a static route to the AS's border router, `10.0.5.150`.

To view the routing table on Router 2, you would do the following:

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

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

To add a second route to network `10.0.5.0`, which has its gateway as the border router, you would do the following:

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

Now the routing table has a route for the border router, which has the IP address `10.0.5.150/24`.

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

Configuring Multihomed Hosts

In Oracle Solaris, a system with more than one interface is considered a *multihomed host*. A multihomed host does not forward IP packets. However, you can configure a multihomed host to run routing protocols. You typically configure the following types of systems as multihomed hosts:

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

Note – When multihomed hosts have different types of firewalls on any of their interfaces, take care to avoid unintentional disruption of the host's packets. This problem arises particularly with stateful firewalls. One solution might be to configure stateless firewalling. For more information about firewalls, refer to "Firewall Systems" in *System Administration Guide: Security Services* or the documentation for your third-party firewall.

▼ How to Create a Multihomed Host

1 Configure each additional network interface that was not configured as part of the Oracle Solaris installation.

Refer to "How to Configure an IP Interface" in *System Administration Guide: Network Interfaces and Network Virtualization*.

2 **Verify that IPv4 forwarding is not enabled on the multihomed host.**

```
# ipadm show-prop -p forwarding ipv4
```

The following sample output shows that IPv4 forwarding is enabled:

```
PROTO PROPERTY     PERM CURRENT      PERSISTENT   DEFAULT      POSSIBLE
ipv4  forwarding   rw   on           --           off          on,off
```

3 **Turn off packet forwarding, if it is enabled on the system.**

```
# ipadm set-prop -p forwarding=off ipv4
```

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

Use either of the following commands to enable the `in.routed` daemon:

- For the `routeadm` command, type the following:

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

- To use SMF, type the following:

  ```
  # svcadm enable route:default
  ```

Example 4–4 Configuring a Multihomed Host

The following example shows how to configure the multihomed host that is shown in Figure 4–3. In the example, the system has the host name `hostc`. This host has two interfaces, which are both connected to network `192.168.5.0`.

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

```
# dladm show-link
LINK     CLASS     MTU     STATE   BRIDGE   OVER
bge0     phys      1500    up      --       --
bge1     phys      1500    up      --       --
bge2     phys      1500    up      --       --
e1000g0  phys      1500    up      --       --
e1000g1  phys      1500    up      --       --#

ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
e1000g0/v4        static   ok           192.168.5.82/24
```

The `dladm show-link` command reports that `hostc` has two interfaces with a total of five possible links. However, only `e1000g0` has been configured with an IP address. To configure `hostc` as a multihomed host, you must configure another link, such as `bge0`. First, you would physically connect the `bge0` interface to the `192.168.5.0` network. Then you would configure the `bge0` interface, and make the interface persist across reboots.

```
# ipadm create-addr -T static -a 192.168.5.85/24 bge0/v4
# ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
e1000g0/v4        static   ok           192.168.5.82/24
bge0/v4           static   ok           192.168.5.85/24
```

Next, you would add the `bge0` interface to the `/etc/hosts` database:

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

Then, you would check the state of packet forwarding and routing on `host3`:

```
# ipadm show-prop -p forwarding ipv4
PROTO PROPERTY     PERM CURRENT   PERSISTENT   DEFAULT      POSSIBLE
ipv4  forwarding   rw   on        --           off          on,off

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

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

The `routeadm` command reports that dynamic routing through the `in.routed` daemon and packet forwarding are currently enabled. However, you would need to disable packet forwarding:

```
# ipadm set-prop -p forwarding=off ipv4
```

When packet forwarding is disabled, `host3` becomes a multihomed host.

Configuring Routing for Single-Interface Systems

Single-interface hosts need to implement some form of routing. If the host is to obtain its routes from one or more local default routers, then you must configure the host to use static routing. Otherwise, dynamic routing is recommended for the host. The following procedures contain the instructions for enabling both routing types.

▼ How to Enable Static Routing on a Single-Interface Host

This procedure enables static routing on a single-interface host. Hosts that use static routing do not run a dynamic routing protocol such as RIP. Instead, the host must rely on the services of a default router for routing information. The figure "IPv4 Autonomous System Topology" on

page 88 shows several default routers and their client hosts. If you supplied the name of a default router when you installed a particular host, that host is already configured to use static routing.

Note – You can also use the following procedure to configure static routing on a multihomed host.

For information about the `/etc/defaultrouter` file, see "`/etc/defaultrouter` File" on page 178. For information about static routing and the routing table, refer to "Routing Tables and Routing Types" on page 94.

1 Verify whether the `/etc/defaultrouter` file is present on the host.

```
# cd /etc
# ls | grep defaultrouter
```

2 Open a text editor to create or modify the `/etc/defaultrouter` file

3 Add an entry for the default router.

```
# vi  /etc/defaultrouter
router-IP
```

where *router-IP* indicates the IP address of the default router for the host to use.

4 Verify that routing and packet forwarding are not running on the host.

```
# # ipadm show-prop -p forwarding ipv4
PROTO PROPERTY     PERM CURRENT   PERSISTENT   DEFAULT      POSSIBLE
ipv4  forwarding   rw   off        --           off          on,off

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

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

5 Add an entry for the default router in the local `/etc/inet/hosts` file.

For information about configuring `/etc/inet/hosts`, refer to "How to Change the IPv4 Address and Other Network Configuration Parameters" on page 83.

Example 4–5 Configuring a Default Router and Static Routing for a Single-Interface Host

The following example shows how to configure static routing for `hostb`, a single-interface host on the network `172.20.1.0` that is shown in Figure 4–3. `hostb` needs to use Router 2 as its default router.

First, you would log in to `hostb` as superuser, or assume an equivalent role. Then, you would determine whether the `/etc/defaultrouter` file is present on the host:

```
# cd /etc
# ls | grep defaultrouter
```

No response from `grep` indicates that you need to create the `/etc/defaultrouter` file.

```
# vi /etc/defaultrouter
172.20.1.10
```

The entry in the `/etc/defaultrouter` file is the IP address of the interface on Router 2, which is attached to the `172.20.1.0` network. Next, you verify whether the host currently enables packet forwarding or routing.

```
# ipadm show-prop -p forwarding ipv4
PROTO PROPERTY     PERM CURRENT   PERSISTENT   DEFAULT      POSSIBLE
ipv4  forwarding   rw   on          --           off          on,off

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

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

Packet forwarding is enabled for this particular host. You would turn it off as follows:

```
# ipadm set-prop -p forwarding=off ipv4
```

Lastly, you would make sure that the host's `/etc/inet/hosts` file has an entry for the new default router.

```
# vi /etc/inet/hosts
127.0.0.1           localhost
172.20.1.18         host2    #primary network interface for host2
172.20.1.10         router2  #default router for host2
```

How to Enable Dynamic Routing on a Single-Interface Host

Dynamic routing is the easiest way to manage routing on a host. Hosts that use dynamic routing run the routing protocols provided by the `in.routed` daemon for IPv4 or `in.ripngd` daemon for IPv6. Use the next procedure to enable IPv4 dynamic routing on a single interface host. For more information about dynamic routing, refer to "Packet Forwarding and Routing on IPv4 Networks" on page 84.

1. **Verify whether the `/etc/defaultrouter` file exists.**

```
# cd /etc
# ls | grep defaultrouter
```

2. **If `/etc/defaultrouter` exists, delete any entry that you find there.**

   An empty `/etc/defaultrouter` file forces the host to use dynamic routing.

3. **Verify whether packet forwarding and routing are enabled on the host.**

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

          Routing services   "route:default ripng:default"

# ipadm show-prop -p forwarding ipv4
PROTO PROPERTY     PERM CURRENT   PERSISTENT   DEFAULT      POSSIBLE
ipv4  forwarding   rw   on          --           off          on,off
```

4. **If packet forwarding is enabled, turn it off**

```
# ipadm set-prop -p forwarding=off ipv4
```

5. **Enable routing protocols on the host.**

   Use either of the following commands:

   - For the `routeadm` command, type the following:

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

   - To use SMF, type the following:

     ```
     # svcadm enable route:default
     ```

   Now IPv4 dynamic routing is enabled. The host's routing table is dynamically maintained by the `in.routed` daemon.

Example 4–6 Running Dynamic Routing on a Single-Interface Host

The following example shows how to configure dynamic routing for `hosta`, a single-interface host on the network `192.168.5.0` that is shown in Figure 4–3. `hosta` currently uses Router 1 as its default router. However, `hosta` now needs to run dynamic routing.

First, you would log in to `hosta` as superuser or assume an equivalent role. Then, you would determine whether the `/etc/defaultrouter` file is present on the host:

```
# cd /etc
# ls | grep defaultrouter
defaultrouter
```

The response from `grep` indicates that a `/etc/defaultrouter` file exists for `hosta`.

```
# vi /etc/defaultrouter
192.168.5.10
```

The file has the entry `192.168.5.10`, which is the IP address for Router 1. You would delete this entry to enable static routing. Next, you would need to verify whether packet forwarding and routing are already enabled for the host.

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

           Routing services   "route:default ripng:default"

# ipadm show-prop -p forwarding ipv4
PROTO PROPERTY     PERM CURRENT   PERSISTENT   DEFAULT      POSSIBLE
ipv4  forwarding   rw   off         --           off          on,off
```

Both routing and packet forwarding are turned off for `hosta`. Turn on routing to complete the configuration of dynamic routing for `hosta`, as follows:

```
# svcadm enable route:default
```

Monitoring and Modifying Transport Layer Services

The transport layer protocols TCP, SCTP, and UDP are part of the standard Oracle Solaris package. These protocols typically need no intervention to run properly. However, circumstances at your site might require you to log or modify services that run over the transport layer protocols. Then, you must modify the profiles for these services by using the Service Management Facility (SMF), which is described in Chapter 11, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*.

The `inetd` daemon is responsible for starting standard Internet services when a system boots. These services include applications that use TCP, SCTP, or UDP as their transport layer protocol. You can modify existing Internet services or add new services using the SMF commands. For more information about `inetd`, refer to "`inetd` Internet Services Daemon" on page 184.

Operations that involve the transport layer protocols include:

- Logging of all incoming TCP connections
- Adding services that run over a transport layer protocol, using SCTP as an example
- Configuring the TCP wrappers facility for access control

For detailed information on the `inetd` daemon refer to the `inetd(1M)`man page.

How to Log the IP Addresses of All Incoming TCP Connections

- **Set TCP tracing to enabled for all services managed by `inetd`.**

```
# inetadm -M tcp_trace=TRUE
```

How to Add Services That Use the SCTP Protocol

The SCTP transport protocol provides services to application layer protocols in a fashion similar to TCP. However, SCTP enables communication between two systems, either or both of which can be multihomed. The SCTP connection is called an *association*. In an association, an application divides the data to be transmitted into one or more message streams, or *multi-streamed*. An SCTP connection can go to endpoints with multiple IP addresses, which is particularly important for telephony applications. The multihoming capabilities of SCTP are a security consideration if your site uses IP Filter or IPsec. Some of these considerations are described in the `sctp(7P)` man page.

By default, SCTP is included in the Oracle Solaris and does not require additional configuration. However, you might need to explicitly configure certain application layer services to use SCTP. Some example applications are `echo` and `discard`. The next procedure shows how to add an echo service that uses an SCTP one-to-one style socket.

Note – You can also use the following procedure to add services for the TCP and UDP transport layer protocols.

The following task shows how to add an SCTP `inet` service that is managed by the `inetd` daemon to the SMF repository. The task then shows how to use the Service Management Facility (SMF) commands to add the service.

- For information about SMF commands, refer to "SMF Command-Line Administrative Utilities" in *System Administration Guide: Basic Administration*.
- For syntactical information, refer to the man pages for the SMF commands, as cited in the procedure.
- For detailed information about SMF refer to the `smf(5)` man page.

Before You Begin Before you perform the following procedure, create a manifest file for the service. The procedure uses as an example a manifest for the `echo` service that is called `echo.sctp.xml`.

1 Log in to the local system with a user account that has write privileges for system files.

2 Edit the `/etc/services` file and add a definition for the new service.

Use the following syntax for the service definition.

service-name |*port/protocol* | *aliases*

3 Add the new service.

Go to the directory where the service manifest is stored and type the following:

```
# cd dir-name
# svccfg import service-manifest-name
```

For a complete syntax of `svccfg`, refer to the `svccfg(1M)` man page.

Suppose you want to add a new SCTP `echo` service using the manifest `echo.sctp.xml` that is currently located in the `service.dir` directory. You would type the following:

```
# cd service.dir
# svccfg import echo.sctp.xml
```

4 Verify that the service manifest has been added:

```
# svcs FMRI
```

For the *FMRI* argument, use the Fault Managed Resource Identifier (FMRI) of the service manifest. For example, for the SCTP `echo` service, you would use the following command:

```
# svcs svc:/network/echo:sctp_stream
```

Your output should resemble the following:

```
    STATE          STIME    FMRI
disabled       16:17:00 svc:/network/echo:sctp_stream
```

For detailed information about the `svcs` command, refer to the `svcs(1)` man page.

The output indicates that the new service manifest is currently disabled.

5 List the properties of the service to determine if you must make modifications.

```
# inetadm -l FMRI
```

For detailed information about the `inetadm` command, refer to the `inetadm(1M)` man page.

For example, for the SCTP `echo` service, you would type the following:

```
# inetadm -l svc:/network/echo:sctp_stream
SCOPE    NAME=VALUE
             name="echo"
```

```
            endpoint_type="stream"
            proto="sctp"
            isrpc=FALSE
            wait=FALSE
            exec="/usr/lib/inet/in.echod -s"
        .
        .
        default  tcp_trace=FALSE
          default  tcp_wrappers=FALSE
```

6 Enable the new service:

```
# inetadm -e FMRI
```

7 Verify that the service is enabled:

For example, for the new `echo` service, you would type the following:

```
# inetadm | grep sctp_stream
.
.
    enabled   online         svc:/network/echo:sctp_stream
```

Example 4–7 Adding a Service That Uses the SCTP Transport Protocol

The following example shows the commands to use and the file entries required to have the echo service use the SCTP transport layer protocol.

```
$ cat /etc/services
.
.
echo            7/tcp
echo            7/udp
echo            7/sctp

# cd service.dir

    # svccfg import echo.sctp.xml

# svcs network/echo*
    STATE          STIME    FMRI
    disabled       15:46:44 svc:/network/echo:dgram
    disabled       15:46:44 svc:/network/echo:stream
    disabled       16:17:00 svc:/network/echo:sctp_stream

# inetadm -l svc:/network/echo:sctp_stream
    SCOPE    NAME=VALUE
             name="echo"
             endpoint_type="stream"
             proto="sctp"
             isrpc=FALSE
             wait=FALSE
             exec="/usr/lib/inet/in.echod -s"
             user="root"
    default  bind_addr=""
    default  bind_fail_max=-1
```

```
    default  bind_fail_interval=-1
    default  max_con_rate=-1
    default  max_copies=-1
    default  con_rate_offline=-1
    default  failrate_cnt=40
    default  failrate_interval=60
    default  inherit_env=TRUE
    default  tcp_trace=FALSE
    default  tcp_wrappers=FALSE

# inetadm -e svc:/network/echo:sctp_stream

# inetadm | grep echo
    disabled  disabled       svc:/network/echo:stream
    disabled  disabled       svc:/network/echo:dgram
    enabled   online         svc:/network/echo:sctp_stream
```

How to Use TCP Wrappers to Control Access to TCP Services

The `tcpd` program implements *TCP wrappers*. TCP wrappers add a measure of security for service daemons such as `ftpd` by standing between the daemon and incoming service requests. TCP wrappers log successful and unsuccessful connection attempts. Additionally, TCP wrappers can provide access control, allowing or denying the connection depending on where the request originates. You can use TCP wrappers to protect daemons such as SSH, Telnet, and FTP. The `sendmail` application can also use TCP wrappers, as described in "Support for TCP Wrappers From Version 8.12 of sendmail" in *System Administration Guide: Network Services*.

1. **Set TCP wrappers to enabled.**

   ```
   # inetadm -M tcp_wrappers=TRUE
   ```

2. **Configure the TCP wrappers access control policy as described in the `hosts_access`(3) man page.**

   This man page can be found in the `/usr/sfw/man` directory.

◆ ◆ ◆ CHAPTER 5

Enabling IPv6 on a Network (Tasks)

This chapter contains tasks for configuring IPv6 on a network. The following major topics are covered:

For an overview of IPv6 concepts, refer to Chapter 2, "Planning an IPv6 Addressing Scheme (Overview)." For IPv6 planning tasks, refer to Chapter 3, "Planning an IPv6 Network (Tasks)." For information about preparing your network to use IP tunnels, refer to "Planning for Tunnels in the Network Topology" on page 69. To find reference information about the tasks in this chapter, refer to Chapter 10, "IPv6 in Depth (Reference)."

Configuring an IPv6 Interface

The initial step in IPv6 configuration is enabling IPv6 on an interface. You can enable IPv6 support during the Oracle Solaris installation process or by configuring IPv6 on the interfaces of an installed system.

During the Oracle Solaris installation process, you can enable IPv6 on one or more of a system's interfaces. After installation, the following IPv6-related files and tables are in place:

- The `/etc/nsswitch.conf` file has been modified to accommodate lookups using IPv6 addresses.
- The IPv6 address selection policy table is created. This table prioritizes the IP address format to use for transmissions over an IPv6-enabled interface.

This section describes how to enable IPv6 on the interfaces of an installed system.

Enabling IPv6 on an Interface (Task Map)

The following table lists different tasks for configuring the IPv6 interfaces. The table includes a description of what each task accomplishes and the section in the current documentation where the specific steps to perform the task are detailed.

Task	Description	For Instructions
Enable IPv6 on an interface on a system that has already been installed with Oracle Solaris.	Use this task for enabling IPv6 on an interface after Oracle Solaris has been installed.	"How to Enable an IPv6 Interface for the Current Session" on page 110
Turn off IPv6 address autoconfiguration.	Use this task if you need to manually configure the interface ID portion of the IPv6 address.	"How to Turn Off IPv6 Address Autoconfiguration" on page 112

▼ How to Enable an IPv6 Interface for the Current Session

Begin your IPv6 configuration process by enabling IPv6 on the interfaces of all systems that will become IPv6 nodes. Initially, the interface obtains its IPv6 address through the autoconfiguration process, as described in "IPv6 Address Autoconfiguration" on page 61. You then can tailor the node's configuration based on its function in the IPv6 network, either as a host, server, or router.

Note – If the interface is on the same link as a router that currently advertises an IPv6 prefix, the interface obtains that site prefix as part of its autoconfigured addresses. For more information, refer to "How to Configure an IPv6-Enabled Router" on page 113.

The following procedure explains how to enable IPv6 for an interface that was added after an Oracle Solaris installation.

Before You Begin Complete the planning tasks for the IPv6 network, such as upgrading hardware and software, and preparing an addressing plan. For more information, see "IPv6 Planning (Task Maps)" on page 63.

1 Enable IPv6 on an interface.

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

where *addrobj* is the IP address identifier that uses the naming convention *interface*/*user-defined-string*, such as `bge0/v6addr`.

Use the same command syntax to add more IPv6 addresses. Make sure that for each additional IPv6 address, you use a different *addrobj* value. For example:

```
# ipadm create-addr -T static ipv6-address  addrobj
```

2 **Start the IPv6 daemon `in.ndpd`.**

```
# /usr/lib/inet/in.ndpd
```

Note – You can display the status of a node's IPv6-enabled interfaces by using the `ipadm show-addr` command.

---

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

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

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

If you need to create temporary addresses for the host's interface, refer to "Using Temporary Addresses for an Interface" on page 117. For details about `/etc/inet/ndpd.conf`, refer to the `ndpd.conf(4)` man page and "`ndpd.conf` Configuration File" on page 203.

Example 5–1 Enabling an IPv6 Interface After Installation

This example shows how to enable IPv6 on the `bge0` interface. Before you begin, check the status of all interfaces configured on the system.

```
# ipadm show-addr
ADDROBJ   TYPE     STATE   ADDR
lo0/v4    static   ok      127.0.0.1/8
bge0/v4   static   ok      172.16.27.74/24
```

Only the `bge0` interface is currently configured for this system. Enable IPv6 on this interface as follows:

```
# ipadm create-addr -T addrconf bge0/v6
# ipadm create-addr -T static -a 2001:db8:3c4d:15:203/64 bge0/v6add
# /usr/lib/inet/in.ndpd
# ipadm show-addr
ADDROBJ       TYPE       STATE   ADDR
lo0/v4        static     ok      127.0.0.1/8
bge0/v4       static     ok      172.16.27.74/24
bge0/v6       addrconf   ok      fe80::203:baff:fe13:14e1/10
lo0/v6        static     ok      ::1/128
bge0/v6add    static     ok      2001:db8:3c4d:15:203/64

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

The example shows the status of the system's interface before and after `bge0`becomes IPv6-enabled. Note that the output indicates that only a link-local address was configured for `bge0`, `fe80::203:baff:fe13:14e1/10`. This address indicates that as of yet no router on the node's local link advertises a site prefix.

After IPv6 is enabled, you can use the `ipadm show-addr` command to display both IPv4 and IPv6 addresses for all interfaces on a system.

Next Steps

- To configure the IPv6 node as a router, go to "Configuring an IPv6 Router" on page 112.
- To disable address autoconfiguration on the node, see "How to Turn Off IPv6 Address Autoconfiguration" on page 112.
- To tailor the node as a server, see the suggestions in "Administering IPv6-Enabled Interfaces on Servers" on page 121.

▼ How to Turn Off IPv6 Address Autoconfiguration

You normally should use address autoconfiguration to generate the IPv6 addresses for the interfaces of hosts and servers. However, sometimes you might want to turn off address autoconfiguration, especially if you want to manually configure a token, as explained in "Configuring an IPv6 Token" on page 119.

1 Create an `/etc/inet/ndpd.conf` file for the node.

The `/etc/inet/ndpd.conf` file defines interface variables for the particular node. This file should have the following contents in order to turn off address autoconfiguration for all of the server's interfaces:

```
if-variable-name StatelessAddrConf false
```

For details about `/etc/inet/ndpd.conf`, refer to the `ndpd.conf(4)` man page and "`ndpd.conf` Configuration File" on page 203.

2 Update the IPv6 daemon with your changes.

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

Configuring an IPv6 Router

The first step in configuring IPv6 on a network is configuring IPv6 on a router. Router configuration involves a number of discrete tasks, which are described in this section. You might perform some or all of the tasks, depending on your site requirements.

IPv6 Router Configuration (Task Map)

Perform the next tasks in the following table in order that is shown to configure the IPv6 network. The table includes a description of what each task accomplishes and the section in the current documentation where the specific steps to perform the task are detailed.

Task	Description	For Instructions
1. Ensure that you have completed the required prerequisites before you begin IPv6 configuration.	You must complete planning tasks and Oracle Solaris installation with IPv6 enabled interfaces before you configure an IPv6-enabled router.	Chapter 3, "Planning an IPv6 Network (Tasks)," and "Configuring an IPv6 Interface" on page 109.
2. Configure a router.	Define the site prefix for the network.	"How to Configure an IPv6-Enabled Router" on page 113
3. Configure tunnel interfaces on the router.	Set up a manual tunnel or a 6to4 tunnel interface on the router. The local IPv6 network needs tunnels to communicate with other, isolated IPv6 networks.	■ "Tunnel Configuration and Administration With the `dladm` Command" on page 159
4. Configure the switches on the network.	If your network configuration includes switches, configure them for IPv6 at this point in the configuration process.	Refer to switch manufacturer's documentation.
5. Configure any hubs on your network.	If your network configuration includes hubs, configure them for IPv6 at this point in the configuration process.	Refer to hub manufacturer's documentation.
6. Configure the network name service for IPv6.	Configure your primary name service (DNS, NIS, or LDAP) to recognize IPv6 addresses after the router is configured for IPv6.	"How to Add IPv6 Addresses to DNS" on page 123
7. (Optional) Modify the addresses for the IPv6-enabled interfaces on hosts and servers.	After IPv6 router configuration, make further modifications on IPv6-enabled hosts and servers.	"Modifying an IPv6 Interface Configuration for Hosts and Servers" on page 116
Configure applications to support IPv6	Different applications might require different actions in order to support IPv6.	Refer to applications' documentation

▼ How to Configure an IPv6-Enabled Router

This procedure assumes that all interfaces of the router were configured for IPv6 during Oracle Solaris installation.

1 **Review which interfaces on the router were configured for IPv6 during installation.**

```
# ipadm show-addr
```

Check the output to ensure that the interfaces that you wanted to configure for IPv6 are now plumbed with link-local addresses. The following sample command output of `ipadm show-addr` shows the IPv4 and IPv6 addresses that were configured for the router's interfaces.

```
ADDROBJ   TYPE       STATE   ADDR
lo0/v4    static     ok      127.0.0.1/8
bge0/v4   static     ok      172.16.27.74/24
bge1/v4   static     ok      172.16.27.84/24
bge0/v6   addrconf   ok      fe80::203:baff:fe11:b115/10
lo0/v6    static     ok      ::1/128
bge1/v6   addrconf   ok      fe80::203:baff:fe11:b116/10
```

The output also shows an IPv6 address object each for the `bge0` and `bge1` network interfaces, namely, `fe80::203:baff:fe11:b115/10` and `fe80::203:baff:fe11:b116/10`.

2 **Configure IPv6 packet forwarding on all interfaces of the router.**

```
# ipadm set-prop -p forwarding ipv6
```

3 **Start the routing daemon.**

The `in.ripngd` daemon handles IPv6 routing.

Turn on IPv6 routing in either of the following ways:

- Use the `routeadm` command as follows:

  ```
  # routeadm -e ipv6-routing -u
  ```

- Use SMF to enable IPv6 routing:

  ```
  # svcadm enable ripng:default
  ```

For syntax information on the `routeadm` command, see the `routeadm(1M)` man page.

4 **Create the `/etc/inet/ndpd.conf` file.**

You specify the site prefix to be advertised by the router and other configuration information in `/etc/inet/ndpd.conf`. This file is read by the `in.ndpd` daemon, which implements the IPv6 Neighbor Discovery protocol.

For a list of variables and allowable values, refer to "`ndpd.conf` Configuration File" on page 203 and the `ndpd.conf(4)`man page.

5 **Type the following text into the `/etc/inet/ndpd.conf` file:**

```
ifdefault AdvSendAdvertisements true
prefixdefault AdvOnLinkFlag on AdvAutonomousFlag on
```

This text tells the `in.ndpd` daemon to send out router advertisements over all interfaces of the router that are configured for IPv6.

6 Add additional text to the `/etc/inet/ndpd.conf` file to configure the site prefix on the various interfaces of the router.

The text should have the following format:

`prefix` *global-routing-prefix*`:`*subnet ID*`/64` *interface*

The following sample `/etc/inet/ndpd.conf` file configures the router to advertise the site prefix `2001:0db8:3c4d::/48` over the interfaces `bge0` and `bge1`.

```
ifdefault AdvSendAdvertisements true
prefixdefault AdvOnLinkFlag on AdvAutonomousFlag on

if bge0 AdvSendAdvertisements 1
prefix 2001:0db8:3c4d:15::0/64 bge0

if bge1 AdvSendAdvertisements 1
prefix 2001:0db8:3c4d:16::0/64 bge1
```

7 Reboot the system.

The IPv6 router begins advertising on the local link any site prefix that is in the `ndpd.conf` file.

Example 5–2 `ipadm show-addr` Output Showing IPv6 Interfaces

The following example shows output from the `ipadm show-addr` command such as you would receive after you finish the "Configuring an IPv6 Router" on page 112 procedure.

```
ADDROBJ      TYPE       STATE   ADDR
lo0/v4       static     ok      127.0.0.1/8
bge0/v4      static     ok      172.16.15.232/24
bge1/v4      static     ok      172.16.16.220/24
bge0/v6      addrconf   ok      fe80::203:baff:fe11:b115/10
lo0/v6       static     ok      ::1/128
bge0/v6add   static     ok      2001:db8:3c4d:15:203:baff:fe11:b115/64
bge1/v6      addrconf   ok      fe80::203:baff:fe11:b116/10
bge1/v6add   static     ok      2001:db8:3c4d:16:203:baff:fe11:b116/64
```

In this example, each interface that was configured for IPv6 now has two addresses. The entry with the address object name such as *interface*`/v6` shows the link-local address for that interface. The entry with the address object name such as *interface*`/v6add` shows a global IPv6 address. This address includes the site prefix that you configured in the `/etc/ndpd.conf` file, in addition to the interface ID. Note that the designation `v6add` is a randomly defined string. You can define other strings to constitute the second part of the address object name, provided that the *interface* reflects the interface over which you are creating the IPv6 addresses, for example `bge0/mystring`, `bge0/ipv6addr`, and so on.

See Also

- To configure any tunnels from the routers that you have identified in your IPv6 network topology, refer to "Tunnel Configuration and Administration With the `dladm` Command" on page 159.

- For information about configuring switches and hubs on your network, refer to the manufacturer's documentation.
- To configure IPv6 hosts, refer to "Modifying an IPv6 Interface Configuration for Hosts and Servers" on page 116.
- To improve IPv6 support on servers, refer to "Administering IPv6-Enabled Interfaces on Servers" on page 121.
- For detailed information about IPv6 commands, files, and daemons, refer to "Oracle Solaris IPv6 Implementation" on page 203.

Modifying an IPv6 Interface Configuration for Hosts and Servers

This section explains how to modify the configuration of IPv6-enabled interfaces on nodes that are hosts or servers. In most instances, you should use address autoconfiguration for IPv6-enabled interfaces, as explained in "Stateless Autoconfiguration Overview" on page 61. However, you can modify the IPv6 address of an interface, if necessary, as explained in the tasks of this section.

Modifying an IPv6 Interface Configuration (Task Map)

The following table lists different tasks to modify an existing IPv6 network. The table includes a description of what each task accomplishes and the section in the current documentation where the specific steps to perform the task are detailed.

Task	Description	For Instructions
Turn off IPv6 address autoconfiguration.	Use this task if you need to manually configure the interface ID portion of the IPv6 address.	"How to Turn Off IPv6 Address Autoconfiguration" on page 112
Create a temporary address for a host.	Hide a host's interface ID by configuring a randomly created temporary address that is used as the lower 64 bits of the address.	"How to Configure a Temporary Address" on page 117
Configure a token for the interface ID of a system.	Create a 64-bit token to be used as the interface ID in an IPv6 address.	"How to Configure a User-Specified IPv6 Token" on page 120

Using Temporary Addresses for an Interface

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

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

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

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

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

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

How to Configure a Temporary Address

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

Refer to "How to Enable an IPv6 Interface for the Current Session" on page 110.

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

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

```
ifdefault TmpAddrsEnabled true
```

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

  ```
  if interface TmpAddrsEnabled true
  ```

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

```
ifdefault TmpValidLifetime duration
```

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

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

```
if interface TmpPreferredLifetime duration
```

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

Note – Default address selection gives a lower priority to IPv6 addresses that have been deprecated. If an IPv6 temporary address is deprecated, default address selection chooses a nondeprecated address as the source address of a packet. A nondeprecated address could be the automatically generated IPv6 address, or possibly, the interface's IPv4 address. For more information about default address selection, see "Administering Default Address Selection" on page 147.

---

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

```
ifdefault TmpRegenAdvance duration
```

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

6 Change the configuration of the `in.ndpd` daemon.

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

7 Verify that temporary addresses have been created by issuing the `ipadm show-addr` command, as shown in Example 5–4.

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

Example 5–3 Temporary Address Variables in the `/etc/inet/ndpd.conf` File

The following example shows a segment of an `/etc/inet/ndpd.conf` file with temporary addresses enabled for the primary network interface.

```
ifdefault TmpAddrsEnabled true

ifdefault TmpValidLifetime 14d

ifdefault TmpPreferredLifetime 7d

ifdefault TmpRegenAdvance 6s
```

Example 5–4 `ipadm show-addr` Command Output with Temporary Addresses Enabled

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

```
# ipadm show-addr -o all
ADDROBJ   TYPE     STATE CURRENT PERSISTENT ADDR
lo0/v6    static   ok    U----   ---        ::1/128
bge0/v6   addrconf ok    U----   ---        fe80::a00:20ff:feb9:4c54/10
bge0/v6a  static   ok    U----   ---        2001:db8:3c4d:15:a00:20ff:feb9:4c54/64
bge0/?    addrconf ok    U--t-   ---        2001:db8:3c4d:15:7c37:e7d1:fc9c:d2cb/64
```

Note that for the address object `bge0/?`, the `t` flag is set under the `CURRENT` field. The flag indicates that the corresponding address has a temporary interface ID.

See Also

- To set up name service support for IPv6 addresses, see "Configuring Name Service Support for IPv6" on page 122.
- To configure IPv6 addresses for a server, see "How to Configure a User-Specified IPv6 Token" on page 120.
- To monitor activities on IPv6 nodes, see Chapter 6, "Administering a TCP/IP Network (Tasks)."

Configuring an IPv6 Token

The 64-bit interface ID of an IPv6 address is also referred to as a *token*, as introduced in "IPv6 Addressing Overview" on page 54. During address autoconfiguration, the token is associated with the interface's MAC address. In most cases, nonrouting nodes, that is IPv6 hosts and servers, should use their autoconfigured tokens.

However, using autoconfigured tokens can be a problem for servers whose interfaces are routinely swapped as part of system maintenance. When the interface card is changed, the MAC address is also changed. Servers that depend on having stable IP addresses can experience

problems as a result. Various parts of the network infrastructure, such as DNS or NIS, might have stored specific IPv6 addresses for the interfaces of the server.

To avoid address change problems, you can manually configure a token to be used as the interface ID in an IPv6 address. To create the token, you specify a hexadecimal number of 64 bits or less to occupy the interface ID portion of the IPv6 address. During subsequent address autoconfiguration, Neighbor Discovery does not create an interface ID that is based on the interface's MAC address. Instead, the manually created token becomes the interface ID. This token remains assigned to the interface, even when a card is replaced.

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

How to Configure a User-Specified IPv6 Token

The next instructions are particularly useful for servers whose interfaces are routinely replaced. They also are valid for configuring user-specified tokens on any IPv6 node.

1 Verify that the interface you want to configure with a token exists.

Note – Ensure that the interface has no configured IPv6 address.

```
# ipadm show-if
IFNAME     STATE    CURRENT      PERSISTENT
lo0        ok       -m-v------4- ---
bge0       ok       bm--------4- ---
# ipadm show-addr
ADDROBJ      TYPE       STATE   ADDR
lo0/v4       static     ok      127.0.0.1/8
```

This output shows that the network interface `bge0` exists with no configured IPv6 address.

2 Create one or more 64-bit hexadecimal numbers to be used as tokens for the node's interfaces. For examples of tokens, refer to "Link-Local Unicast Address" on page 58.

3 Configure each interface with a token.

Use the following form of the `ipadm` command for each interface to have a user-specified interface ID (token):

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

For example, you would use the following command to configure interface `bge0` with a token:

```
# ipadm create-addr -T addrconf -i ::1a:2b:3c:4d/64 bge0/v6add
```

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

Repeat this step for every interface that will have a user-specified token.

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

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

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

The following example shows `bge0` being configured with an IPv6 address and a token.

```
# ipadm show-if
IFNAME     STATE    CURRENT      PERSISTENT
lo0        ok       -m-v------4- ---
bge0       ok       bm--------4- ---

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

# ipadm create-addr -T addrconf -i ::1a:2b:3c:4d/64 bge0/v6
# pkill -HUP -in.ndpd
# ipadm show-addr
ADDROBJ      TYPE       STATE   ADDR
lo0/v6       static     ok      ::1/128
bge0/v6      addrconf   ok      fe80::1a:2b:3c:4d/10
bge0/v6      addrconf   ok      2002:a08:39f0:1:1a:2b:3c:4d/64
```

After the token is configured, the address object `bge0/v6` has both a link local address as well as an address with `1a:2b:3c:4d`configured for its interface ID. Note that this token can no longer be modified for this interface after `bge0/v6` was created.

See Also

- To update the name services with the IPv6 addresses of the server, see "Configuring Name Service Support for IPv6" on page 122.
- To monitor server performance, see Chapter 6, "Administering a TCP/IP Network (Tasks)."

Administering IPv6-Enabled Interfaces on Servers

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

▼ How to Enable IPv6 on a Server's Interfaces

Before You Begin The next procedure assumes the following:

- Oracle Solaris is already installed on the server.
- You enabled IPv6 on the server's interfaces either during Oracle Solaris installation or later, using the procedures in "Configuring an IPv6 Interface" on page 109.

If applicable, upgrade the application software to support IPv6. Note that many applications that run on the IPv4 protocol stack also successfully run on IPv6. For more information, refer to "How to Prepare Network Services for IPv6 Support" on page 68.

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

   For more information, refer to "Configuring an IPv6 Router" on page 112.

2. **Use the appropriate strategy for the interface ID for the server's IPv6-enabled interfaces.**

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

   - For an IPv6-enabled interface that you do not plan to replace, use the autoconfigured IPv6 address, as introduced in "IPv6 Address Autoconfiguration" on page 61.
   - For IPv6-enabled interfaces that must appear anonymous outside the local network, consider using a randomly generated token for the interface ID. For instructions and an example, refer to "How to Configure a Temporary Address" on page 117.
   - For IPv6-enabled interfaces that you plan to swap on a regular basis, create tokens for the interface IDs. For instructions and an example, refer to "How to Configure a User-Specified IPv6 Token" on page 120.

Configuring Name Service Support for IPv6

This section describes how to configure the DNS and NIS name services to support IPv6 services.

Note – LDAP supports IPv6 without requiring IPv6-specific configuration tasks.

For full details for administering DNS, NIS, and LDAP, refer to the *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

How to Add IPv6 Addresses to DNS

1. **Edit the appropriate DNS zone file by adding AAAA records for each IPv6-enabled node:**

   host-name `IN AAAA` *host-address*

2. **Edit the DNS reverse zone file and add PTR records:**

   host-address `IN PTR` *hostname*

   For detailed information on DNS administration, refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

Example 5–6 DNS Reverse Zone File

This example shows an IPv6 address in the reverse zone file.

```
$ORIGIN    ip6.int.
8.2.5.0.2.1.e.f.f.f.9.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.2.0.0.0 \
    IN        PTR        vallejo.Eng.apex.COM.
```

How to Display IPv6 Name Service Information

You can use the `nslookup` command to display IPv6 name service information.

1. **Under your user account, run the `nslookup` command.**

   `% /usr/sbin/nslookup`

   The default server name and address appear, followed by the `nslookup` command's angle bracket prompt.

2. **View information about a particular host by typing the following commands at the angle bracket prompt:**

   `>set q=any`
   >host-name

3. **Type the following command to view only AAAA records:**

   `>set q=AAAA`
   hostname

4. **Quit the `nslookup` command by typing `exit`.**

Example 5–7 Using `nslookup` to Display IPv6 Information

This example shows the results of `nslookup` in an IPv6 network environment.

```
%  /usr/sbin/nslookup
Default Server:  dnsserve.local.com
Address:  10.10.50.85
> set q=AAAA
> host85
Server:  dnsserve.local.com
Address:  10.10.50.85

host85.local.com      IPv6 address = 2::9256:a00:fe12:528
> exit
```

How to Verify That DNS IPv6 PTR Records Are Updated Correctly

In this procedure, you use the `nslookup` command to display PTR records for DNS IPv6.

1. **Under your user account, run the `nslookup` command.**

   ```
   % /usr/sbin/nslookup
   ```

   The default server name and address display, followed by the `nslookup` command's angle bracket prompt.

2. **Type the following at the angle bracket prompt to see the PTR records:**

   ```
   >set q=PTR
   ```

3. **Quit the command by typing `exit`.**

Example 5–8 Using `nslookup` to Display PTR Records

The following example shows the PTR record display from the `nslookup` command.

```
%  /usr/sbin/nslookup
Default Server:  space1999.Eng.apex.COM
Address:  192.168.15.78
> set q=PTR
> 8.2.5.0.2.1.e.f.f.f.0.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.2.0.0.0.ip6.int

8.2.5.0.2.1.e.f.f.f.0.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.2.0.0.0.ip6.int name =
vallejo.ipv6.Eng.apex.COM
ip6.int nameserver = space1999.Eng.apex.COM
> exit
```

How to Display IPv6 Information Through NIS

In this procedure, you use the `ypmatch` command to display IPv6 information through NIS:

- **Under your user account, type the following to display IPv6 addresses in NIS:**

  `% ypmatch` *hostname* `hosts .`*byname*

  The information about the specified *hostname* is displayed.

◆ ◆ ◆ CHAPTER 6

Administering a TCP/IP Network (Tasks)

This chapter contains tasks for administering a TCP/IP network. The following topics are covered:

Note – To monitor network interfaces, see "Monitoring IP Interfaces and Addresses" in *System Administration Guide: Network Interfaces and Network Virtualization*.

The tasks assume that you have an operational TCP/IP network at your site, either IPv4-only or dual-stack IPv4/IPv6. If you want to implement IPv6 at your site but have not done so, refer to following chapters for more information:

- To plan an IPv6 implementation, refer to Chapter 3, "Planning an IPv6 Network (Tasks)."
- To configure IPv6 and create a dual-stack network environment, refer to Chapter 5, "Enabling IPv6 on a Network (Tasks)."

Major TCP/IP Administrative Tasks (Task Map)

The following table lists other miscellaneous tasks to administer the network after initial configuration, such as displaying network information. The table includes a description of what each task accomplishes and the section in the current documentation where the specific steps to perform the task are detailed.

Task	Description	For Information
Display statistics on a per-protocol basis.	Monitor the performance of the network protocols on a particular system.	"How to Display Statistics by Protocol" on page 129
Display network status.	Monitor your system by displaying all sockets and routing table entries. The output includes the inet address family for IPv4 and inet6 address family for IPv6.	"How to Display the Status of Sockets" on page 132
Display the status of network interfaces.	Monitor the performance of network interfaces, which is useful for troubleshooting transmission problems.	"How to Display Network Interface Status" on page 131
Display packet transmission status.	Monitor the state of packets as they are sent over the wire.	"How to Display the Status of Transmissions for Packets of a Specific Address Type" on page 134
Control the display output of IPv6-related commands.	Controls the output of the `ping`, `netstat`, and `traceroute` commands. Creates a file that is named `inet_type`. Sets the `DEFAULT_IP` variable in this file.	"How to Control the Display Output of IP-Related Commands" on page 137
Monitor network traffic.	Displays all IP packets by using the `snoop` command.	"How to Monitor IPv6 Network Traffic" on page 143
Trace all routes that are known to the network's routers.	Uses the `traceroute` command to show all routes.	"How to Trace All Routes" on page 140

Note – To monitor network interfaces, refer to "Monitoring IP Interfaces and Addresses" in *System Administration Guide: Network Interfaces and Network Virtualization*

Monitoring Network Status With the netstat Command

The netstat command generates displays that show network status and protocol statistics. You can display the status of TCP, SCTP, and UDP endpoints in table format. You can also display routing table information and interface information.

The netstat command displays various types of network data, depending on the selected command-line option. These displays are the most useful for system administration. The basic syntax for netstat follows:

netstat [-m] [-n] [-s] [-i | -r] [-f*address-family*]

This section describes the most commonly used options of the netstat command. For a detailed description of all netstat options, refer to the netstat(1M) man page.

▼ How to Display Statistics by Protocol

The netstat -s option displays protocol statistics for the UDP, TCP, SCTP, ICMP, and IP protocols.

Note – You can use your Oracle Solaris user account to obtain output from the netstat command.

- **Display the protocol status.**

```
$ netstat -s
```

Example 6–1 Network Protocol Statistics

The following example shows the output of the netstat -s command. Parts of the output have been truncated. The output can indicate areas where a protocol is having problems. For example, statistical information from ICMPv4 and ICMPv6 can indicate where the ICMP protocol has found errors.

```
RAWIP
        rawipInDatagrams    =  4701     rawipInErrors       =     0
        rawipInCksumErrs    =     0     rawipOutDatagrams   =     4
        rawipOutErrors      =     0

UDP
        udpInDatagrams      = 10091     udpInErrors         =     0
        udpOutDatagrams     = 15772     udpOutErrors        =     0

TCP     tcpRtoAlgorithm     =     4     tcpRtoMin           =   400
        tcpRtoMax           = 60000     tcpMaxConn          =    -1
        .
        .
```

```
        tcpListenDrop       =     0     tcpListenDropQ0     =     0
        tcpHalfOpenDrop     =     0     tcpOutSackRetrans   =     0

IPv4    ipForwarding        =     2     ipDefaultTTL        =   255
        ipInReceives        =300182     ipInHdrErrors       =     0
        ipInAddrErrors      =     0     ipInCksumErrs       =     0
        .
        .
        ipsecInFailed       =     0     ipInIPv6            =     0
        ipOutIPv6           =     3     ipOutSwitchIPv6     =     0

IPv6    ipv6Forwarding      =     2     ipv6DefaultHopLimit =   255
        ipv6InReceives      = 13986     ipv6InHdrErrors     =     0
        ipv6InTooBigErrors  =     0     ipv6InNoRoutes      =     0
        .
        .
        rawipInOverflows    =     0     ipv6InIPv4          =     0

       ipv6OutIPv4         =     0     ipv6OutSwitchIPv4   =     0

ICMPv4  icmpInMsgs          = 43593     icmpInErrors        =     0
        icmpInCksumErrs     =     0     icmpInUnknowns      =     0
        .
        .
        icmpInOverflows     =     0

ICMPv6  icmp6InMsgs         = 13612     icmp6InErrors       =     0
        icmp6InDestUnreachs =     0     icmp6InAdminProhibs =     0
        .
        .
        icmp6OutGroupQueries=     0     icmp6OutGroupResps  =     2
        icmp6OutGroupReds   =     0

IGMP:
      12287 messages received
          0 messages received with too few bytes
          0 messages received with bad checksum
      12287 membership queries received
SCTP  sctpRtoAlgorithm     =  vanj
      sctpRtoMin           =  1000
      sctpRtoMax           = 60000
      sctpRtoInitial       =  3000
      sctpTimHearBeatProbe =     2
      sctpTimHearBeatDrop  =     0
      sctpListenDrop       =     0
      sctpInClosed         =     0
```

▼ How to Display the Status of Transport Protocols

You can display the status of the transport protocols through the `netstat` command. For detailed information, refer to the `netstat(1M)` man page.

1 Display the status of the TCP and SCTP transport protocols on a system.

```
$ netstat
```

2. **Display the status of a particular transport protocol on a system.**

   ```
   $ netstat -P transport-protocol
   ```

   Values for the *transport-protocol* variable are `tcp`, `sctp`, or `udp`.

Example 6–2 Displaying the Status of the TCP and SCTP Transport Protocols

This example shows the output of the basic `netstat` command. Note that IPv4-only information is displayed.

```
$ netstat

TCP: IPv4
   Local Address      Remote Address    Swind Send-Q  Rwind Recv-Q      State
------------------ -------------------- ----- ------  ----- ------     -------
lhost-1.login      abc.def.local.Sun.COM.980 49640      0     49640    0 ESTABLISHED
lhost-1.login      ghi.jkl.local.Sun.COM.1020 49640     1     49640    0 ESTABLISHED
remhost-1.1014     mno.pqr.remote.Sun.COM.nfsd 49640    0     49640    0 TIME_WAIT
SCTP:
Local Address    Remote Address  Swind  Send-Q  Rwind  Recv-Q StrsI/O  State
---------------- --------------  -----  ------ ------ ------  ------   -------
 *.echo            0.0.0.0            0       0 102400      0   128/1   LISTEN
 *.discard         0.0.0.0            0       0 102400      0   128/1   LISTEN
 *.9001            0.0.0.0            0       0 102400      0   128/1   LISTEN
```

Example 6–3 Displaying the Status of a Particular Transport Protocol

This example shows the results when you specify the `-P` option of `netstat`.

```
$ netstat -P tcp

TCP: IPv4
   Local Address      Remote Address    Swind Send-Q  Rwind Recv-Q      State
------------------ -------------------- ----- ------  ----- ------     -------
lhost-1.login      abc.def.local.Sun.COM.980 49640      0     49640    0 ESTABLISHED
lhost.login        ghi.jkl.local.Sun.COM.1020 49640     1     49640    0 ESTABLISHED
remhost.1014       mno.pqr.remote.Sun.COM.nfsd 49640    0     49640    0 TIME_WAIT

TCP: IPv6
 Local Address    Remote Address        Swind Send-Q Rwind Recv-Q   State If
---------------- ---------------------- ------ ----- ------ ------------ -----
localhost.38983   localhost.32777       49152      0 49152      0 ESTABLISHED
localhost.32777   localhost.38983       49152      0 49152      0 ESTABLISHED
localhost.38986   localhost.38980       49152      0 49152      0 ESTABLISHED
```

▼ How to Display Network Interface Status

The `i` option of the `netstat` command shows the state of the network interfaces that are configured on the local system. With this option, you can determine the number of packets a system transmits and receives on each network.

- **Display the status of interfaces on the network.**

  ```
  $ netstat -i
  ```

Example 6–4 Network Interface Status Display

The next example shows the status of IPv4 and IPv6 packet flow through the host's interfaces.

For example, the input packet count (`Ipkts`) that is displayed for a server can increase each time a client tries to boot, while the output packet count (`Opkts`) remains steady. This outcome suggests that the server is seeing the boot request packets from the client. However, the server does not know to respond to them. This confusion might be caused by an incorrect address in the `hosts`, or `ethers` database.

However, if the input packet count is steady over time, then the machine does not see the packets at all. This outcome suggests a different type of failure, possibly a hardware problem.

```
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback      localhost      142    0     142    0     0      0
bge0  1500 host58        host58        1106302 0     52419  0     0      0

Name  Mtu  Net/Dest      Address                   Ipkts  Ierrs Opkts  Oerrs Collis
lo0   8252 localhost     localhost                  142    0     142    0     0
bge0  1500 fe80::a00:20ff:feb9:4c54/10 fe80::a00:20ff:feb9:4c54 1106305 0 52422 0  0
```

▼ How to Display the Status of Sockets

The `-a` option of the `netstat` command enables you to view the status of sockets on the local host.

- **Type the following to display the status of sockets and routing table entries:**

  You can use your user account to run this option of `netstat`.

  ```
  % netstat -a
  ```

Example 6–5 Displaying All Sockets and Routing Table Entries

The output of the `netstat -a` command shows extensive statistics. The following example shows portions of typical `netstat -a` output.

```
UDP: IPv4
   Local Address         Remote Address     State
-------------------- -------------------- -------
      *.bootpc                            Idle
host85.bootpc                             Idle
      *.*                                 Unbound
      *.*                                 Unbound
      *.sunrpc                            Idle
      *.*                                 Unbound
```

```
      *.32771                            Idle
      *.sunrpc                           Idle
      *.*                                Unbound
      *.32775                            Idle
      *.time                             Idle
       .
       .
      *.daytime                          Idle
      *.echo                             Idle
      *.discard                          Idle

UDP: IPv6
   Local Address                      Remote Address                    State      If
--------------------------------- --------------------------------- ---------- -----
      *.*                                                           Unbound
      *.*                                                           Unbound
      *.sunrpc                                                      Idle
      *.*                                                           Unbound
      *.32771                                                       Idle
      *.32778                                                       Idle
      *.syslog                                                      Idle
      .
      .
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
      *.*                  *.*                0      0 49152      0 IDLE
localhost.4999             *.*                0      0 49152      0 LISTEN
      *.sunrpc             *.*                0      0 49152      0 LISTEN
      *.*                  *.*                0      0 49152      0 IDLE
      *.sunrpc             *.*                0      0 49152      0 LISTEN
      .
      .
      *.printer            *.*                0      0 49152      0 LISTEN
      *.time               *.*                0      0 49152      0 LISTEN
      *.daytime            *.*                0      0 49152      0 LISTEN
      *.echo               *.*                0      0 49152      0 LISTEN
      *.discard            *.*                0      0 49152      0 LISTEN
      *.chargen            *.*                0      0 49152      0 LISTEN
      *.shell              *.*                0      0 49152      0 LISTEN
      *.shell              *.*                0      0 49152      0 LISTEN
      *.kshell             *.*                0      0 49152      0 LISTEN
      *.login
       .
       .
           *.*                0      0 49152      0 LISTEN
   *TCP: IPv6
 Local Address            Remote Address        Swind Send-Q Rwind Recv-Q   State If
---------------------- ---------------------- ----- ------ ----- ------     ----
   *.*                      *.*                  0      0 49152      0      IDLE
   *.sunrpc                 *.*                  0      0 49152      0      LISTEN
   *.*                      *.*                  0      0 49152      0      IDLE
   *.32774                  *.*                  0      0 49152
```

How to Display the Status of Transmissions for Packets of a Specific Address Type

Use the -f option of the netstat command to view statistics related to packet transmissions of a particular address family.

- **View statistics for transmissions of either IPv4 or IPv6 packets.**

  ```
  $ netstat -f inet | inet6
  ```

  To view IPv4 transmission information, type inet as the argument to netstat -f. Use inet6 as the argument to netstat -f to view IPv6 information.

Example 6–6 Status of IPv4 Packet Transmission

The following example shows output from the netstat -f inet command.

```
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
host58.734         host19.nfsd       49640      0 49640      0 ESTABLISHED
host58.38063       host19.32782      49640      0 49640      0 CLOSE_WAIT
host58.38146       host41.43601      49640      0 49640      0 ESTABLISHED
host58.996         remote-host.login 49640      0 49206      0 ESTABLISHED
```

Example 6–7 Status of IPv6 Packet Transmission

The following example shows output from the netstat -f inet6 command.

```
TCP: IPv6
 Local Address          Remote Address        Swind Send-Q Rwind Recv-Q   State    If
------------------ ------------------------ ----- ------ ----- ------ --------- -----
localhost.38065         localhost.32792       49152   0 49152      0    ESTABLISHED
localhost.32792         localhost.38065       49152   0 49152      0    ESTABLISHED
localhost.38089         localhost.38057       49152   0 49152      0    ESTABLISHED
```

How to Display the Status of Known Routes

The -r option of the netstat command displays the routing table for the local host. This table shows the status of all routes that the host knows about. You can run this option of netstat from your user account.

- **Display the IP routing table.**

  ```
  $ netstat -r
  ```

Example 6–8 Routing Table Output by the netstat Command

The following example shows output from the netstat -r command.

```
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
host15               myhost               U         1  31059  bge0
10.0.0.14            myhost               U         1      0  bge0
default              distantrouter        UG        1      2  bge0
localhost            localhost            UH        42019361  lo0

Routing Table: IPv6
  Destination/Mask      Gateway                           Flags Ref  Use   If
--------------------- --------------------------          ----- --- ------ -----
2002:0a00:3010:2::/64 2002:0a00:3010:2:1b2b:3c4c:5e6e:abcd U     1   0      bge0:1
fe80::/10             fe80::1a2b:3c4d:5e6f:12a2           U     1   23     bge0
ff00::/8              fe80::1a2b:3c4d:5e6f:12a2           U     1   0      bge0
default               fe80::1a2b:3c4d:5e6f:12a2           UG    1   0      bge0
localhost             localhost                           UH    9   21832  lo0
```

The following table describes the meaning of the various parameters of the screen output of the `netstat –r` command.

Parameter	Description
`Destination` `Destination/Mask`	Specifies the host that is the destination endpoint of the route. Note that the IPv6 routing table shows the prefix for a 6to4 tunnel endpoint (`2002:0a00:3010:2::/64`) as the route destination endpoint.
`Gateway`	Specifies the gateway to use for forwarding packets.
`Flags`	Indicates the current status of the route. The `U` flag indicates that the route is up. The `G` flag indicates that the route is to a gateway.
`Use`	Shows the number of packets sent.
`Interface`	Indicates the particular interface on the local host that is the source endpoint of the transmission.

Probing Remote Hosts With the `ping` Command

You can use the `ping` command to determine the status of a remote host. When you run `ping`, the ICMP protocol sends a datagram to the host that you specify, asking for a response. ICMP is the protocol responsible for error handling on a TCP/IP network. When you use `ping`, you can find out whether an IP connection exists for the specified remote host.

The following is the basic syntax of `ping`:

`/usr/sbin/ping` *host [timeout]*

In this syntax, *host* is the name of the remote host. The optional *timeout* argument indicates the time in seconds for the `ping` command to continue trying to reach the remote host. The default is 20 seconds. For additional syntax and options, refer to the `ping(1M)` man page.

How to Determine if a Remote Host Is Running

- **Type the following form of the `ping` command:**

  ```
  $ ping hostname
  ```

  If host *hostname* is accepting ICMP transmissions, this message is displayed:

  ```
  hostname is alive
  ```

  This message indicates that *hostname* responded to the ICMP request. However, if *hostname* is down or cannot receive the ICMP packets, you receive the following response from the `ping` command:

  ```
  no answer from hostname
  ```

How to Determine if a Host Is Dropping Packets

Use the `-s` option of the `ping` command to determine if a remote host is running but nevertheless losing packets.

- **Type the following form of the `ping` command:**

  ```
  $ ping -s hostname
  ```

Example 6–9 `ping` Output for Detecting Packet Dropping

The `ping -s` *hostname* command continually sends packets to the specified host until you send an interrupt character or a time out occurs. The responses on your screen resemble the following:

```
& ping -s host1.domain8
PING host1.domain8 : 56 data bytes
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=0. time=1.67 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=1. time=1.02 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=2. time=0.986 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=3. time=0.921 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=4. time=1.16 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=5. time=1.00 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=5. time=1.980 ms

^C

----host1.domain8  PING Statistics----
7 packets transmitted, 7 packets received, 0% packet loss
round-trip (ms)  min/avg/max/stddev = 0.921/1.11/1.67/0.26
```

The packet-loss statistic indicates whether the host has dropped packets. If `ping` fails, check the status of the network that is reported by the `ipadm` and `netstat` commands. Refer to

"Monitoring IP Interfaces and Addresses" in *System Administration Guide: Network Interfaces and Network Virtualization* and "Monitoring Network Status With the `netstat` Command" on page 129.

Administering and Logging Network Status Displays

The following tasks show how to check the status of the network by using well-known networking commands.

▼ How to Control the Display Output of IP-Related Commands

You can control the output of the `netstat` command to display IPv4 information only, or both IPv4 and IPv6 information.

1. **Create the `/etc/default/inet_type` file.**

2. **Add one of the following entries to `/etc/default/inet_type`, as required for your network:**

   - To display IPv4 information only:

     ```
     DEFAULT_IP=IP_VERSION4
     ```

   - To display both IPv4 and IPv6 information:

     ```
     DEFAULT_IP=BOTH
     ```

     Or

     ```
     DEFAULT_IP=IP_VERSION6
     ```

     For more information about the `inet_type` file, see the `inet_type(4)` man page.

Note – The `-f` flag in the `netstat` command overrides the values set in the `inet_type` file.

Example 6–10 Controlling Output to Select IPv4 and IPv6 Information

- When you specify the `DEFAULT_IP=BOTH` or `DEFAULT_IP=IP_VERSION6` variable in the `inet_type` file, you should have the following output:

```
% ipadm show-addr
ADDROBJ       TYPE       STATE   ADDR
lo0/v4        static     ok      127.0.0.1/8
bge0/v4       static     ok      10.46.86.54/24
lo0/v6        static     ok      ::1/128
bge0/v6       addrconf   ok      fe80::a00:fe73:56a8/10
bge0/v6add    static     ok      2001:db8:3c4d:5:a00:fe73:56a8/64
```

- When you specify the `DEFAULT_IP=IP_VERSION4` variable in the `inet_type` file, you should have the following output:

```
% ipadm show-addr
ADDROBJ       TYPE       STATE   ADDR
lo0/v4        static     ok      127.0.0.1/8
bge0/v4       static     ok      10.46.86.54/24
```

How to Log Actions of the IPv4 Routing Daemon

If you suspect a malfunction of `routed`, the IPv4 routing daemon, you can start a log that traces the daemon's activity. The log includes all packet transfers when you start the `routed` daemon.

- **Create a log file of routing daemon actions:**

```
# /usr/sbin/in.routed /var/log-file-name
```

Caution – On a busy network, this command can generate almost continuous output.

Example 6–11 Network Log for the `in.routed` Daemon

The following example shows the beginning of the log that is created by the procedure "How to Log Actions of the IPv4 Routing Daemon" on page 138.

```
-- 2003/11/18 16:47:00.000000 --
Tracing actions started
RCVBUF=61440
Add interface lo0  #1   127.0.0.1      -->127.0.0.1/32
   <UP|LOOPBACK|RUNNING|MULTICAST|IPv4> <PASSIVE>
Add interface bge0 #2   10.10.48.112    -->10.10.48.0/25
    <UP|BROADCAST|RUNNING|MULTICAST|IPv4>
turn on RIP
Add    10.0.0.0        -->10.10.48.112      metric=0  bge0  <NET_SYN>
Add    10.10.48.85/25  -->10.10.48.112      metric=0  bge0  <IF|NOPROP>
```

How to Trace the Activities of the IPv6 Neighbor Discovery Daemon

If you suspect a malfunction of the IPv6 `in.ndpd` daemon, you can start a log that traces the daemon's activity. This trace is displayed on the standard output until terminated. This trace includes all packet transfers when you start the `in.ndpd` daemon.

1. **Start a trace of the `in.ndpd` daemon.**

```
# /usr/lib/inet/in.ndpd -t
```

2. **Terminate the trace as needed by typing Control-C.**

Example 6–12 Trace of the `in.ndpd` Daemon

The following output shows the beginning of a trace of `in.ndpd`.

```
# /usr/lib/inet/in.ndpd -t
Nov 18 17:27:28 Sending solicitation to  ff02::2 (16 bytes) on bge0
Nov 18 17:27:28         Source LLA: len 6 <08:00:20:b9:4c:54>
Nov 18 17:27:28 Received valid advert from fe80::a00:20ff:fee9:2d27 (88 bytes) on bge0
Nov 18 17:27:28         Max hop limit: 0
Nov 18 17:27:28         Managed address configuration: Not set
Nov 18 17:27:28         Other configuration flag: Not set
Nov 18 17:27:28         Router lifetime: 1800
Nov 18 17:27:28         Reachable timer: 0
Nov 18 17:27:28         Reachable retrans timer: 0
Nov 18 17:27:28         Source LLA: len 6 <08:00:20:e9:2d:27>
Nov 18 17:27:28         Prefix: 2001:08db:3c4d:1::/64
Nov 18 17:27:28                 On link flag:Set
Nov 18 17:27:28                 Auto addrconf flag:Set
Nov 18 17:27:28                 Valid time: 2592000
Nov 18 17:27:28                 Preferred time: 604800
Nov 18 17:27:28         Prefix: 2002:0a00:3010:2::/64
Nov 18 17:27:28                 On link flag:Set
Nov 18 17:27:28                 Auto addrconf flag:Set
Nov 18 17:27:28                 Valid time: 2592000
Nov 18 17:27:28                 Preferred time: 604800
```

Displaying Routing Information With the `traceroute` Command

The `traceroute` command traces the route an IP packet follows to a remote system. For technical details about `traceroute`, see the `traceroute(1M)` man page.

You use the `traceroute` command to uncover any routing misconfiguration and routing path failures. If a particular host is unreachable, you can use `traceroute` to see what path the packet follows to the remote host and where possible failures might occur.

The `traceroute` command also displays the round trip time for each gateway along the path to the target host. This information can be useful for analyzing where traffic is slow between the two hosts.

▼ How to Find Out the Route to a Remote Host

- **Type the following to discover the route to a remote system:**

  `%` **`traceroute`** *destination-hostname*

  You can run this form of the `traceroute` command from your user account.

Example 6–13 Using the traceroute Command to Show the Route to a Remote Host

The following output from the traceroute command shows the seven–hop path a packet follows from the local system nearhost to the remote system farhost. The output also shows the times for a packet to traverse each hop.

```
istanbul% traceroute farhost.faraway.com
    traceroute to farhost.faraway.com (172.16.64.39), 30 hops max, 40 byte packets
     1  frbldg7c-86 (172.16.86.1)  1.516 ms  1.283 ms  1.362 ms
     2  bldg1a-001 (172.16.1.211)  2.277 ms  1.773 ms  2.186 ms
     3  bldg4-bldg1 (172.16.4.42)  1.978 ms  1.986 ms  13.996 ms
     4  bldg6-bldg4 (172.16.4.49)  2.655 ms  3.042 ms  2.344 ms
     5  ferbldg11a-001 (172.16.1.236)  2.636 ms  3.432 ms  3.830 ms
     6  frbldg12b-153 (172.16.153.72)  3.452 ms  3.146 ms  2.962 ms
     7  sanfrancisco (172.16.64.39)  3.430 ms  3.312 ms  3.451 ms
```

How to Trace All Routes

This procedure uses the -a option of the traceroute command to trace all routes.

- **Type the following command on the local system:**

  % **traceroute -a***host-name*

  You can run this form of the traceroute command from your user account.

Example 6–14 Tracing All Routes to a Dual-Stack Host

This example shows all possible routes to a dual-stack host.

```
% traceroute -a v6host.remote.com
traceroute: Warning: Multiple interfaces found; using 2::56:a0:a8 @ eri0:2
traceroute to v6host (2001:db8:4a3b::102:a00:fe79:19b0),30 hops max, 60 byte packets
 1  v6-rout86 (2001:db8:4a3b:56:a00:fe1f:59a1)  35.534 ms  56.998 ms *
 2  2001:db8::255:0:c0a8:717  32.659 ms  39.444 ms *
 3  farhost.faraway.COM (2001:db8:4a3b::103:a00:fe9a:ce7b)  401.518 ms  7.143 ms *
 4  distant.remote.com (2001:db8:4a3b::100:a00:fe7c:cf35)  113.034 ms  7.949 ms *
 5  v6host (2001:db8:4a3b::102:a00:fe79:19b0)  66.111 ms *  36.965 ms

traceroute to v6host.remote.com  (192.168.10.75),30 hops max,40 byte packets
 1  v6-rout86 (172.16.86.1)  4.360 ms  3.452 ms  3.479 ms
 2  flrmpj17u.here.COM (172.16.17.131)  4.062 ms  3.848 ms  3.505 ms
 3  farhost.farway.com (10.0.0.23)  4.773 ms *  4.294 ms
 4  distant.remote.com (192.168.10.104)  5.128 ms  5.362 ms *
 5  v6host  (192.168.15.85)  7.298 ms  5.444 ms *
```

Monitoring Packet Transfers With the snoop Command

You can use the snoop command to monitor the state of data transfers. snoop captures network packets and displays their contents in the format that you specify. Packets can be displayed as soon as they are received, or saved to a file. When snoop writes to an intermediate file, packet loss under busy trace conditions is unlikely. snoop itself is then used to interpret the file.

To capture packets to and from the default interface in promiscuous mode, you must assume the Network Management role or become superuser. In summary form, snoop displays only the data that pertains to the highest-level protocol. For example, an NFS packet only displays NFS information. The underlying RPC, UDP, IP, and Ethernet frame information is suppressed but can be displayed if either of the verbose options is chosen.

Use snoop frequently and consistently to become familiar with normal system behavior. For assistance in analyzing packets, look for a recent white paper and RFC, and seek the advice of an expert in a particular area, such as NFS or NIS. For details on using snoop and its options, refer to the snoop(1M) man page.

▼ How to Check Packets From All Interfaces

1. **Print information about the interfaces that are attached to the system.**

   ```
   # ipadm show-if
   ```

   The snoop command normally uses the first non-loopback device, typically the primary network interface.

2. **Begin packet capture by typing snoop without arguments, as shown in Example 6–15.**

3. **Use Control-C to halt the process.**

Example 6–15 Output From the snoop Command

The basic snoop command returns output that resembles the following, for a dual-stack host.

```
% snoop
Using device /dev/bge (promiscuous mode)
router5.local.com -> router5.local.com ARP R 10.0.0.13, router5.local.com is
    0:10:7b:31:37:80
router5.local.com -> BROADCAST     TFTP Read "network-confg" (octet)
myhost -> DNSserver.local.com      DNS C 192.168.10.10.in-addr.arpa. Internet PTR ?
DNSserver.local.com  myhost        DNS R 192.168.10.10.in-addr.arpa. Internet PTR
   niserve2.
.
.
.
fe80::a00:20ff:febb:e09 -> ff02::9 RIPng R (5 destinations)
```

The packets that are captured in this output show a remote login section, including lookups to the NIS and DNS servers for address resolution. Also included are periodic ARP packets from the local router and advertisements of the IPv6 link-local address to `in.ripngd`.

▼ How to Capture snoop Output Into a File

1 Capture a snoop session into a file.

```
# snoop -o filename
```

For example:

```
# snoop -o /tmp/cap
Using device /dev/eri (promiscuous mode)
30 snoop: 30 packets captured
```

In the example, 30 packets have been captured in a file named `/tmp/cap`. The file can be in any directory with enough disk space. The number of packets that are captured is displayed on the command line, enabling you to press Control-C to abort at any time.

`snoop` creates a noticeable networking load on the host machine, which can distort the results. To see the actual results, run `snoop` from a third system.

2 Inspect the snoop output captures file.

```
# snoop -i filename
```

Example 6–16 Contents of a snoop Output Captures File

The following output shows a variety of captures such as you might receive as output from the `snoop -i` command.

```
# snoop -i /tmp/cap
1   0.00000 fe80::a00:20ff:fee9:2d27 -> fe80::a00:20ff:fecd:4375
    ICMPv6 Neighbor advertisement
...
10  0.91493    10.0.0.40 -> (broadcast)  ARP C Who is 10.0.0.40, 10.0.0.40 ?
34  0.43690 nearserver.here.com  -> 224.0.1.1  IP  D=224.0.1.1 S=10.0.0.40 LEN=28,
      ID=47453, TO =0x0, TTL=1
35  0.00034  10.0.0.40 -> 224.0.1.1    IP  D=224.0.1.1 S=10.0.0.40 LEN=28, ID=57376,
     TOS=0x0, TTL=47
```

▼ How to Check Packets Between an IPv4 Server and a Client

1 Establish a snoop system off a hub that is connected to either the client or the server.

The third system (the `snoop` system) checks all the intervening traffic, so the `snoop` trace reflects what is actually happening on the wire.

2. **Type snoop with options and save the output to a file.**

3. **Inspect and interpret the output.**

   Refer to RFC 1761, Snoop Version 2 Packet Capture File Format (`http://www.ietf.org/rfc/rfc1761.txt?number=1761`) for details of the snoop capture file.

How to Monitor IPv6 Network Traffic

You can use the snoop command to display only IPv6 packets.

- **Capture IPv6 packets.**

  ```
  # snoop ip6
  ```

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

Example 6–17 Displaying Only IPv6 Network Traffic

The following example shows typical output such as you might receive from running the snoop ip6 command on a node.

```
# snoop ip6
fe80::a00:20ff:fecd:4374 -> ff02::1:ffe9:2d27 ICMPv6 Neighbor solicitation
fe80::a00:20ff:fee9:2d27 -> fe80::a00:20ff:fecd:4375 ICMPv6 Neighbor
      solicitation
fe80::a00:20ff:fee9:2d27 -> fe80::a00:20ff:fecd:4375 ICMPv6 Neighbor
      solicitation
fe80::a00:20ff:febb:e09 -> ff02::9      RIPng R (11 destinations)
fe80::a00:20ff:fee9:2d27 -> ff02::1:ffcd:4375 ICMPv6 Neighbor solicitation
```

Monitoring Packets by Using IP Layer Devices

IP layer devices are introduced in Oracle Solaris to enhance IP observability. These devices provide access to all packets with addresses that are associated with the system's network interface. The addresses include local addresses as well as addresses that are hosted on non-loopback interfaces or logical interfaces. The observable traffic can be both IPv4 and IPv6 addresses. Thus, you can monitor all traffic that is destined to the system. The traffic can be loopback IP traffic, packets from remote machines, packets that are being sent from the system, or all forwarded traffic.

With IP layer devices, an administrator for a global zone can monitor traffic between zones as well as within a zone. An administrator of a non-global zone can also observe traffic that is sent and received by that zone.

To monitor traffic on the IP layer, a new option, `-I`, is added to the snoop command. This option specifies for the command to use the new IP layer devices instead of the underlying link-layer device to display traffic data.

Note – To understand the distinctions between layers, see "Data Encapsulation and the TCP/IP Protocol Stack" in *System Administration Guide: IP Services*.

How to Check Packets on the IP Layer

1. **If necessary, print the information about the interfaces that are attached to the system.**

   ```
   # ipadm show-if
   ```

2. **Capture IP traffic on a specific interface.**

   ```
   # snoop -I interface [-V | -v]
   ```

Examples

All the examples are based on the following system configuration:

```
# ipadm show-addr
ADDROBJ      TYPE       STATE   ADDR
lo0/v4       static     ok      127.0.0.1/8
bge0/v4      static     ok      192.68.25.5/24
lo0/?        static     ok      127.0.0.1/8
bge0/?       static     ok      172.0.0.3/24
bge0/?       static     ok      172.0.0.1/24
lo0/?        static     ok      127.0.0.1/8
```

Suppose that two zones, `sandbox` and `toybox`, are using the following IP addresses:

- `sandbox` – `172.0.0.3`
- `toybox` – `172.0.0.1`

You can issue the `snoop -I` command on the different interfaces on the system. The packet information that is displayed depends on whether you are an administrator for the global zone or for the non-global zone.

EXAMPLE 6–18 Traffic on the Loopback Interface

```
# snoop -I lo0
Using device ipnet/lo0 (promiscuous mode)
   localhost -> localhost    ICMP Echo request (ID: 5550 Sequence number: 0)
   localhost -> localhost    ICMP Echo reply (ID: 5550 Sequence number: 0)
```

To generate a verbose output, use the `-v` option.

```
# snoop -v -I lo0
Using device ipnet/lo0 (promiscuous mode)
IPNET:  ----- IPNET Header -----
IPNET:
IPNET:  Packet 1 arrived at 10:40:33.68506
```

```
IPNET:  Packet size = 108 bytes
IPNET:  dli_version = 1
IPNET:  dli_type = 4
IPNET:  dli_srczone = 0
IPNET:  dli_dstzone = 0
IPNET:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
...
```

Support for observing packets on the IP layer introduces a new ipnet header that precedes the packets that are being observed. Both the source and destination IDs are indicated. The '0' ID indicates that the traffic is being generated from the global zone.

EXAMPLE 6–19 Packet Flow in the `bge0` Device in Local Zones

```
# snoop -I bge0
Using device ipnet/bge0 (promiscuous mode)
toybox -> sandbox TCP D=22 S=62117 Syn Seq=195630514 Len=0 Win=49152 Options=<mss
sandbox -> toybox TCP D=62117 S=22 Syn Ack=195630515 Seq=195794440 Len=0 Win=49152
toybox -> sandbox TCP D=22 S=62117 Ack=195794441 Seq=195630515 Len=0 Win=49152
sandbox -> toybox TCP D=62117 S=22 Push Ack=195630515 Seq=195794441 Len=20 Win=491
```

The output shows traffic that occurs in the different zones within the system. You can see all packets that are associated with the `bge0` IP addresses, including packets that are locally delivered to other zones. If you generate a verbose output, you can see the zones that are involved in the flow of packets.

```
# snoop -I bge0 -v port 22
IPNET:  ----- IPNET Header -----
IPNET:
IPNET:  Packet 5 arrived at 15:16:50.85262
IPNET:  Packet size = 64 bytes
IPNET:  dli_version = 1
IPNET:  dli_type = 0
IPNET:  dli_srczone = 0
IPNET:  dli_dstzone = 1
IPNET:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:         .... ..0. = not ECN capable transport
IP:         .... ...0 = no ECN congestion experienced
IP:   Total length = 40 bytes
IP:   Identification = 22629
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
```

EXAMPLE 6–19 Packet Flow in the bge0 Device in Local Zones *(Continued)*

```
IP:   Fragment offset = 0 bytes
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = 0000
IP:   Source address = 172.0.0.1, 172.0.0.1
IP:   Destination address = 172.0.0.3, 172.0.0.3
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 46919
TCP:  Destination port = 22
TCP:  Sequence number = 3295338550
TCP:  Acknowledgement number = 3295417957
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x10
TCP:        0... .... = No ECN congestion window reduced
TCP:        .0.. .... = No ECN echo
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP         .... 0... = No push
TCP         .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 49152
TCP:  Checksum = 0x0014
TCP:  Urgent pointer = 0
TCP:  No options
TCP:
```

The ipnet header indicates that the packet is coming from the global zone (ID 0) to Sandbox (ID 1).

EXAMPLE 6–20 Observing Traffic by Identifying the Zone

```
# snoop -I hme0 sandboxsnoop -I bge0 sandbox
Using device ipnet/hme0 (promiscuous mode)
toybox -> sandbox TCP D=22 S=61658 Syn Seq=374055417 Len=0 Win=49152 Options=<mss
sandbox -> toybox TCP D=61658 S=22 Syn Ack=374055418 Seq=374124525 Len=0 Win=49152
toybox -> sandbox TCP D=22 S=61658 Ack=374124526 Seq=374055418 Len=0 Win=49152
#
```

The ability to observe packets by identifying zone is useful in systems that have multiple zones. Currently, you can only identify zone by using the zone ID. Using snoop with zone names is not supported.

Administering Default Address Selection

Oracle Solaris enables a single interface to have multiple IP addresses. For example, technologies, such as network multipathing (IPMP) enable multiple network interface cards (NICs) to connect to the same IP link layer. That link can have one or more IP addresses. Additionally, interfaces on IPv6-enabled systems have a link-local IPv6 address, at least one IPv6 routing address, and an IPv4 address for at least one interface.

When the system initiates a transaction, an application makes a call to the `getaddrinfo` socket. `getaddrinfo` discovers the possible address in use on the destination system. The kernel then prioritizes this list to find the best destination to use for the packet. This process is called *destination address ordering*. The Oracle Solaris kernel then selects the appropriate format for the source address, given the best destination address for the packet. The process is known as *address selection*. For more information on destination address ordering, see the `getaddrinfo(3SOCKET)` man page.

Both IPv4-only and dual-stack IPv4/IPv6 systems must perform default address selection. In most circumstances, you do not need to change the default address selection mechanisms. However, you might need to change the priority of address formats to support IPMP or to prefer 6to4 address formats, for example.

How to Administer the IPv6 Address Selection Policy Table

The following procedure explains how to modify the address selection policy table. For conceptual information about IPv6 default address selection, refer to "`ipaddrsel` Command" on page 207.

Caution – Do not change the IPv6 address selection policy table, except for the reasons shown in the next task. You can cause problems on the network with a badly constructed policy table. Be sure to save a backup copy of the policy table, as is done in the next procedure.

1 Review the current IPv6 address selection policy table.

```
# ipaddrsel
# Prefix                  Precedence Label
::1/128                           50 Loopback
::/0                              40 Default
2002::/16                         30 6to4
::/96                             20 IPv4_Compatible
::ffff:0.0.0.0/96                 10 IPv4
```

2 Make a backup copy of the default address policy table.

```
# cp /etc/inet/ipaddrsel.conf /etc/inet/ipaddrsel.conf.orig
```

3 Use a text editor to add your customizations to `/etc/inet/ipaddrsel.conf`.

Use the following syntax for entries in `/etc/inet/ipaddrsel`:

prefix/prefix-length precedence label [# comment]

Here are some common modifications that you might want to make to your policy table:

- Give the highest priority to 6to4 addresses.

```
2002::/16                       50 6to4
::1/128                         45 Loopback
```

  The 6to4 address format now has the highest priority, 50. Loopback, which previously had a 50 precedence, now has a 45 precedence. The other addressing formats remain the same.

- Designate a specific source address to be used in communications with a specific destination address.

```
::1/128                         50 Loopback
2001:1111:1111::1/128           40 ClientNet
2001:2222:2222::/48             40 ClientNet
::/0                            40 Default
```

  This particular entry is useful for hosts with only one physical interface. Here `2001:1111:1111::1/128` is preferred as the source address on all packets that are bound for destinations within network `2001:2222:2222::/48`. The 40 priority gives higher precedence to the source address `2001:1111:1111::1/128` than to other address formats configured for the interface.

- Favor IPv4 addresses over IPv6 addresses.

```
::ffff:0.0.0.0/96               60 IPv4
::1/128                         50 Loopback
.
.
```

  The IPv4 format `::ffff:0.0.0.0/96` has its precedence changed from the default 10 to 60, the highest priority in the table.

4 Load the modified policy table into the kernel.

```
ipaddrsel -f /etc/inet/ipaddrsel.conf
```

5 If the modified policy table has problems, restore the default IPv6 address selection policy table.

```
# ipaddrsel -d
```

▼ How to Modify the IPv6 Address Selection Table for the Current Session Only

When you edit the `/etc/inet/ipaddrsel.conf`, file, any modifications that you make persist across reboots. If you want the modified policy table to exist only in the current session, follow this procedure.

1. **Copy the contents of `/etc/inet/ipaddrsel` into *filename*, where *filename* represents a name of your choice.**

   ```
   # cp /etc/inet/ipaddrsel filename
   ```

2. **Edit the policy table in *filename* to your specifications.**

3. **Load the modified policy table into the kernel.**

   ```
   # ipaddrsel -f filename
   ```

   The kernel uses the new policy table until you reboot the system.

◆ ◆ ◆ CHAPTER 7

Configuring IP Tunnels

This chapter contains descriptions of IP tunnels as well as procedures for configuring and maintaining tunnels in Oracle Solaris.

What's New in IP Tunnel Administration

In this Oracle Solaris release, tunnel administration has been revised to become consistent with the new model for network data-link administration. Tunnels are now created and configured by using new `dladm` subcommands. Tunnels can now also use other data-link features of the new administration model. For example, support for administratively-chosen names allows tunnels to be assigned meaningful names. For a description of the new model to administer data links, see *Solaris Express Developer Edition What's New*. For more information about the `dladm` subcommands, see the `dladm`(1M) man page.

Overview of IP Tunnels

IP tunnels provide a means to transport data packets between domains when the protocol in those domains is not supported by intermediary networks. For example, with the introduction of the IPv6 protocol, IPv6 networks require a way to communicate outside their borders in an environment where most networks use the IPv4 protocol. Communication becomes possible by using tunnels. The IP tunnel provides a virtual link between two nodes that are reachable by using IP. The link can thus be used to transport IPv6 packets over the IPv4 networks to enable IPv6 communication between the two IPv6 sites.

Types of Tunnels

Tunneling involves the encapsulation of an IP packet within another packet. This encapsulation allows the packet to reach its destination through intermediary networks that do not support the packet's protocol.

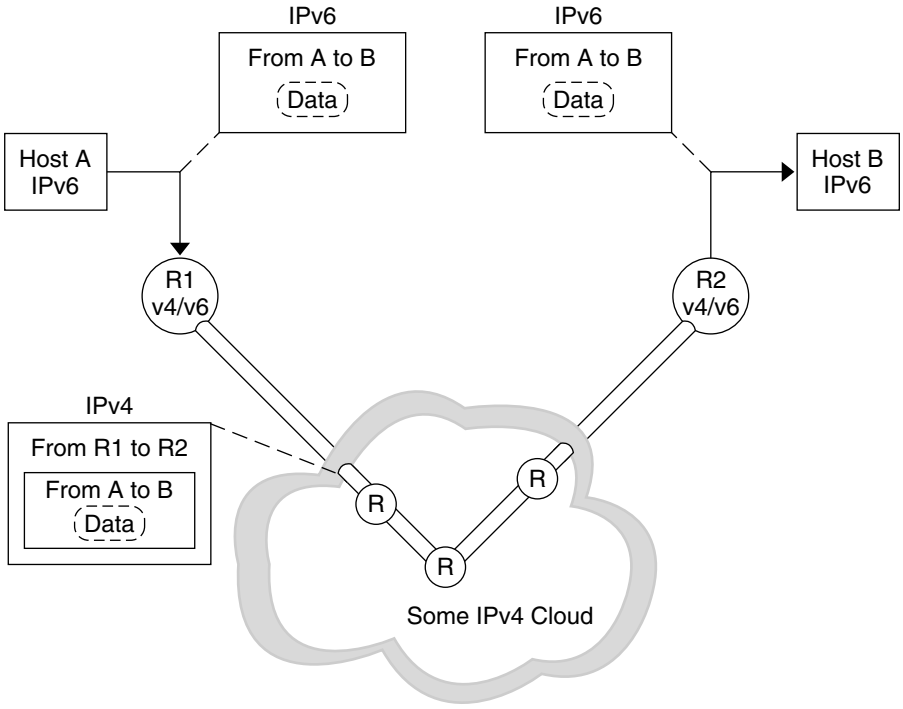
Tunnels differ depending on the type of packet encapsulation. The following types of tunnels are supported in Oracle Solaris:

- *IPv4 tunnels* – IPv4 or IPv6 packets are encapsulated in an IPv4 header and sent to a preconfigured unicast IPv4 destination. To indicate more specifically the packets that flow over the tunnel, IPv4 tunnels are also called either *IPv4 over IPv4 tunnels* or *IPv6 over IPv4 tunnels*.
- *IPv6 tunnels* – IPv4 or IPv6 packets are encapsulated in an IPv6 header and sent to a preconfigured unicast IPv6 destination. To indicate more specifically the packets that flow over the tunnel, IPv6 tunnels are also called either *IPv4 over IPv6 tunnels* or *IPv6 over IPv6 tunnels*.
- *6to4 tunnels* – IPv6 packets are encapsulated in an IPv4 header and sent to an IPv4 destination that is automatically determined on a per-packet basis. The determination is based on an algorithm that is defined in the 6to4 protocol.

Tunnels in the Combined IPv6 and IPv4 Network Environments

Most sites that have IPv6 domains communicate with other IPv6 domains by traversing IPv4 networks, which are more prevalent than IPv6–only networks. The following figure illustrates the tunneling mechanism between two IPv6 hosts through IPv4 routers, which are indicated in the figure by "R."

FIGURE 7–1 IPv6 Tunneling Mechanism



In the figure, the tunnel consists of two routers that are configured to have a virtual point-to-point link between the two routers over the IPv4 network.

An IPv6 packet is encapsulated within an IPv4 packet. The boundary router of the IPv6 network sets up a point-to-point tunnel over various IPv4 networks to the boundary router of the destination IPv6 network. The packet is transported over the tunnel to the destination boundary router, where the packet is decapsulated. The router then forwards the separate IPv6 packet to the destination node.

6to4 Tunnels

Oracle Solaris includes 6to4 tunnels as a preferred interim method for making the transition from IPv4 to IPv6 addressing. 6to4 tunnels enable isolated IPv6 sites to communicate across an automatic tunnel over an IPv4 network that does not support IPv6. To use 6to4 tunnels, you must configure a boundary router on your IPv6 network as one endpoint of the 6to4 automatic tunnel. Thereafter, the 6to4 router can participate in a tunnel to another 6to4 site, or, if required, to a native IPv6, non-6to4 site.

This section provides reference materials on the following 6to4 topics:

- Topology of a 6to4 tunnel
- Description of the packet flow across a 6to4 tunnel
- Topology of a tunnel between a 6to4 router and a 6to4 relay router
- Points to consider before you configure 6to4 relay router support

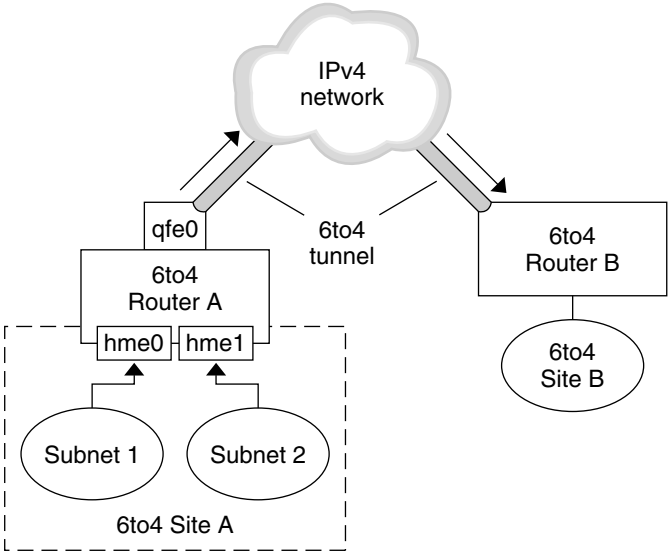
The following table describes additional tasks to configure 6to4 tunnels and the resources to obtain additional useful information.

Task or Detail	For Information
Tasks for configuring a 6to4 tunnel	"How to Configure a 6to4 Tunnel" on page 165
6to4-related RFC	RFC 3056, "Connection of IPv6 Domains via IPv4 Clouds" (`http://www.ietf.org/rfc/rfc3056.txt`)
Detailed information about the `6to4relay` command, which enables support for tunnels to a 6to4 relay router	`6to4relay`(1M)
6to4 security issues	Security Considerations for 6to4 (`http://www.ietf.org/rfc/rfc3964.txt`)

Topology of a 6to4 Tunnel

A 6to4 tunnel provides IPv6 connectivity to all 6to4 sites everywhere. Likewise, the tunnel also functions a link to all IPv6 sites, including the native IPv6 internet, provided that the tunnel is configured to forward to a relay router. The following figure shows how a 6to4 tunnel provides this connectivity between 6to4 sites.

FIGURE 7–2 Tunnel Between Two 6to4 Sites



The figure depicts two isolated 6to4 networks, Site A and Site B. Each site has configured a router with an external connection to an IPv4 network. A 6to4 tunnel across the IPv4 network provides a connection to link 6to4 sites.

Before an IPv6 site can become a 6to4 site, you must configure at least one router interface for 6to4 support. This interface must provide the external connection to the IPv4 network. The address that you configure on `qfe0` must be globally unique. In this figure, boundary Router A's interface `qfe0` connects Site A to the IPv4 network. Interface `qfe0` must already be configured with an IPv4 address before you can configure `qfe0` as a 6to4 pseudo-interface.

In the figure, 6to4 Site A is composed of two subnets, which are connected to interfaces `hme0` and `hme1` on Router A. All IPv6 hosts on either subnet of Site A automatically reconfigure with 6to4-derived addresses upon receipt of the advertisement from Router A.

Site B is another isolated 6to4 site. To correctly receive traffic from Site A, a boundary router on Site B must be configured for 6to4 support. Otherwise, packets that the router receives from Site A are not recognized and are then dropped.

Packet Flow Through the 6to4 Tunnel

This section describes the flow of packets from a host at one 6to4 site to a host at a remote 6to4 site. This scenario uses the topology that is shown in Figure 7–2. Moreover, the scenario assumes that the 6to4 routers and the 6to4 hosts are already configured.

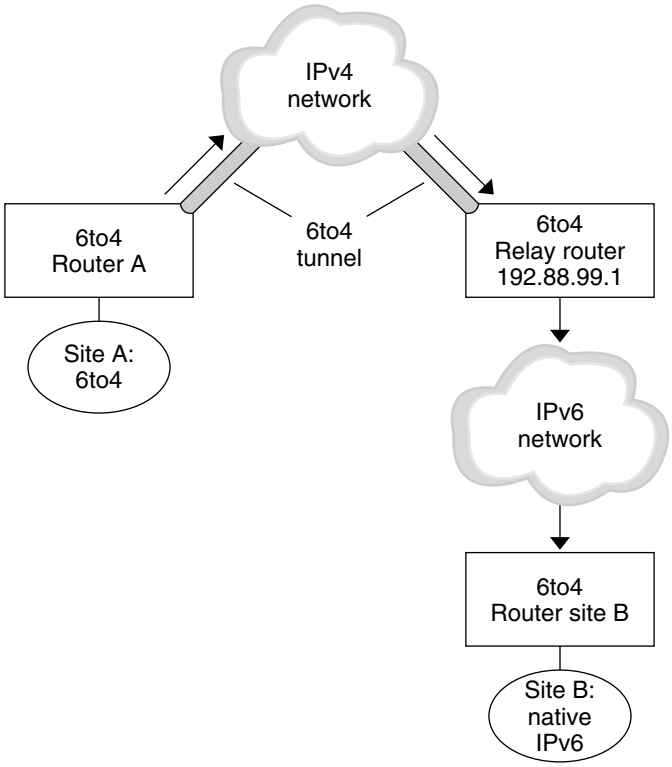
1. A host on Subnet 1 of 6to4 Site A sends a transmission, with a host at 6to4 Site B as the destination. Each packet header has a 6to4-derived source address and 6to4-derived destination address.

2. Site A's router encapsulates each 6to4 packet within an IPv4 header. In this process, the router sets the IPv4 destination address of the encapsulating header to Site B's router address. For each IPv6 packet that flows through the tunnel interface, the packet's IPv6 destination address also contains the IPv4 destination address. Thus, the router is able to determine the IPv4 destination address that is set on the encapsulating header. Then, the router uses standard IPv4 routing procedures to forward the packet over the IPv4 network.
3. Any IPv4 routers that the packets encounter use the packets' IPv4 destination address for forwarding. This address is the globally unique IPv4 address of the interface on Router B, which also serves as the 6to4 pseudo-interface.
4. Packets from Site A arrive at Router B, which decapsulates the IPv6 packets from the IPv4 header.
5. Router B then uses the destination address in the IPv6 packet to forward the packets to the recipient host at Site B.

Considerations for Tunnels to a 6to4 Relay Router

6to4 relay routers function as endpoints for tunnels from 6to4 routers that need to communicate with native IPv6, non-6to4 networks. Relay routers are essentially bridges between the 6to4 site and native IPv6 sites. Because this solution might be insecure, by default, Oracle Solaris does not enable 6to4 relay router support. However, if your site requires such a tunnel, you can use the `6to4relay` command to enable the following tunneling scenario.

FIGURE 7–3 Tunnel From a 6to4 Site to a 6to4 Relay Router



In Figure 7–3, 6to4 Site A needs to communicate with a node at the native IPv6 Site B. The figure shows the path of traffic from Site A onto a 6to4 tunnel over an IPv4 network. The tunnel has 6to4 Router A and a 6to4 relay router as its endpoints. Beyond the 6to4 relay router is the IPv6 network, to which IPv6 Site B is connected.

Packet Flow Between a 6to4 Site and a Native IPv6 Site

This section describes the flow of packets from a 6to4 site to a native IPv6 site. This scenario uses the topology that is shown in Figure 7–3.

1. A host on 6to4 Site A sends a transmission that specifies as the destination a host at native IPv6 Site B. Each packet header has a 6to4-derived address as its source address. The destination address is a standard IPv6 address.
2. Site A's 6to4 router encapsulates each packet within an IPv4 header, which has the IPv4 address of the 6to4 relay router as its destination. The 6to4 router uses standard IPv4 routing procedures to forward the packet over the IPv4 network. Any IPv4 routers that the packets encounter forward the packets to the 6to4 relay router.

3. The physically closest anycast 6to4 relay router to Site A retrieves the packets that are destined for the `192.88.99.1` anycast group.

Note – 6to4 relay routers that are part of the 6to4 relay router anycast group have the IP address `192.88.99.1`. This anycast address is the default address for 6to4 relay routers. If you need to use a specific 6to4 relay router, you can override the default and specify that router's IPv4 address.

---

4. The relay router decapsulates the IPv4 header from the 6to4 packets, revealing the native IPv6 destination address.
5. The relay router then sends the now IPv6-only packets onto the IPv6 network, where the packets are ultimately retrieved by a router at Site B. The router then forwards the packets to the destination IPv6 node.

Deploying Tunnels

To properly deploy IP tunnels, you need to perform two main tasks. First, you create the tunnel link. Then, you configure an IP interface over the tunnel. This section briefly describes the requirements for creating tunnels and their corresponding IP interfaces.

Requirements for Creating Tunnels

To successfully create tunnels, you must observe the following requirements:

- If you use host names instead of literal IP addresses, these names must resolve to valid IP addresses that are compatible with the tunnel type.
- The IPv4 or IPv6 tunnel that you create must not share the same tunnel source address and tunnel destination address with another configured tunnel.
- The IPv4 or IPv6 tunnel that you create must not share the same tunnel source address with an existing 6to4 tunnel.
- If you create a 6to4 tunnel, that tunnel must not share the same tunnel source address with another configured tunnel.

For information about setting up tunnels in your network, refer to "Planning for Tunnels in the Network Topology" on page 69.

Requirements for Tunnels and IP Interfaces

Each tunnel type has specific IP address requirements on the IP interface that you configure over the tunnel. The requirements are summarized in the following table.

TABLE 7–1 Tunnels and IP Interface Requirements

Tunnel Type	IP Interface Allowed Over Tunnel	IP Interface Requirement
IPv4 tunnel	IPv4 interface	Local and remote addresses are manually specified.
	IPv6 interface	Local and remote link-local addresses are automatically set when you issue the `ipadm create-addr -T addrconf` command. For details see the `ipadm`(1M) man page.
IPv6 tunnel	IPv4 interface	Local and remote addresses are manually specified.
	IPv6 interface	Local and remote link-local addresses are automatically set when you issue the `ipadm create-addr -T addrconf` command. For details see the `ipadm`(1M) man page.
6to4 tunnel	IPv6 interface only	Default IPv6 address is automatically selected when you issue the `ipadm create-if` command. For details see the `ipadm`(1M) man page.

You can override the default IPv6 interface address of 6to4 tunnels by specifying a different IPv6 address. Such additional IPv6 addresses can be included in the tunnel's `hostname6.`*tunnel-name* file. Otherwise, the file can remain empty.

Similarly, to override the link-local addresses that are automatically set for IPv6 interfaces over IPv4 or IPv6 tunnels, you can specify different source and destination addresses in the tunnel's host file.

Tunnel Configuration and Administration With the `dladm` Command

This section describes procedures that use the `dladm` command to configure tunnels.

dladm Subcommands

Previously, all aspects of IP tunneling configuration required the use of the ifconfig command. Beginning with this Oracle Solaris release, tunnel administration is now separated from IP interface configuration. The data-link aspect of IP tunnels is now administered with the dladm command. Additionally, IP interface configuration, including the IP tunnel interface, is performed with the ipadm command.

To maintain compatibility with the implementation in previous Oracle Solaris releases, the ifconfig command remains as a valid method for configuring tunnel links.

The following subcommands of dladm are used to configure IP tunnels:

- create-iptun
- modify-iptun
- show-iptun
- delete-iptun
- set-linkprop

For details about the dladm command, refer to the dladm(1M) man page.

Note – IP tunnel administration is closely associated with IPsec configuration. For example, IPsec virtual private networks (VPNs) are one of the primary uses of IP tunneling. For more information about security in Oracle Solaris, see Part III, "IP Security." To configure IPsec, see Chapter 19, "Configuring IPsec (Tasks)."

Configuring Tunnels (Task Map)

Task	Description	For Instructions
Create an IP tunnel.	Configure the tunnel to be used for communicating across networks.	"How to Create and Configure an IP Tunnel" on page 161
Modify a tunnel's configuration.	Change the tunnel's original parameters, such as the tunnel's source or destination address.	"How to Modify an IP Tunnel Configuration" on page 168
Display a tunnel configuration.	Show configuration information for either a specific tunnel or all of the system's IP tunnels.	"How to Display an IP Tunnel's Configuration" on page 169
Delete a tunnel.	Delete a tunnel configuration.	"How to Delete an IP Tunnel" on page 171

How to Create and Configure an IP Tunnel

1 Create the tunnel.

```
# dladm create-iptun [-t] -T type -a [local|remote]=addr,... tunnel-link
```

The following options or arguments are available for this command:

`-t` — Creates a temporary tunnel. By default, the command creates a persistent tunnel.

Note – If you want to configure a persistent IP interface over the tunnel, then you must create a persistent tunnel and not use the `-t` option.

`-T` *type* — Specifies the type of tunnel you want to create. This argument is required to create all tunnel types.

`-a [local|remote]=`*address,...* — Specifies literal IP addresses or host names that correspond to the local address and the remote tunnel address. The addresses must be valid and already created in the system. Depending on the type of tunnel, you specify either only one address, or both local and remote addresses. If specifying both local and remote addresses, you must separate the addresses with a comma.

- `IPv4` tunnels require local and remote IPv4 addresses to function.
- `IPv6` tunnels require local and remote IPv6 addresses to function.
- `6to4` tunnels require a local IPv4 address to function.

Note – For persistent IP tunnel data-link configurations, if you are using host names for addresses, these host names are saved in the configuration storage. During a subsequent system boot, if the names resolve to IP addresses that are different from the IP addresses used when the tunnel was created, then the tunnel acquires a new configuration.

tunnel-link — Specifies the IP tunnel link. With support for meaningful names in a network-link administration, tunnel names are no longer restricted to the type of tunnel that you are creating. Instead, a tunnel can be assigned any

administratively chosen name. Tunnel names consist of a string and the physical point of attachment (PPA) number, for example, *mytunnel0*. For rules governing the assignment of meaningful names, refer to "Rules for Valid Link Names" in *System Administration Guide: Network Interfaces and Network Virtualization*.

If you do not specify the tunnel link, then the name is automatically supplied according to the following naming conventions:

- For IPv4 tunnels: `ip.tun#`
- For IPv6 tunnels: `ip6.tun#`
- For 6to4 tunnels: `ip.6to4tun#`

The # is the lowest available PPA number for the tunnel type that you are creating.

2 (Optional) Set values for the hop limit or the encapsulation limit.

```
# dladm set-linkprop -p [hoplimit=value] [encaplimit=value] tunnel-link
```

hoplimit — Specifies the hop limit of the tunnel interface for tunneling over IPv6. The *hoplimit* is the equivalent of the IPv4 time to live (TTL) field for tunneling over IPv4.

encaplimit — Specifies the number of levels of nested tunneling that are allowed for a packet. This option applies only to IPv6 tunnels.

Specifies the number of levels of nested tunneling that are allowed for a packet. This option applies only to IPv6 tunnels.

Note – The values of that you set for `hoplimit` and `encaplimit` must remain within acceptable ranges. The `hoplimit` and `encaplimit` are tunnel link properties. Thus, these properties are administered by the same `dladm` subcommands as for other link properties. The subcommands are `dladm set-linkprop`, `dladm reset-linkprop`, and `dladm show-linkprop`. Refer to the `dladm`(1M) man page for the different subcommands that are used with the `dladm` command to administer links.

---

3 Create an IP interface over the tunnel.

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

where *tunnel-interface* uses the same name as the tunnel link.

4 Assign local and remote IP addresses to the tunnel interface.

```
# ipadm create-addr [-t] -T static -a local=address,remote=address addrobj
```

`-t`	Indicates a temporary IP configuration rather than a persistent IP configuration over the tunnel. If you do not use this option, then the IP interface configuration is a persistent configuration.
`-T static`	Indicates that static IP addresses are used instead of the dynamic IP procedures.
`-a local=`*address*`,remote=`*address*	Specifies the IP addresses of the tunnel interface. Both source and destination IP addresses are required, as represented by `local` and `remote`. Local and remote addresses can either be IPv4 or IPv6 addresses.
addrobj	Specifies the address object that owns the local and remote addresses. The `addrobj` must use the format *interface*/*user-specified-string*. The *user-specified-string* refers to a string of alphanumeric characters that begins with an alphabet character and has a maximum length of 32 characters.

For more information about the `ipadm` command and the different options to configure IP interfaces, including tunnel interfaces, see the `ipadm`(1M) man page and Part II, "Administering Single Interfaces," in *System Administration Guide: Network Interfaces and Network Virtualization*.

5 Add the tunnel configuration information to the `/etc/hosts` file.

6 (Optional) Verify the status of the tunnel's IP interface configuration.

```
# ipadm show-addr interface
```

Example 7–1 Creating an IPv6 Interface Over an IPv4 Tunnel

This example shows how to create a persistent IPv6 over IPv4 tunnel.

```
# dladm create-iptun -T ipv4 -a local=63.1.2.3,remote=192.4.5.6 private0
# dladm set-linkprop -p hoplimit=200 private0
# ipadm create-ip private0
# ipadm create-addr -T addrconf private0/v6
# ipadm show-addr private/
ADDROBJ       TYPE     STATE   ADDR
private0/v6   static   ok      fe80::a08:392e/10 --> fe80::8191:9a56
```

To add alternative addresses, use the same syntax while using a different *user-specified-string* for *addrobj*. For example, you can add a global address as follows:

```
# ipadm create-addr -T static -a local=2001:db8:4728::1, \
remote=2001:db8:4728::2 private0/global
# ipadm show-addr private0/
```

```
ADDROBJ           TYPE       STATE   ADDR
private0/v6       addrconf   ok      fe80::a08:392e/10 --> fe80::8191:9a56
private0/global   static     ok      2001:db8:4728::1 --> 2001:db8:4728::2
```

Note that the prefix `2001:db8` for the IPv6 address is a special IPv6 prefix that is used specifically for documentation examples. For a description of IPv6 addresses and format, see "IPv6 Addressing Overview" on page 54.

Example 7–2 Creating an IPv4 Interface Over an IPv4 Tunnel

This example shows how to create a persistent IPv4 over IPv4 tunnel.

```
# dladm create-iptun -T ipv4 -a local=63.1.2.3,remote=192.4.5.6 vpn0
# ipadm create-ip vpn0
# ipadm create-addr -T static -a local=10.0.0.1,remote=10.0.0.2 vpn0/v4
# ipadm show-addr
ADDROBJ   TYPE     STATE   ADDR
lo0/v4    static   ok      127.0.0.1
vpn0/v4   static   ok      10.0.0.1-->10.0.0.2
```

You can further configure IPsec policy to provide secure connections for the packets that flow over this tunnel. For information about IPsec configuration, see Chapter 19, "Configuring IPsec (Tasks)."

Example 7–3 Creating an IPv6 Interface Over an IPv6 Tunnel

This example shows how to create a persistent IPv6 over IPv6 tunnel.

```
# dladm create-iptun -T ipv6 -a local=2001:db8:feed::1234,remote=2001:db8:beef::4321 \
tun0
# ipadm create-ip tun0
# ipadm create-addr -T addrconf tun0/v6
# ipadm show-addr
ADDROBJ   TYPE       STATE   ADDR
lo0/v6    static     ok      ::1/128
tun0/v6   addrconf   ok      2001:db8:feed::1234 --> 2001:db8:beef::4321
```

To add addresses such as a global address or alternative local and remote addresses, use the `ipadm` command as follows:

```
# ipadm create-addr -T static  \
-a local=2001:db8::4728:56bc,remote=2001:db8::1428:57ab tun0/alt
# ipadm show-addr tun0/
ADDROBJ    TYPE      STATE ADDR
tun0/v6    addrconf  ok    2001:db8:feed::1234 --> 2001:db8:beef::4321
tun0/alt   static    ok    2001:db8::4728:56bc --> 2001:db8::1428:57ab
```

How to Configure a 6to4 Tunnel

In 6to4 tunnels, a 6to4 router must act as the IPv6 router to the nodes in the network's 6to4 sites. Thus, when configuring a 6to4 router, that router must also be configured as an IPv6 router on its physical interfaces. For more information about IPv6 routing, see "IPv6 Routing" on page 220.

1 Create a 6to4 tunnel.

```
# dladm create-iptun -T 6to4 -a local=address  tunnel-link
```

The following options or arguments are available for this command:

`-a` local=*address* — Specifies the tunnel local address, which must already be existing in the system to be a valid address.

tunnel-link — Specifies the IP tunnel link. With support for meaningful names in a network-link administration, tunnel names are no longer restricted to the type of tunnel that you are creating. Instead, a tunnel can be assigned any administratively-chosen name. Tunnel names consist of a string and the PPA number, for example, *mytunnel0*. For rules governing the assignment of meaningful names, refer to "Rules for Valid Link Names" in *System Administration Guide: Network Interfaces and Network Virtualization*.

2 Create the tunnel IP interface.

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

where *tunnel-interface* uses the same name as the tunnel link.

3 (Optional) Add alternative IPv6 addresses for the tunnel's use.

4 Edit the `/etc/inet/ndpd.conf` file to advertise 6to4 routing by adding the following two lines:

```
if subnet-interface AdvSendAdvertisements 1
IPv6-address  subnet-interface
```

The first line specifies the subnet that receives the advertisement. The *subnet-interface* refers to the link to which the subnet is connected. The IPv6 address on the second line must have the 6to4 prefix `2000` that is used for IPv6 addresses in 6to4 tunnels.

For detailed information about the `ndpd.conf` file, refer to the `ndpd.conf(4)` man page.

5 Enable IPv6 forwarding.

```
# ipadm set-prop -p forwarding=on ipv6
```

6 **Reboot the router.**

Alternatively, you can issue a `sighup` to the `/etc/inet/in.ndpd` daemon to begin sending router advertisements. The IPv6 nodes on each subnet to receive the 6to4 prefix now autoconfigure with new 6to4-derived addresses.

7 **Add the new 6to4-derived addresses of the nodes to the name service that is used at the 6to4 site.**

For instructions, go to "Configuring Name Service Support for IPv6" on page 122.

Example 7–4 Creating a 6to4 Tunnel

In this example, the subnet interface is `bge0` to which the `/etc/inet/ndpd.conf` will refer in the appropriate step.

This example shows how to create a 6to4 tunnel. Note that only IPv6 interfaces can be configured over 6to4 tunnels.

```
# dladm create-iptun -T 6to4 -a local=192.168.35.10 tun0
# ipadm create-ip tun0
# ipadm show-addr
ADDROBJ       TYPE     STATE   ADDR
lo0/v4        static   ok      127.0.0.1/8
bge0/static   static   ok      192.168.35.10/24
lo0/v6        static   ok      ::1/128
tun0/_a       static   ok      2002:c0a8:57bc::1/64

# ipadm create-addr -T static -a 2002:c0a8:230a::2/16 tun0/a2
# ipadm create-addr -T static -a 2002:c0a8:230a::3/16 tun0/a3
# ipadm show-addr tun0/
ADDROBJ       TYPE     STATE   ADDR
lo0/v4        static   ok      127.0.0.1/8
bge0/static   static   ok      192.168.35.10/24
lo0/v6        static   ok      ::1/128
tun0/_a       static   ok      2002:c0a8:57bc::1/64
tun0/a2       static   ok      2002:c0a8:230a::2/16
tun0/a3       static   ok      2002:c0a8:230a::3/16

# vi /etc/inet/ndpd.conf
if bge0 AdvSendAdvertisements 1
2002:c0a8:57bc::1/64 bge0

# ipadm set-prop -p forwarding=on ipv6
```

Note that for 6to4 tunnels, the prefix for the IPv6 address is `2002`. For further explanations, see "Prefixes in IPv6" on page 55.

▼ How to Configure a 6to4 Tunnel to a 6to4 Relay Router

Caution – Because of major security issues, by default, 6to4 relay router support is disabled in Oracle Solaris. See "Security Issues When Tunneling to a 6to4 Relay Router" on page 175.

Before You Begin Before you enable a tunnel to a 6to4 relay router, you must have completed the following tasks:

- Configured a 6to4 router at your site, as explained in "How to Create and Configure an IP Tunnel" on page 161
- Reviewed the security issues that are involved in tunneling to a 6to4 relay router

1 Enable a tunnel to the 6to4 relay router by using either of the following formats:

- Enable a tunnel to an anycast 6to4 relay router.

  ```
  # /usr/sbin/6to4relay -e
  ```

  The `-e` option sets up a tunnel between the 6to4 router and an anycast 6to4 relay router. Anycast 6to4 relay routers have the well-known IPv4 address `192.88.99.1`. The anycast relay router that is physically nearest to your site becomes the endpoint for the 6to4 tunnel. This relay router then handles packet forwarding between your 6to4 site and a native IPv6 site.

  For detailed information about anycast 6to4 relay routers, refer to RFC 3068, "An Anycast Prefix for 6to4 Relay Routers" (`ftp://ftp.rfc-editor.org/in-notes/rfc3068.txt`).

- Enable a tunnel to a specific 6to4 relay router.

  ```
  # /usr/sbin/6to4relay -e -a relay-router-address
  ```

  The `-a` option indicates that a specific router address is to follow. Replace *relay-router-address* with the IPv4 address of the specific 6to4 relay router with which you want to enable a tunnel.

The tunnel to the 6to4 relay router remains active until you remove the 6to4 tunnel pseudo-interface.

2 Delete the tunnel to the 6to4 relay router, when the tunnel is no longer needed:

```
# /usr/sbin/6to4relay -d
```

3 **(Optional) Make the tunnel to the 6to4 relay router persistent across reboots.**

Your site might have a compelling reason to have the tunnel to the 6to4 relay router reinstated each time the 6to4 router reboots. To support this scenario, you must do the following:

a. **Edit the `/etc/default/inetinit` file.**

The line that you need to modify is at the end of the file.

b. **Change the "NO" value in the line `ACCEPT6TO4RELAY=NO` to "YES".**

c. **(Optional) Create a tunnel to a specific 6to4 relay router that persists across reboots.**

For the parameter `RELAY6TO4ADDR`, change the address `192.88.99.1` to the IPv4 address of the 6to4 relay router that you want to use.

Example 7–5 Getting Status Information About 6to4 Relay Router Support

You can use the `/usr/bin/6to4relay` command to find out whether support for 6to4 relay routers is enabled. The next example shows the output when support for 6to4 relay routers is disabled, as is the default in Oracle Solaris:

```
# /usr/sbin/6to4relay
6to4relay: 6to4 Relay Router communication support is disabled.
```

When support for 6to4 relay routers is enabled, you receive the following output:

```
# /usr/sbin/6to4relay
6to4relay: 6to4 Relay Router communication support is enabled.
IPv4 remote address of Relay Router=192.88.99.1
```

▼ How to Modify an IP Tunnel Configuration

- **Change the tunnel's configuration.**

```
# dladm modify-iptun -a [local|remote]=addr,... tunnel-link
```

You cannot modify an existing tunnel's type. Thus, the `-T` *type* option is not allowed for this command. Only the following tunnel parameters can be modified:

`-a [local|remote]=`*address,...* Specifies literal IP addresses or host names that correspond to the local address and the remote tunnel address. Depending on the type of tunnel, you specify either only one address, or both local and remote addresses. If specifying both local and remote addresses, you must separate the addresses with a comma.

- IPv4 tunnels require local and remote IPv4 addresses to function.
- IPv6 tunnels require local and remote IPv6 addresses to function.
- 6to4 tunnels require a local IPv4 address to function.

For persistent IP tunnel data-link configurations, if you are using host names for addresses, these host names are saved in the configuration storage. During a subsequent system boot, if the names resolve to IP addresses that are different from the IP addresses used when the tunnel was created, then the tunnel acquires a new configuration.

If you are changing the tunnel's local and remote addresses, ensure that these addresses are consistent with the type of tunnel that you are modifying.

Note – If you want to change the name of the tunnel link, do not use the `modify-iptun` subcommand. Instead, use `dladm rename-link`.

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

Similarly, do not use the `modify-iptun` command to change tunnel properties such as the `hoplimit` or `encaplimit`. Instead, use the `dladm set-linkprop` command to set values for these properties.

Example 7–6 Modifying a Tunnel's Address and Properties

This example consists of two procedures. First, the local and remote addresses of the IPv4 tunnel `vpn0` are temporarily changed. When the system is later rebooted, the tunnel reverts to using the original addresses. A second procedure changes the `hoplimit` of `vpn0` to 60.

```
# dladm modify-iptun -t -a local=10.8.48.149,remote=192.1.2.3 vpn0

# dladm set-linkprop -p hoplimit=60 vpn0
```

▼ How to Display an IP Tunnel's Configuration

- **Display the IP tunnel's configuration.**

  ```
  # dladm show-iptun [-p] -o fields [tunnel-link]
  ```

  The following options can be used with the command:

`-p`	Displays the information in a machine-parseable format. This is argument is optional.
`-o` *fields*	Displays selected fields that provide specific tunnel information.
tunnel-link	Specifies the tunnel whose configuration information you want to display. This is argument is optional. If you omit the tunnel name, the command displays the information about all the tunnels on in the system.

Example 7–7 Displaying Information About All Tunnels

In this example, only one tunnel exists on the system.

```
# dladm show-iptun
LINK     TYPE     FLAGS     LOCAL          REMOTE
tun0     6to4     --        192.168.35.10  --
vpn0     ipv4     --        10.8.48.149    192.1.2.3
```

Example 7–8 Displaying Selected Fields in a Machine-Parseable Format

In this example, only specific fields with tunnel information are displayed.

```
# dladm show-iptun -p -o link,type,local
tun0:6to4:192.168.35.10
vpn0:ipv4:10.8.48.149
```

▼ How to Display an IP Tunnel's Properties

- **Display the tunnel link's properties.**

  `# dladm show-linkprop [-c] [-o` *fields***`] [`***tunnel-link***`]`**

  The following options can be used with the command:

`-c`	Displays the information in a machine-parseable format. This argument is optional.
`-o` *fields*	Displays selected fields that provide specific information about the link's properties.
tunnel-link	Specifies the tunnel whose information about properties you want to display. This argument is optional. If you omit the tunnel name, the command displays the information about all the tunnels on in the system.

Example 7–9 Displaying a Tunnel's Properties

This example shows how to display all of a tunnel's link properties.

```
# dladm show-linkprop tun0
LINK     PROPERTY     PERM     VALUE     DEFAULT     POSSIBLE
tun0     autopush     --       --        --          --
tun0     zone         rw       --        --          --
tun0     state        r-       up        up          up,down
tun0     mtu          r-       65515     --          576-65495
tun0     maxbw        rw       --        --          --
tun0     cpus         rw       --        --          --
tun0     priority     rw       high      high        low,medium,high
tun0     hoplimit     rw       64        64          1-255
```

How to Delete an IP Tunnel

1 **Use the appropriate syntax to unplumb the IP interface that is configured over the tunnel depending on the type of interface.**

```
# ipadm delete-ip tunnel-link
```

Note – To successfully delete a tunnel, no existing IP interface can be plumbed on the tunnel.

2 **Delete the IP tunnel.**

```
# dladm delete-iptun tunnel-link
```

The only option for this command is `-t`, which causes the tunnel to be deleted temporarily. When you reboot the system, the tunnel is restored.

Example 7–10 Deleting an IPv6 Tunnel That is Configured With an IPv6 Interface

In this example, a persistent tunnel is permanently deleted.

```
# ipadm delete-ip ip6.tun0
# dladm delete-iptun ip6.tun0
```

◆ ◆ ◆ CHAPTER 8

Troubleshooting Network Problems (Tasks)

This chapter contains solutions for common problems that might occur on your network. The following topics are covered:

General Network Troubleshooting Tips

One of the first signs of trouble on a network is a loss of communications by one or more hosts. If a host does not to come up at all the first time that the host is added to the network, the problem might be in one of the configuration files. The problem might also be a faulty network interface card. If a single host suddenly develops a problem, the network interface might be the cause. If the hosts on a network can communicate with each other but not with other networks, the problem could lie with the router. Or, the problem could be in another network.

You can use the `ipadm` command to obtain information on network interfaces. Use the `netstat` command to display routing tables and protocol statistics. Third-party network diagnostic programs provide a number of troubleshooting tools. Refer to third-party documentation for information.

Less obvious are the causes of problems that degrade performance on the network. For example, you can use tools such as `ping` to quantify problems such as the loss of packets by a host.

Running Basic Diagnostic Checks

If the network has problems, you can run a series of software checks to diagnose and fix basic, software-related problems.

▼ How to Perform Basic Network Software Checking

1. **Use the `netstat` command to display network information.**

   For syntax and information about the `netstat` command, refer to "Monitoring Network Status With the `netstat` Command" on page 129 and the `netstat`(1M) man page.

2. **Check the `hosts` database to ensure that the entries are correct and current.**

   For information about the `/etc/inet/hosts` database, refer to "`hosts` Database" on page 178 and the `hosts`(4) man page.

3. **If you are running the Reverse Address Resolution Protocol (RARP), check the Ethernet addresses in the `ethers` database to ensure that the entries are correct and current.**

4. **Try to connect to the local host by using the `telnet` command.**

   For syntax and information about `telnet`, refer to the `telnet`(1) man page.

5. **Ensure that the network daemon `inetd` is running.**

   ```
   # ps -ef | grep inetd
   ```

   The following output verifies that the `inetd` daemon is running:

   ```
   root 57 1 0 Apr 04 ? 3:19 /usr/sbin/inetd -s
   ```

6. **If IPv6 is enabled on your network, verify that the IPv6 daemon `in.ndpd` is running:**

   ```
   # ps -ef | grep in.ndpd
   ```

   The following output verifies that the `in.ndpd` daemon is running:

   ```
   root 123  1 0  Oct 27 ?  0:03 /usr/lib/inet/in.ndpd
   ```

Common Problems When Deploying IPv6

This section describes issues and problems that you might encounter while planning and deploying IPv6 at your site. For actual planning tasks, refer to Chapter 3, "Planning an IPv6 Network (Tasks)."

IPv4 Router Cannot Be Upgraded to IPv6

If your existing equipment cannot be upgraded, you might have to purchase IPv6-ready equipment. Check the manufacturers' documentation for any equipment-specific procedures you might have to perform to support IPv6.

Certain IPv4 routers cannot be upgraded for IPv6 support. If this situation applies to your topology, physically wire an IPv6 router next to the IPv4 router. Then, you can tunnel from the

IPv6 router over the IPv4 router. For tasks for configuring tunnels, refer to "Tunnel Configuration and Administration With the `dladm` Command" on page 159.

Problems After Upgrading Services to IPv6

You might encounter the following situations when preparing services for IPv6 support:

- Certain applications, even after they are ported to IPv6, do not turn on IPv6 support by default. You might have to configure these applications to turn on IPv6.
- A server that runs multiple services, some of which are IPv4 only, and others that are both IPv4 and IPv6, can experience problems. Some clients might need to use both types of services, which leads to confusion on the server side.

Current ISP Does Not Support IPv6

If you want to deploy IPv6 but your current ISP does not offer IPv6 addressing, consider the following alternatives to changing ISPs:

- Hire an ISP to provide a second line for IPv6 communications from your site. This solution is expensive.
- Get a *virtual ISP*. A virtual ISP provides your site with IPv6 connectivity but no link. Instead, you create a tunnel from your site, over your IPv4 ISP, to the virtual ISP.
- Use a 6to4 tunnel over your ISP to other IPv6 sites. For an address, use the registered IPv4 address of the 6to4 router as the public topology part of the IPv6 address.

Security Issues When Tunneling to a 6to4 Relay Router

By nature, a tunnel between a 6to4 router and a 6to4 relay router is insecure. Security problems, such as the following, are inherent in such a tunnel:

- Though 6to4 relay routers do encapsulate and decapsulate packets, these routers do not check the data that is contained within the packets.
- Address spoofing is a major issue on tunnels to a 6to4 relay router. For incoming traffic, the 6to4 router is unable to match the IPv4 address of the relay router with the IPv6 address of the source. Therefore, the address of the IPv6 host can easily be spoofed. The address of the 6to4 relay router can also be spoofed.
- By default, no trust mechanism exists between 6to4 routers and 6to4 relay routers. Thus, a 6to4 router cannot identify whether the 6to4 relay router is to be trusted, or even if it is a legitimate 6to4 relay router. A trust relationship between the 6to4 site and the IPv6 destination must exist, or both sites leave themselves open to possible attacks.

These problems and other security issues that are inherent with 6to4 relay routers are explained in the Internet Draft, *Security Considerations for 6to4*. Generally, you should consider enabling support for 6to4 relay routers for the following reasons only:

- Your 6to4 site intends to communicate with a private, trusted IPv6 network. For example, you might enable 6to4 relay router support on a campus network that consists of isolated 6to4 sites and native IPv6 sites.
- Your 6to4 site has a compelling business reason to communicate with certain native IPv6 hosts.
- You have implemented the checks and trust models that are suggested in the Internet Draft, *Security Considerations for 6to4*.

◆ ◆ ◆ CHAPTER 9

TCP/IP and IPv4 in Depth (Reference)

This chapter provides TCP/IP network reference information about network configuration files, including the types, their purpose, and the format of the file entries. The existing network databases are also described in detail. The chapter also shows how the structure of IPv4 addresses are derived, based on defined network classifications and subnet numbers.

This chapter contains the following information:

TCP/IP Configuration Files

Each system on the network obtains its TCP/IP configuration information from the following TCP/IP configuration files and network databases:

- `/etc/defaultdomain` file
- `/etc/defaultrouter` file (optional)
- `hosts` database
- `netmasks` database (optional)

The Oracle Solaris installation program creates these files as part of the installation process. You can also edit the files manually, as explained in this section. The `hosts` and `netmasks` databases are two of the network databases read by the name services available on Oracle Solaris networks. "Network Databases and the `nsswitch.conf` File" on page 185 describes in detail the concept of network databases. .

/etc/defaultdomain File

This file should contain one entry: the fully qualified domain name of the administrative domain to which the local host's network belongs. You can supply this name to the Oracle Solaris installation program or edit the file at a later date. For more information on network domains, refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

/etc/defaultrouter File

This file can contain an entry for each router that is directly connected to the network. The entry should be the name for the network interface that functions as a router between networks. The presence of the `/etc/defaultrouter` file indicates that the system is configured to support static routing.

hosts Database

The `hosts` database contains the IPv4 addresses and host names of systems on your network. If you use the NIS or DNS name service, or the LDAP directory service, the `hosts` database is maintained in a database that is designated for host information. For example, on a network that runs NIS, the `hosts` database is maintained in the `hostsbyname` file.

If you use local files for the name service, the `hosts` database is maintained in the `/etc/inet/hosts` file. This file contains the host names and IPv4 addresses of the primary network interface, other network interfaces that are attached to the system, and any other network addresses that the system must check for.

Note – For compatibility with BSD-based operating systems, the `/etc/hosts` file is a symbolic link to `/etc/inet/hosts`.

/etc/inet/hosts File Format

The `/etc/inet/hosts` file uses the basic syntax that follows. Refer to the `hosts(4)` man page for complete syntax information.

IPv4-address hostname [nicknames] [#comment]

IPv4-address	Contains the IPv4 address for each interface that the local host must recognize.
hostname	Contains the host name that is assigned to the system at setup, plus the host names that are assigned to additional network interfaces that the local host must recognize.

[nickname] Is an optional field that contains a nickname for the host.

[#comment] Is an optional field for a comment.

Initial `/etc/inet/hosts` File

When you run the Oracle Solaris installation program on a system, the program configures the initial `/etc/inet/hosts` file. This file contains the minimum entries that the local host requires. The entries include the loopback address, the host IPv4 address, and the host name.

For example, the Oracle Solaris installation program might create the following `/etc/inet/hosts` file for system `tenere` shown in Figure 4–1:

EXAMPLE 9–1 `/etc/inet/hosts` File for System `tenere`

```
127.0.0.1     localhost         loghost    #loopback address
192.168.200.3   tenere                        #host name
```

Loopback Address

In Example 9–1, the IPv4 address `127.0.0.1` is the *loopback address*. The loopback address is the reserved network interface that is used by the local system to allow interprocess communication. This address enables the host to send packets to itself. Every system on a TCP/IP network must use the IP address `127.0.0.1` for IPv4 loopback on the local host.

Host Name

The IPv4 address `192.168.200.1` and the name `tenere` are the address and host name of the local system. They are assigned to the system's primary network interface.

Multiple Network Interfaces

Some systems have more than one network interface, because they are either routers or multihomed hosts. Each network interface that is attached to the system requires its own IP address and associated name. During installation, you must configure the primary network interface. If a particular system has multiple interfaces at installation time, the Oracle Solaris installation program also prompts you about these additional interfaces. You can optionally configure one or more additional interfaces at this time, or manually, at a later date.

After the Oracle Solaris installation, you can configure additional interfaces for a router or multihomed host by adding interface information to the systems' `/etc/inet/hosts` file. For more information on configuring routers and multihomed hosts refer to "Configuring an IPv4 Router" on page 90 and "Configuring Multihomed Hosts" on page 97.

Example 9–2 shows the `/etc/inet/hosts` file for system `timbuktu` that is shown in Figure 4–1.

EXAMPLE 9–2 `/etc/inet/hosts` File for System `timbuktu`

```
127.0.0.1        localhost     loghost
192.168.200.70   timbuktu      #This is the local host name
192.168.201.10   timbuktu-201  #Interface to network 192.9.201
```

With these two interfaces, `timbuktu` connects networks `192.168.200` and `192.168.201` as a router.

How Name Services Affect the `hosts` Database

The NIS and DNS name services, and LDAP directory service, maintain host names and addresses on one or more servers. These servers maintain `hosts` databases that contain information for every host and router (if applicable) on the servers' network. Refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for more information about these services.

When Local Files Provide the Name Service

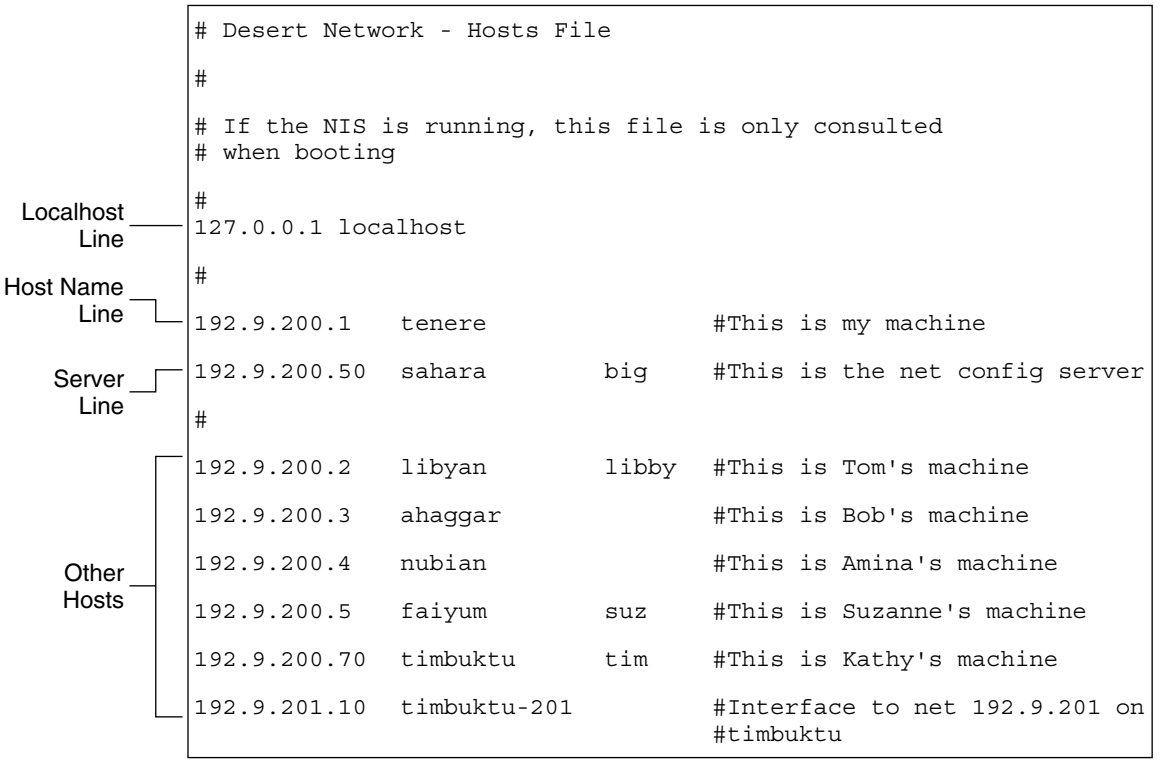
On a network that uses local files for the name service, systems that run in local files mode consult their individual `/etc/inet/hosts` files for IPv4 addresses and host names of other systems on the network. Therefore, these system's `/etc/inet/hosts` files must contain the following:

- Loopback address
- IPv4 address and host name of the local system (primary network interface)
- IPv4 address and host name of additional network interfaces that are attached to this system, if applicable
- IPv4 addresses and host names of all hosts on the local network
- IPv4 addresses and host names of any routers that this system must know about, if applicable
- IPv4 address of any system your system wants to refer to by its host name

Figure 9–1 shows the `/etc/inet/hosts` file for system `tenere`. This system runs in local files mode. Notice that the file contains the IPv4 addresses and host names for every system on the `192.9.200` network. The file also contains the IPv4 address and interface name `timbuktu-201`. This interface connects the `192.9.200` network to the `192.9.201` network.

A system that is configured as a network client uses the local `/etc/inet/hosts` file for its loopback address and IPv4 address.

FIGURE 9–1 `/etc/inet/hosts` File for a System Running in Local Files Mode



`netmasks` Database

You need to edit the `netmasks` database as part of network configuration *only* if you have set up subnetting on your network. The `netmasks` database consists of a list of networks and their associated subnet masks.

Note – When you create subnets, each new network must be a separate physical network. You cannot apply subnetting to a single physical network.

What Is Subnetting?

Subnetting is a method for maximizing the limited 32-bit IPv4 addressing space and reducing the size of the routing tables in a large internetwork. With any address class, subnetting provides a means of allocating a part of the host address space to network addresses, which lets you have more networks. The part of the host address space that is allocated to new network addresses is known as the *subnet number*.

In addition to making more efficient use of the IPv4 address space, subnetting has several administrative benefits. Routing can become very complicated as the number of networks grows. A small organization, for example, might give each local network a class C number. As the organization grows, the administration of a number of different network numbers could become complicated. A better idea is to allocate a few class B network numbers to each major division in an organization. For example, you could allocate one Class B network to Engineering, one Class B to Operations, and so on. Then, you could divide each class B network into additional networks, using the additional network numbers gained by subnetting. This division can also reduce the amount of routing information that must be communicated among routers.

Creating the Network Mask for IPv4 Addresses

As part of the subnetting process, you need to select a network-wide *netmask*. The netmask determines how many and which bits in the host address space represent the subnet number and how many and which bits represent the host number. Recall that the complete IPv4 address consists of 32 bits. Depending on the address class, as many as 24 bits and as few as 8 bits can be available for representing the host address space. The netmask is specified in the `netmasks` database.

If you plan to use subnets, you must determine your netmask before you configure TCP/IP. If you plan to install the operating system as part of network configuration, the Oracle Solaris installation program requests the netmask for your network.

As described in "Designing an IPv4 Addressing Scheme" on page 38, 32-bit IP addresses consist of a network part and a host part. The 32 bits are divided into 4 bytes. Each byte is assigned to either the network number or the host number, depending on the network class.

For example, in a class B IPv4 address, the 2 bytes on the left are assigned to the network number, and the 2 bytes on the right are assigned to the host number. In the class B IPv4 address `172.16.10`, you can assign the 2 bytes on the right to hosts.

If you are to implement subnetting, you need to use some of the bits in the bytes that are assigned to the host number to apply to subnet addresses. For example, a 16-bit host address space provides addressing for 65,534 hosts. If you apply the third byte to subnet addresses and the fourth byte to host addresses, you can address up to 254 networks, with up to 254 hosts on each network.

The bits in the host address bytes that are applied to subnet addresses and those applied to host addresses are determined by a *subnet mask*. Subnet masks are used to select bits from either byte for use as subnet addresses. Although netmask bits must be contiguous, they need not align on byte boundaries.

The netmask can be applied to an IPv4 address by using the bitwise logical AND operator. This operation selects out the network number and subnet number positions of the address.

Netmasks can be explained in terms of their binary representation. You can use a calculator for binary-to-decimal conversion. The following examples show both the decimal and binary forms of the netmask.

If a netmask `255.255.255.0` is applied to the IPv4 address `172.16.41.101`, the result is the IPv4 address of `172.16.41.0`.

`172.16.41.101` & `255.255.255.0` = `172.16.41.0`

In binary form, the operation is as follows:

10000001.10010000.00101001.01100101 (IPv4 address)

ANDed with

11111111.11111111.11111111.00000000 (netmask)

Now the system looks for a network number of `172.16.41` instead of a network number of `172.16`. If your network has the number `172.16.41`, that number is what the system checks for and finds. Because you can assign up to 254 values to the third byte of the IPv4 address space, subnetting lets you create address space for 254 networks, where previously space was available for only one.

If you are providing address space for only two additional networks, you can use the following subnet mask:

`255.255.192.0`

This netmask provides the following result:

11111111.11111111.1100000.00000000

This result still leaves 14 bits available for host addresses. Because all 0s and 1s are reserved, at least 2 bits must be reserved for the host number.

`/etc/inet/netmasks` File

If your network runs NIS or LDAP, the servers for these name services maintain `netmasks` databases. For networks that use local files for the name service, this information is maintained in the `/etc/inet/netmasks` file.

Note – For compatibility with BSD-based operating systems, the `/etc/netmasks` file is a symbolic link to `/etc/inet/netmasks`.

The following example shows the `/etc/inet/netmasks` file for a class B network.

EXAMPLE 9–3 /etc/inet/netmasks File for a Class B Network

```
# The netmasks file associates Internet Protocol (IPv4) address
# masks with IPv4 network numbers.
#
#     network-number    netmask
#
# Both the network-number and the netmasks are specified in
# “decimal dot” notation, e.g:
#
#        128.32.0.0   255.255.255.0
192.168.0.0  255.255.255.0
```

If the /etc/netmasks file does not exist, create it with a text editor. Use the following syntax:

network-number netmask-number

Refer to the netmasks(4) man page for complete details.

When creating netmask numbers, type the network number that is assigned by the ISP or Internet Registry (not the subnet number) and the netmask number in /etc/inet/netmasks. Each subnet mask should be on a separate line.

For example:

```
128.78.0.0        255.255.248.0
```

You can also type symbolic names for network numbers in the /etc/inet/hosts file. You can then use these network names instead of the network numbers as parameters to commands.

inetd Internet Services Daemon

The inetd daemon starts up Internet standard services when a system boots, and can restart a service while a system is running. Use the Service Management Facility (SMF) to modify the standard Internet services or to have additional services started by the inetd daemon.

Use the following SMF commands to manage services started by inetd:

svcadm	For administrative actions on a service, such as enabling, disabling, or restarting. For details, refer to the svcadm(1M) man page.
svcs	For querying the status of a service. For details, refer to the svcs(1) man page.
inetadm	For displaying and modifying the properties of a service. For details, refer to the inetadm(1M) man page.

The proto field value in the inetadm profile for a particular service indicates the transport layer protocol on which the service runs. If the service is IPv4-only, the proto field must be specified as tcp, udp, or sctp.

- For instructions on using the SMF commands, refer to "SMF Command-Line Administrative Utilities" in *System Administration Guide: Basic Administration*.
- For a task that uses the SMF commands to add a service that runs over SCTP, refer to "How to Add Services That Use the SCTP Protocol" on page 104.
- For information on adding services that handle both IPv4 requests and IPv6 requests, refer to "inetd Internet Services Daemon" on page 184

Network Databases and the nsswitch.conf File

The network databases are files that provide information that is needed to configure the network. The network databases follow:

- hosts
- netmasks
- ethers database
- bootparams
- protocols
- services
- networks

As part of the configuration process, you edit the hosts database and the netmasks database, if your network is subnetted. Two network databases, bootparams and ethers, are used to configure systems as network clients. The remaining databases are used by the operating system and seldom require editing.

Although nsswitch.conf file is not a network database, you need to configure this file along with the relevant network databases. nsswitch.conf specifies which name service to use for a particular system: local files, NIS, DNS, or LDAP.

How Name Services Affect Network Databases

The format of your network database depends on the type of name service you select for your network. For example, the hosts database contains, at least the host name and IPv4 address of the local system and any network interfaces that are directly connected to the local system. However, the hosts database could contain other IPv4 addresses and host names, depending on the type of name service on your network.

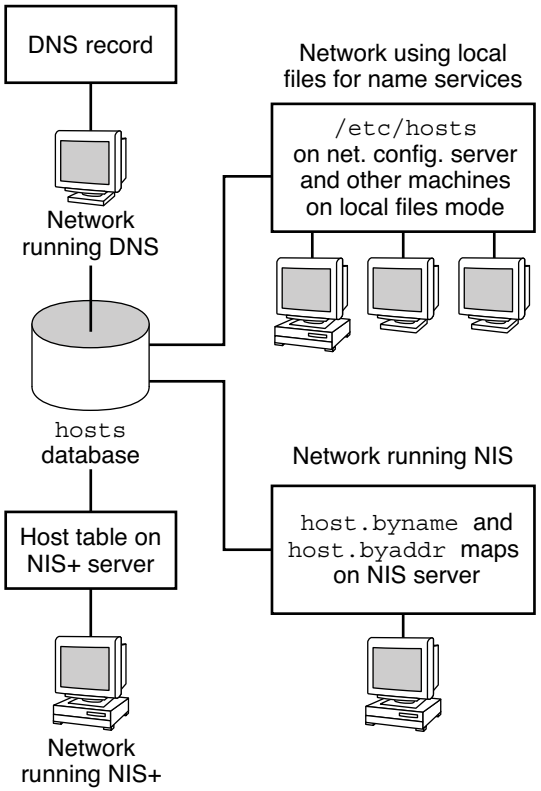
The network databases are used as follows:

- Networks that use local files for their name service rely on files in the /etc/inet and /etc directories.
- NIS uses databases that are called NIS maps.
- DNS uses records with host information.

Note – DNS boot and data files do not correspond directly to the network databases.

The following figure shows the forms of the hosts database that are used by these name services.

FIGURE 9–2 Forms of the hosts Database Used by Name Services



The following table lists the network databases and their corresponding local files and NIS maps.

Note – The ipnodes database is removed from Oracle Solaris releases.

TABLE 9–1 Network Databases and Corresponding Name Service Files

Network Database	Local Files	NIS Maps
hosts	/etc/inet/hosts	hosts.byaddr hosts.byname
ipnodes	/etc/inet/ipnodes	ipnodes.byaddr ipnodes.byname
netmasks	/etc/inet/netmasks	netmasks.byaddr
ethers	/etc/ethers	ethers.byname ethers.byaddr
bootparams	/etc/bootparams	bootparams
protocols	/etc/inet/protocols	protocols.byname protocols.bynumber
services	/etc/inet/services	services.byname
networks	/etc/inet/networks	networks.byaddr networks.byname

This book discusses network databases as they are viewed by networks that use local files for name services.

- Information about the hosts database is in "hosts Database" on page 178.
- Information about the netmasks database is in "netmasks Database" on page 181.

Refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on network databases correspondences in NIS, DNS, and LDAP.

nsswitch.conf File

The /etc/nsswitch.conf file defines the search order of the network databases. The Oracle Solaris installation program creates a default /etc/nsswitch.conf file for the local system, based on the name service you indicate during the installation process. If you selected the "None" option, indicating local files for name service, the resulting nsswitch.conf file resembles the following example.

EXAMPLE 9–4 nsswitch.conf for Networks Using Files for Name Service

```
# /etc/nsswitch.files:
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it does not use any naming service.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file contains "switch.so" as a
# nametoaddr library for "inet" transports.
```

EXAMPLE 9–4 nsswitch.conf for Networks Using Files for Name Service *(Continued)*

```
passwd:          files
group:           files
hosts:           files
networks:        files
protocols:       files
rpc:             files
ethers:          files
netmasks:        files
bootparams:      files
publickey:       files
# At present there isn't a 'files' backend for netgroup; the
# system will figure it out pretty quickly,
# and won't use netgroups at all.
netgroup:        files
automount:       files
aliases:         files
services:        files
```

The nsswitch.conf(4) man page describes the file in detail. The basic syntax is shown here:

database name-service-to-search

The *database* field can list one of many types of databases that are searched by the operating system. For example, the field could indicate a database that affects users, such as passwd or aliases, or a network database. The parameter *name-service-to-search* can have the values files or nis for the network databases. The hosts database can also have dns as a name service to search. You can also list more than one name service, such as files.

In Example 9–4, the only search option that is indicated is files. Therefore, the local system obtains security and automounting information, in addition to network database information, from files that are located in its /etc and /etc/inet directories.

Changing nsswitch.conf

The /etc directory contains the nsswitch.conf file that is created by the Oracle Solaris installation program. This directory also contains template files for the following name services:

- nsswitch.files
- nsswitch.nis

If you want to change from one name service to another name service, you can copy the appropriate template to nsswitch.conf. You can also selectively edit the nsswitch.conf file, and change the default name service to search for individual databases.

For example, on a network that runs NIS, you might have to change the nsswitch.conf file on network clients. The search path for the bootparams and ethers databases must list files as the first option, and then nis. The following example shows the correct search paths.

EXAMPLE 9–5 nsswitch.conf for a Client on a Network Running NIS

```
# /etc/nsswitch.conf:#
.
.
passwd:        files nis
group:         files nis

hosts:         files nis
networks:      files nis
protocols:     files nis
rpc:           files nis
ethers:        files nis
netmasks:      files nis
bootparams:    files nis
publickey:     files nis
netgroup:      nis

automount:     files nis
aliases:       files nis

# for efficient getservbyname() avoid nis
services:      files nis
```

For complete details on the name service switch, refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

bootparams Database

The bootparams database contains information that is used by systems that are configured to boot in network client mode. You need to edit this database if your network has network clients. See "Configuring Network Clients" on page 82 for the procedures. The database is built from information that is entered into the /etc/bootparams file.

The bootparams(4) man page contains the complete syntax for this database. Basic syntax is shown here:

system-name file-key-server-name:pathname

For each network client system, the entry might contain the following information: the name of the client, a list of keys, the names of servers, and path names. The first item of each entry is the name of the client system. All items but the first item are optional. An example follows.

EXAMPLE 9–6 bootparams Database

```
myclient   root=myserver : /nfsroot/myclient  \
swap=myserver : /nfsswap//myclient \
dump=myserver : /nfsdump/myclient
```

In this example, the term dump= tells client hosts not to look for a dump file.

Wildcard Entry for bootparams

In most instances, use the wildcard entry when editing the bootparams database to support clients. This entry follows:

```
*  root=server:/path dump=:
```

The asterisk (*) wildcard indicates that this entry applies to all clients that are not specifically named within the bootparams database.

ethers Database

The ethers database is built from information that is entered into the /etc/ethers file. This database associates host names to their *Media Access Control* (MAC) addresses. You need to create an ethers database only if you are running the RARP daemon. That is, you need to create this database if you are configuring network clients.

RARP uses the file to map MAC addresses to IP addresses. If you are running the RARP daemon in.rarpd, you need to set up the ethers file and maintain this file on all hosts that are running the daemon to reflect changes to the network.

The ethers(4) man page contains the complete syntax for this database. The basic syntax is shown here:

MAC-address hostname #comment

MAC-address	MAC address of the host
hostname	Official name of the host
#comment	Any note that you want to append to an entry in the file

The equipment manufacturer provides the MAC address. If a system does not display the MAC address during the system booting process, see your hardware manuals for assistance.

When adding entries to the ethers database, ensure that host names correspond to the primary names in the hosts not to the nicknames, as follows.

EXAMPLE 9–7 Entries in the ethers Database

```
8:0:20:1:40:16  fayoum
8:0:20:1:40:15  nubian
8:0:20:1:40:7   sahara    # This is a comment
8:0:20:1:40:14  tenere
```

Other Network Databases

The remaining network databases seldom need to be edited.

networks database

The networks database associates network names with network numbers, enabling some applications to use and display names rather than numbers. The networks database is based on information in the /etc/inet/networks file. This file contains the names of all networks to which your network connects through routers.

The Oracle Solaris installation program configures the initial networks database. However, if you add a new network to your existing network topology, you must update this database.

The networks(4) man page contains the complete syntax for /etc/inet/networks. The basic format is shown here:

network-name network-number nickname(s) #comment

network-name	Official name for the network
network-number	Number assigned by the ISP or Internet Registry
nickname	Any other name by which the network is known
#comment	Any note that you want to append to an entry in the file

You must maintain the networks file. The netstat program uses the information in this database to produce status tables.

A sample /etc/networks file follows.

EXAMPLE 9–8 /etc/networks File

```
#ident    "@(#)networks    1.4    92/07/14 SMI"    /* SVr4.0 1.1    */
#
# The networks file associates Internet Protocol (IP) network
# numbers with network names. The format of this file is:
#
#     network-name              network-number              nicnames . . .

# The loopback network is used only for intra-machine communication
loopback              127

#
# Internet networks
#
arpanet     10       arpa  # Historical
#
# local networks

eng   192.168.9 #engineering
acc   192.168.5 #accounting
prog  192.168.2 #programming
```

protocols Database

The protocols database lists the TCP/IP protocols that are installed on your system and their protocol numbers. The Oracle Solaris installation program automatically creates the database. This file seldom requires any administration.

The protocols(4) man page describes the syntax of this database. An example of the /etc/inet/protocols file follows.

EXAMPLE 9–9 /etc/inet/protocols File

```
#
# Internet (IP) protocols
#
ip    0   IP    # internet protocol, pseudo protocol number
icmp  1   ICMP  # internet control message protocol
tcp   6   TCP   # transmission control protocol
udp  17   UDP   # user datagram protocol
```

services Database

The services database lists the names of TCP and UDP services and their well-known port numbers. This database is used by programs that call network services. The Oracle Solaris installation automatically creates the services database. Generally, this database does not require any administration.

The services(4) man page contains complete syntax information. An excerpt from a typical /etc/inet/services file follows.

EXAMPLE 9–10 /etc/inet/services File

```
#
# Network services
#
echo      7/udp
echo      7/tcp
echo      7/sctp6
discard   9/udp     sink null
discard   11/tcp
daytime   13/udp
daytime   13/tcp
netstat   15/tcp
ftp-data  20/tcp
ftp       21/tcp
telnet    23/tcp
time      37/tcp    timeserver
time      37/udp    timeserver
name      42/udp    nameserver
whois     43/tcp    nickname
```

Routing Protocols in Oracle Solaris

This section describes two routing protocols supported in Oracle Solaris: Routing Information Protocol (RIP) and ICMP Router Discovery (RDISC). RIP and RDISC are both standard TCP/IP protocols. For complete lists of routing protocols available in Oracle Solaris, refer to Table 4–1 and Table 4–2.

Routing Information Protocol (RIP)

RIP is implemented by `in.routed`, the routing daemon, which automatically starts when the system boots. When run on a router with the `s` option specified, `in.routed` fills the kernel routing table with a route to every reachable network and advertises "reachability" through all network interfaces.

When run on a host with the `q` option specified, `in.routed` extracts routing information but does not advertise reachability. On hosts, routing information can be extracted in two ways:

- Do *not* specify the `S` flag (capital "S": "Space-saving mode"). `in.routed` builds a full routing table exactly as it does on a router.
- Specify the `S` flag. `in.routed` creates a minimal kernel table, containing a single default route for each available router.

ICMP Router Discovery (RDISC) Protocol

Hosts use RDISC to obtain routing information from routers. Thus, when hosts are running RDISC, routers must also run another protocol, such as RIP, in order to exchange router information.

RDISC is implemented by `in.routed`, which should run on both routers and hosts. On hosts, `in.routed` uses RDISC to discover default routes from routers that advertise themselves through RDISC. On routers, `in.routed` uses RDISC to advertise default routes to hosts on directly-connected networks. See the `in.routed(1M)` man page and the `gateways(4)` man page.

Network Classes

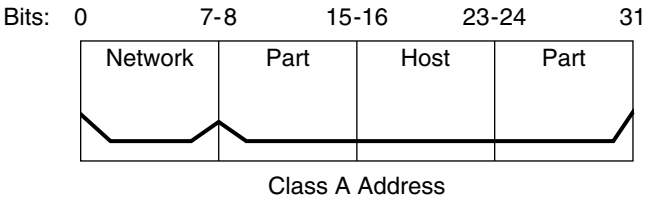
Note – Class-based network numbers are no longer available from the IANA, though many older networks are still class-based.

This section provides details about IPv4 network classes. Each class uses the 32-bit IPv4 address space differently, providing more or fewer bits for the network part of the address. These classes are class A, class B, and class C.

Class A Network Numbers

A class A network number uses the first 8 bits of the IPv4 address as its "network part." The remaining 24 bits contain the host part of the IPv4 address, as the following figure illustrates.

FIGURE 9–3 Byte Assignment in a Class A Address

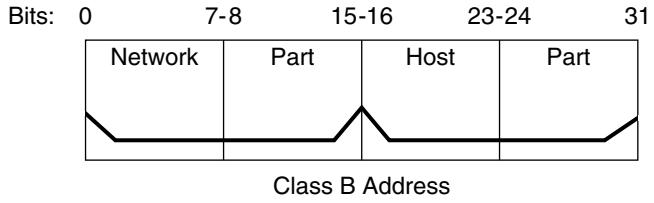


The values that are assigned to the first byte of class A network numbers fall within the range 0–127. Consider the IPv4 address `75.4.10.4`. The value 75 in the first byte indicates that the host is on a class A network. The remaining bytes, `4.10.4`, establish the host address. Only the first byte of a class A number is registered with the IANA. Use of the remaining three bytes is left to the discretion of the owner of the network number. Only 127 class A networks exist. Each one of these numbers can accommodate a maximum of 16,777,214 hosts.

Class B Network Numbers

A class B network number uses 16 bits for the network number and 16 bits for host numbers. The first byte of a class B network number is in the range 128–191. In the number `172.16.50.56`, the first two bytes, `172.16`, are registered with the IANA, and compose the network address. The last two bytes, `50.56`, contain the host address, and are assigned at the discretion of the owner of the network number. The following figure graphically illustrates a class B address.

FIGURE 9–4 Byte Assignment in a Class B Address

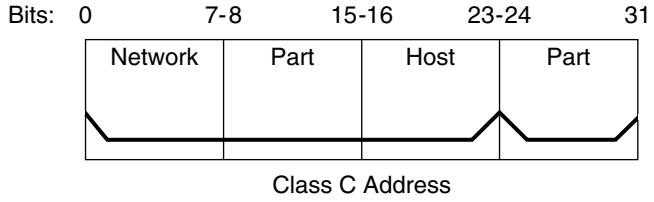


Class B is typically assigned to organizations with many hosts on their networks.

Class C Network Numbers

Class C network numbers use 24 bits for the network number and 8 bits for host numbers. Class C network numbers are appropriate for networks with few hosts—the maximum being 254. A class C network number occupies the first three bytes of an IPv4 address. Only the fourth byte is assigned at the discretion of the network owners. The following figure graphically represents the bytes in a class C address.

FIGURE 9–5 Byte Assignment in a Class C Address



The first byte of a class C network number covers the range 192–223. The second and third bytes each cover the range 1– 255. A typical class C address might be `192.168.2.5`. The first three bytes, `192.168.2`, form the network number. The final byte in this example, `5`, is the host number.

◆ ◆ ◆ CHAPTER 10

IPv6 in Depth (Reference)

This chapter contains the following reference information about Oracle Solaris IPv6 implementation.

For an overview of IPv6, refer to Chapter 2, "Planning an IPv6 Addressing Scheme (Overview)." For tasks on configuring an IPv6-enabled network, refer to Chapter 5, "Enabling IPv6 on a Network (Tasks)." For all information about IP tunnels, refer to Chapter 7, "Configuring IP Tunnels."

IPv6 Addressing Formats Beyond the Basics

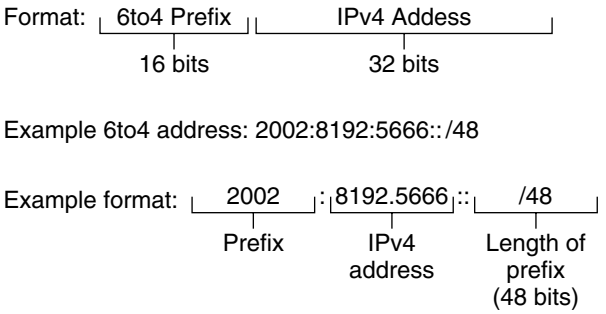
Chapter 2, "Planning an IPv6 Addressing Scheme (Overview)," introduces the most common IPv6 addressing formats: unicast site address and link-local address. This section includes in-depth explanations of addressing formats that are not covered in detail in Chapter 2, "Planning an IPv6 Addressing Scheme (Overview)":

6to4-Derived Addresses

If you plan to configure a 6to4 tunnel from a router or host endpoint, you must advertise the 6to4 site prefix in the `/etc/inet/ndpd.conf` file on the endpoint system. For an introduction and tasks for configuring 6to4 tunnels, refer to Chapter 7, "Configuring IP Tunnels."

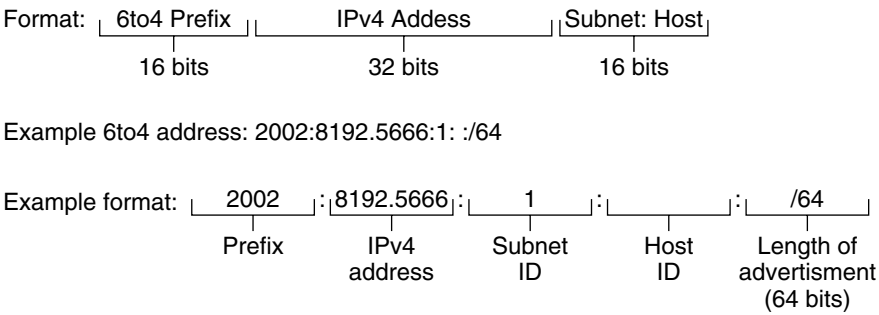
The next figure shows the parts of a 6to4 site prefix.

FIGURE 10–1 Parts of a 6to4 Site Prefix



The next figure shows the parts of a subnet prefix for a 6to4 site, such as you would include in the `ndpd.conf` file.

FIGURE 10–2 Parts of a 6to4 Subnet Prefix



This table explains the parts of a 6to4 subnet prefix, their respective lengths, and their definitions.

Part	Length	Definition
Prefix	16 bits	6to4 prefix label 2002 (0x2002).
IPv4 address	32 bits	Unique IPv4 address that is already configured on the 6to4 interface. For the advertisement, you specify the hexadecimal representation of the IPv4 address, rather than the IPv4 dotted-decimal representation.
Subnet ID	16 bits	Subnet ID, which must be a value that is unique for the link at your 6to4 site.

6to4-Derived Addressing on a Host

When an IPv6 host receives the 6to4-derived prefix by way of a router advertisement, the host automatically reconfigures a 6to4-derived address on an interface. The address has the following format:

prefix:IPv4-address:subnet-ID:interface-ID/64

The output from the `ipadm -a` command on a host with a 6to4 interface might resemble the following:

```
ADDROBJ   TYPE       STATE   ADDR
bge0/?    addrconf   ok      2002:8192:56bb:9258:a00:20ff:fea9:4521/64
```

This table explains the parts of the 6to4-derived address, their lengths and the information they provide.

Address Part	Length	Definition
prefix	16 bits	`2002`, which is the 6to4 prefix
IPv4-address	32 bits	`8192:56bb`, which is the IPv4 address, in hexadecimal notation, for the 6to4 pseudo-interface that is configured on the 6to4 router
subnet-ID	16 bits	`9258`, which is the address of the subnet of which this host is a member
interface-ID	64 bits	`a00:20ff:fea9:4521`, which is the interface ID of the host interface that is configured for 6to4

IPv6 Multicast Addresses in Depth

The IPv6 multicast address provides a method for distributing identical information or services to a defined group of interfaces, called the *multicast group*. Typically, the interfaces of the multicast group are on different nodes. An interface can belong to any number of multicast

groups. Packets sent to the multicast address go to all members of the multicast group. For example, one use of multicast addresses is for broadcasting information, similar to the capability of the IPv4 broadcast address.

The following table shows the format of the multicast address.

TABLE 10–1 IPv6 Multicast Address Format

8 bits	4 bits	4 bits	8 bits	8 bits	64 bits	32 bits
`11111111`	*FLGS*	`SCOP`	*Reserved*	*Plen*	*Network prefix*	*Group ID*

The following is a summary of the contents of each field.

- `11111111` – Identifies the address as a multicast address.
- *FLGS* – Set of the four flags 0,0,P,T. The first two flags must be zero. The P field has one of the following values:
  - 0 = Multicast address that is not assigned based on the network prefix
  - 1 = Multicast address that is assigned based on the network prefix

  If P is set to 1, then T must also be 1.
- *Reserved* - Reserved value of zero.
- *Plen* - Number of bits in the site prefix that identify the subnet, for a multicast address that is assigned based on a site prefix.
- *Group ID* - Identifier for the multicast group, either permanent or dynamic.

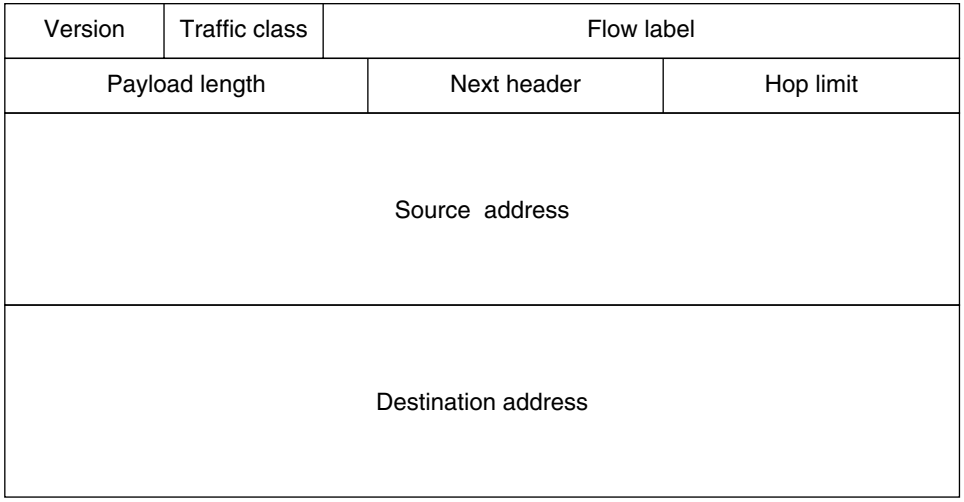
For complete details about the multicast format, refer to RFC 3306, "Unicast-Prefix-based IPv6 Multicast Addresses (`ftp://ftp.rfc-editor.org/in-notes/rfc3306.txt`).

Some IPv6 multicast addresses are permanently assigned by the Internet Assigned Numbers Authority (IANA). Some examples are the All Nodes Multicast Addresses and All Routers Multicast Addresses that are required by all IPv6 hosts and IPv6 routers. IPv6 multicast addresses can also be dynamically allocated. For more information about the proper use of multicast addresses and groups, see RFC 3307, "Allocation Guidelines for IPv6 Multicast Addresses".

IPv6 Packet Header Format

The IPv6 protocol defines a set of headers, including the basic IPv6 header and the IPv6 extension headers. The following figure shows the fields that appear in the IPv6 header and the order in which the fields appear.

FIGURE 10–3 IPv6 Basic Header Format



The following list describes the function of each header field.

- **Version** – 4-bit version number of Internet Protocol = 6.
- **Traffic class** – 8-bit traffic class field.
- **Flow label** – 20-bit field.
- **Payload length** – 16-bit unsigned integer, which is the rest of the packet that follows the IPv6 header, in octets.
- **Next header** – 8-bit selector. Identifies the type of header that immediately follows the IPv6 header. Uses the same values as the IPv4 protocol field.
- **Hop limit** – 8-bit unsigned integer. Decremented by one by each node that forwards the packet. The packet is discarded if the hop limit is decremented to zero.
- **Source address** – 128 bits. The address of the initial sender of the packet.
- **Destination address** – 128 bits. The address of the intended recipient of the packet. The intended recipient is not necessarily the recipient if an optional routing header is present.

IPv6 Extension Headers

IPv6 options are placed in separate extension headers that are located between the IPv6 header and the transport-layer header in a packet. Most IPv6 extension headers are not examined or processed by any router along a packet's delivery path until the packet arrives at its final destination. This feature provides a major improvement in router performance for packets that contain options. In IPv4, the presence of any options requires the router to examine all options.

Unlike IPv4 options, IPv6 extension headers can be of arbitrary length. Also, the number of options that a packet carries is not limited to 40 bytes. This feature, in addition to the manner in which IPv6 options are processed, permits IPv6 options to be used for functions that are not practical in IPv4.

To improve performance when handling subsequent option headers, and the transport protocol that follows, IPv6 options are always an integer multiple of 8 octets long. The integer multiple of 8 octets retains the alignment of subsequent headers.

The following IPv6 extension headers are currently defined:

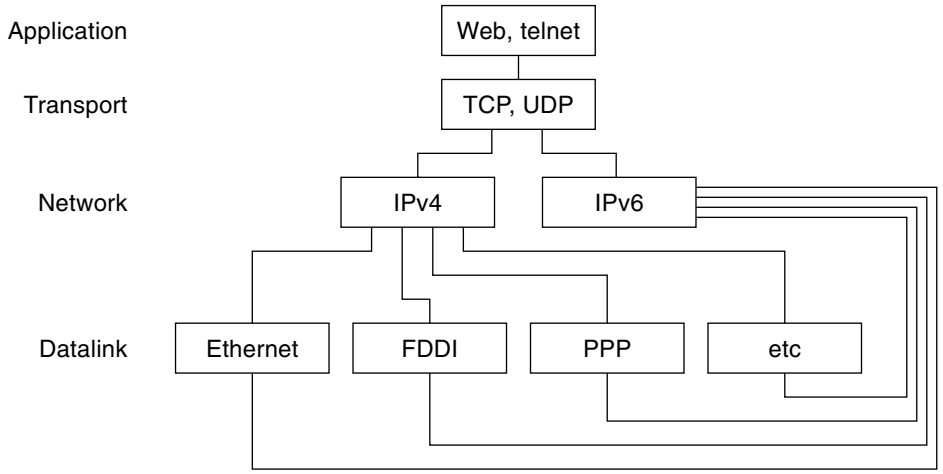
- **Routing** – Extended routing, such as IPv4 loose source route
- **Fragmentation** – Fragmentation and reassembly
- **Authentication** – Integrity and authentication, and security
- **Encapsulating Security Payload** – Confidentiality
- **Hop-by-Hop options** – Special options that require hop-by-hop processing
- **Destination options** – Optional information to be examined by the destination node

Dual-Stack Protocols

The term *dual-stack* normally refers to a complete duplication of all levels in the protocol stack from applications to the network layer. One example of complete duplication is a system that runs both the OSI and TCP/IP protocols.

Oracle Solaris is *dual-stack*, meaning that Oracle Solaris implements both IPv4 and IPv6 protocols. When you install the operating system, you can choose to enable the IPv6 protocols in the IP layer or use only the default IPv4 protocols. The remainder of the TCP/IP stack is identical. Consequently, the same transport protocols, TCP UDP and SCTP, can run over both IPv4 and IPv6. Also, the same applications can run over both IPv4 and IPv6. Figure 10–4 shows how the IPv4 and IPv6 protocols work as a dual-stack throughout the various layers of the Internet protocol suite.

FIGURE 10–4 Dual-Stack Protocol Architecture



In the dual-stack scenario, subsets of both hosts and routers are upgraded to support IPv6, in addition to IPv4. The dual-stack approach ensures that the upgraded nodes can always interoperate with IPv4-only nodes by using IPv4.

Oracle Solaris IPv6 Implementation

This section describes the files, commands, and daemons that enable IPv6 in Oracle Solaris.

IPv6 Configuration Files

This section describes the configuration files that are part of an IPv6 implementation:

- "`ndpd.conf` Configuration File" on page 203
- "`/etc/inet/ipaddrsel.conf` Configuration File" on page 207

`ndpd.conf` Configuration File

The `/etc/inet/ndpd.conf` file is used to configure options that are used by the `in.ndpd` Neighbor Discovery daemon. For a router, you primarily use `ndpd.conf` to configure the site prefix to be advertised to the link. For a host, you use `ndpd.conf` to turn off address autoconfiguration or to configure temporary addresses.

The next table shows the keywords that are used in the `ndpd.conf` file.

TABLE 10–2 `/etc/inet/ndpd.conf` Keywords

Variable	Description
`ifdefault`	Specifies the router behavior for all interfaces. Use the following syntax to set router parameters and corresponding values: `ifdefault [`*variable-value*`]`
`prefixdefault`	Specifies the default behavior for prefix advertisements. Use the following syntax to set router parameters and corresponding values: `prefixdefault [`*variable-value*`]`
`if`	Sets per-interface parameters. Use the following syntax: `if` *interface* `[`*variable-value*`]`
`prefix`	Advertises per-interface prefix information. Use the following syntax: `prefix` *prefix/length interface* `[`*variable-value*`]`

In the `ndpd.conf` file, you use the keywords in this table with a set of router configuration variables. These variables are defined in detail in RFC 2461, Neighbor Discovery for IP Version 6 (IPv6) (`http://www.ietf.org/rfc/rfc2461.txt?number=2461`).

The next table shows the variables for configuring an interface, along with brief definitions.

TABLE 10–3 `/etc/inet/ndpd.conf` Interface Configuration Variables

Variable	Default	Definition
`AdvRetransTimer`	0	Specifies the value in the Retrans Timer field in the advertisement messages sent by the router.
`AdvCurHopLimit`	Current diameter of the Internet	Specifies the value to be placed in the current hop limit in the advertisement messages sent by the router.
`AdvDefaultLifetime`	3 + `MaxRtrAdvInterval`	Specifies the default lifetime of the router advertisements.
`AdvLinkMTU`	0	Specifies a maximum transmission unit (MTU) value to be sent by the router. The zero indicates that the router does not specify MTU options.
`AdvManaged Flag`	False	Indicates the value to be placed in the Manage Address Configuration flag in the router advertisement.
`AdvOtherConfigFlag`	False	Indicates the value to be placed in the Other Stateful Configuration flag in the router advertisement.
`AdvReachableTime`	0	Specifies the value in the Reachable Time field in the advertisement messages sent by the router.

TABLE 10–3 `/etc/inet/ndpd.conf` Interface Configuration Variables *(Continued)*

Variable	Default	Definition
`AdvSendAdvertisements`	False	Indicates whether the node should send out advertisements and respond to router solicitations. You need to explicitly set this variable to "TRUE" in the `ndpd.conf` file to turn on router advertisement functions. For more information, refer to "How to Configure an IPv6-Enabled Router" on page 113.
`DupAddrDetect Transmits`	1	Defines the number of consecutive neighbor solicitation messages that the Neighbor Discovery protocol should send during duplicate address detection of the local node's address.
`MaxRtrAdvInterval`	600 seconds	Specifies the maximum time to wait between sending unsolicited multicast advertisements.
`MinRtrAdvInterval`	200 seconds	Specifies the minimum time to wait between sending unsolicited multicast advertisements.
`StatelessAddrConf`	True	Controls whether the node configures its IPv6 address through stateless address autoconfiguration. If False is declared in `ndpd.conf`, then the address must be manually configured. For more information, refer to "How to Configure a User-Specified IPv6 Token" on page 120.
`TmpAddrsEnabled`	False	Indicates whether a temporary address should be created for all interfaces or for a particular interface of a node. For more information, refer to "How to Configure a Temporary Address" on page 117.
`TmpMaxDesyncFactor`	600 seconds	Specifies a random value to be subtracted from the preferred lifetime variable `TmpPreferredLifetime` when `in.ndpd` starts. The purpose of the `TmpMaxDesyncFactor` variable is to prevent all the systems on your network from regenerating their temporary addresses at the same time. `TmpMaxDesyncFactor` allows you to change the upper bound on that random value.
`TmpPreferredLifetime`	False	Sets the preferred lifetime of a temporary address. For more information, refer to "How to Configure a Temporary Address" on page 117.
`TmpRegenAdvance`	False	Specifies the lead time in advance of address deprecation for a temporary address. For more information, refer to "How to Configure a Temporary Address" on page 117.
`TmpValidLifetime`	False	Sets the valid lifetime for a temporary address. For more information, refer to "How to Configure a Temporary Address" on page 117.

The next table shows the variables that are used for configuring IPv6 prefixes.

TABLE 10–4 `/etc/inet/ndpd.conf` Prefix Configuration Variables

Variable	Default	Definition
`AdvAutonomousFlag`	True	Specifies the value to be placed in the Autonomous Flag field in the Prefix Information option.
`AdvOnLinkFlag`	True	Specifies the value to be placed in the on-link flag (“L-bit”) in the Prefix Information option.
`AdvPreferredExpiration`	Not set	Specifies the preferred expiration date of the prefix.
`AdvPreferredLifetime`	604800 seconds	Specifies the value to be placed in the preferred lifetime in the Prefix Information option.
`AdvValidExpiration`	Not set	Specifies the valid expiration date of the prefix.
`AdvValidLifetime`	2592000 seconds	Specifies the valid lifetime of the prefix that is being configured.

EXAMPLE 10–1 `/etc/inet/ndpd.conf` File

The following example shows how the keywords and configuration variables are used in the `ndpd.conf` file. Remove the comment (#) to activate the variable.

```
# ifdefault      [variable-value ]*
# prefixdefault [variable-value ]*
# if ifname    [variable-value ]*
# prefix prefix/length ifname
#
#  Per interface configuration variables
#
#DupAddrDetectTransmits
#AdvSendAdvertisements
#MaxRtrAdvInterval
#MinRtrAdvInterval
#AdvManagedFlag
#AdvOtherConfigFlag
#AdvLinkMTU
#AdvReachableTime
#AdvRetransTimer
#AdvCurHopLimit
#AdvDefaultLifetime
#
# Per Prefix:  AdvPrefixList configuration variables
#
#
#AdvValidLifetime
#AdvOnLinkFlag
#AdvPreferredLifetime
#AdvAutonomousFlag
#AdvValidExpiration
#AdvPreferredExpiration

ifdefault AdvReachableTime 30000 AdvRetransTimer 2000
prefixdefault AdvValidLifetime 240m AdvPreferredLifetime 120m
```

EXAMPLE 10–1 `/etc/inet/ndpd.conf` File *(Continued)*

```
if qe0 AdvSendAdvertisements 1
prefix 2:0:0:56::/64 qe0
prefix fec0:0:0:56::/64 qe0

if qe1 AdvSendAdvertisements 1
prefix 2:0:0:55::/64 qe1
prefix fec0:0:0:56::/64 qe1

if hme1 AdvSendAdvertisements 1
prefix  2002:8192:56bb:1::/64 qfe0

if hme1 AdvSendAdvertisements 1
prefix  2002:8192:56bb:2::/64 hme1
```

`/etc/inet/ipaddrsel.conf` Configuration File

The `/etc/inet/ipaddrsel.conf` file contains the IPv6 default address selection policy table. When you install Oracle Solaris with IPv6 enabled, this file contains the contents that are shown in Table 10–5.

You can edit the contents of `/etc/inet/ipaddrsel.conf`. However, in most cases, you should refrain from modifying this file. If modification is necessary, refer to the procedure "How to Administer the IPv6 Address Selection Policy Table" on page 147. For more information on `ippaddrsel.conf`, refer to "Reasons for Modifying the IPv6 Address Selection Policy Table" on page 208 and the `ipaddrsel.conf(4)` man page.

IPv6-Related Commands

This section describes commands that are added with the Oracle Solaris IPv6 implementation. The text also describes modifications to existing commands to support IPv6.

`ipaddrsel` Command

The `ipaddrsel` command enables you to modify the IPv6 default address selection policy table.

The Oracle Solaris kernel uses the IPv6 default address selection policy table to perform destination address ordering and source address selection for an IPv6 packet header. The `/etc/inet/ipaddrsel.conf` file contains the policy table.

The following table lists the default address formats and their priorities for the policy table. You can find technical details for IPv6 address selection in the `inet6(7P)` man page.

TABLE 10–5 IPv6 Address Selection Policy Table

Prefix	Precedence	Definition
`::1/128`	50	Loopback
`::/0`	40	Default
`2002::/16`	30	6to4
`::/96`	20	IPv4 Compatible
`::ffff:0:0/96`	10	IPv4

In this table, IPv6 prefixes (`::1/128` and `::/0`) take precedence over 6to4 addresses (`2002::/16`) and IPv4 addresses (`::/96` and `::ffff:0:0/96`). Therefore, by default, the kernel selects the global IPv6 address of the interface for packets going to another IPv6 destination. The IPv4 address of the interface has a lower priority, particularly for packets going to an IPv6 destination. Given the selected IPv6 source address, the kernel also uses the IPv6 format for the destination address.

Reasons for Modifying the IPv6 Address Selection Policy Table

Under most instances, you do not need to change the IPv6 default address selection policy table. If you do need to administer the policy table, you use the `ipaddrsel` command.

You might want to modify the policy table under the following circumstances:

- If the system has an interface that is used for a 6to4 tunnel, you can give higher priority to 6to4 addresses.
- If you want a particular source address to be used only in communications with a particular destination address, you can add these addresses to the policy table. Then, you can use `ipadm` to flag these addresses as preferred. For more information about the `ipadm` command, refer to the `ipadm(1M)` man page.
- If you want IPv4 addresses to take precedence over IPv6 addresses, you can change the priority of `::ffff:0:0/96` to a higher number.
- If you need to assign a higher priority to deprecated addresses, you can add the deprecated address to the policy table. For example, site-local addresses are now deprecated in IPv6. These addresses have the prefix `fec0::/10`. You can change the policy table to give higher priority to site-local addresses.

For details about the `ipaddrsel` command, refer to the `ipaddrsel(1M)` man page.

`6to4relay` Command

6to4 tunneling enables communication between isolated 6to4 sites. However, to transfer packets with a native, non-6to4 IPv6 site, the 6to4 router must establish a tunnel with a 6to4 relay

router. The *6to4 relay router* then forwards the 6to4 packets to the IPv6 network and ultimately, to the native IPv6 site. If your 6to4-enabled site must exchange data with a native IPv6 site, you use the `6to4relay` command to enable the appropriate tunnel.

Because the use of relay routers is insecure, tunneling to a relay router is disabled by default in Oracle Solaris. Carefully consider the issues that are involved in creating a tunnel to a 6to4 relay router before deploying this scenario. For detailed information on 6to4 relay routers, refer to "Considerations for Tunnels to a 6to4 Relay Router" on page 156. If you decide to enable 6to4 relay router support, you can find the related procedures in "How to Create and Configure an IP Tunnel" on page 161.

Syntax of 6to4relay

The `6to4relay` command has the following syntax:

```
6to4relay -e [-a IPv4-address] -d -h
```

`-e`	Enables support for tunnels between the 6to4 router and an anycast 6to4 relay router. The tunnel endpoint address is then set to `192.88.99.1`, the default address for the anycast group of 6to4 relay routers.
`-a` *IPv4-address*	Enables support for tunnels between the 6to4 router and a 6to4 relay router with the specified *IPv4-address*.
`-d`	Disables support for tunneling to the 6to4 relay router, the default for Oracle Solaris.
`-h`	Displays help for `6to4relay`.

For more information, refer to the `6to4relay`(1M) man page.

EXAMPLE 10–2 Default Status Display of 6to4 Relay Router Support

The `6to4relay` command, without arguments, shows the current status of 6to4 relay router support. This example shows the default for the Oracle Solaris implementation of IPv6.

```
# /usr/sbin/6to4relay
6to4relay:6to4 Relay Router communication support is disabled
```

EXAMPLE 10–3 Status Display With 6to4 Relay Router Support Enabled

If relay router support is enabled, `6to4relay` displays the following output:

```
# /usr/sbin/6to4relay
6to4relay:6to4 Relay Router communication support is enabled
IPv4 destination address of Relay Router=192.88.99.1
```

EXAMPLE 10–4 Status Display With a 6to4 Relay Router Specified

If you specify the `-a` option and an IPv4 address to the `6to4relay` command, the IPv4 address that you give with `-a` is displayed instead of `192.88.99.1`.

`6to4relay` does not report successful execution of the `-d`, `-e`, and `-a` *IPv4 address* options. However, `6to4relay` does display any error messages that might be generated when you run these options.

`netstat` Command Modifications for IPv6 Support

The `netstat` command displays both IPv4 and IPv6 network status. You can choose which protocol information to display by setting the `DEFAULT_IP` value in the `/etc/default/inet_type` file or by using the `-f` command-line option. With a permanent setting of `DEFAULT_IP`, you can ensure that `netstat` displays only IPv4 information. You can override this setting by using the `-f` option. For more information on the `inet_type` file, see the `inet_type(4)` man page.

The `-p` option of the `netstat` command displays the net-to-media table, which is the ARP table for IPv4 and the neighbor cache for IPv6. See the `netstat(1M)` man page for details. See "How to Display the Status of Sockets" on page 132 for descriptions of procedures that use this command.

`snoop` Command Modifications for IPv6 Support

The `snoop` command can capture both IPv4 and IPv6 packets. This command can display IPv6 headers, IPv6 extension headers, ICMPv6 headers, and Neighbor Discovery protocol data. By default, the `snoop` command displays both IPv4 and IPv6 packets. If you specify the `ip` or `ip6` protocol keyword, the `snoop` command displays only IPv4 or IPv6 packets. The IPv6 filter option enables you to filter through all packets, both IPv4 and IPv6, displaying only the IPv6 packets. See the `snoop(1M)` man page for details. See "How to Monitor IPv6 Network Traffic" on page 143 for procedures that use the `snoop` command.

`route` Command Modifications for IPv6 Support

The `route` command operates on both IPv4 and IPv6 routes, with IPv4 routes as the default. If you use the `-inet6` option on the command line immediately after the `route` command, operations are performed on IPv6 routes. See the `route(1M)` man page for details.

`ping` Command Modifications for IPv6 Support

The `ping` command can use both IPv4 and IPv6 protocols to probe target hosts. Protocol selection depends on the addresses that are returned by the name server for the specific target host. By default, if the name server returns an IPv6 address for the target host, the `ping`

command uses the IPv6 protocol. If the server returns only an IPv4 address, the `ping` command uses the IPv4 protocol. You can override this action by using the `-A` command-line option to specify which protocol to use.

For detailed information, see the `ping`(1M) man page. For procedures that use `ping`, refer to "Probing Remote Hosts With the `ping` Command" on page 135.

`traceroute` Command Modifications for IPv6 Support

You can use the `traceroute` command to trace both the IPv4 and IPv6 routes to a specific host. From a protocol perspective, `traceroute` uses the same algorithm as `ping`. Use the `-A` command-line option to override this selection. You can trace each individual route to every address of a multihomed host by using the `-a` command-line option.

For detailed information, see the `traceroute`(1M) man page. For procedures that use `traceroute`, refer to "Displaying Routing Information With the `traceroute` Command" on page 139.

IPv6-Related Daemons

This section discusses the IPv6-related daemons.

`in.ndpd` Daemon, for Neighbor Discovery

The`in.ndpd` daemon implements the IPv6 Neighbor Discovery protocol and router discovery. The daemon also implements address autoconfiguration for IPv6. The following shows the supported options of `in.ndpd`.

`-d` Turns on debugging.

`-D` Turns on debugging for specific events.

`-f` Specifies a file to read configuration data from, instead of the default `/etc/inet/ndpd.conf` file.

`-I` Prints related information for each interface.

`-n` Does not loop back router advertisements.

`-r` Ignores received packets.

`-v` Specifies verbose mode, reporting various types of diagnostic messages.

`-t` Turns on packet tracing.

The `in.ndpd` daemon is controlled by parameters that are set in the `/etc/inet/ndpd.conf` configuration file and any applicable parameters in the `/var/inet/ndpd_state.`*interface* startup file.

When the `/etc/inet/ndpd.conf` file exists, the file is parsed and used to configure a node as a router. Table 10–2 lists the valid keywords that might appear in this file. When a host is booted, routers might not be immediately available. Advertised packets by the router might be dropped. Also, advertised packets might not reach the host.

The `/var/inet/ndpd_state.`*interface* file is a state file. This file is updated periodically by each node. When the node fails and is restarted, the node can configure its interfaces in the absence of routers. This file contains the interface address, the last time that the file was updated, and how long the file is valid. This file also contains other parameters that are "learned" from previous router advertisements.

Note – You do not need to alter the contents of state files. The `in.ndpd` daemon automatically maintains state files.

See the `in.ndpd(1M)` man page and the `ndpd.conf(4)` man page for lists of configuration variables and allowable values.

`in.ripngd` Daemon, for IPv6 Routing

The `in.ripngd` daemon implements the Routing Information Protocol next-generation for IPv6 routers (RIPng). RIPng defines the IPv6 equivalent of RIP. When you configure an IPv6 router with the `routeadm` command and turn on IPv6 routing, the `in.ripngd` daemon implements RIPng on the router.

The following shows the supported options of RIPng.

`-p` *n* — *n* specifies the alternate port number that is used to send or receive RIPnG packets.

`-q` — Suppresses routing information.

`-s` — Forces routing information even if the daemon is acting as a router.

`-P` — Suppresses use of poison reverse.

`-S` — If `in.ripngd` does not act as a router, the daemon enters only a default route for each router.

`inetd` Daemon and IPv6 Services

An IPv6-enabled server application can handle both IPv4 requests and IPv6 requests, or IPv6 requests only. The server always handles requests through an IPv6 socket. Additionally, the

server uses the same protocol that the corresponding client uses. To add or modify a service for IPv6, use the commands available from the Service Management Facility (SMF).

- For information about the SMF commands, refer to "SMF Command-Line Administrative Utilities" in *System Administration Guide: Basic Administration*.
- For an example task that uses SMF to configure an IPv4 service manifest that runs over SCTP, refer to "How to Add Services That Use the SCTP Protocol" on page 104.

To configure an IPv6 service, you must ensure that the `proto` field value in the `inetadm` profile for that service lists the appropriate value:

- For a service that handles both IPv4 and IPv6 requests, choose `tcp6`, `udp6`, or `sctp`. A `proto` value of `tcp6`, `udp6`, or `sctp6` causes `inetd` to pass on an IPv6 socket to the server. The server contains an IPv4-mapped address in case a IPv4 client has a request.
- For a service that handles only IPv6 requests, choose `tcp6only` or `udp6only`. With either of these values for `proto`, `inetd` passes the server an IPv6 socket.

If you replace an Oracle Solaris command with another implementation, you must verify that the implementation of that service supports IPv6. If the implementation does not support IPv6, then you must specify the `proto` value as either `tcp`, `udp`, or `sctp`.

Here is a profile that results from running `inetadm` for an `echo` service manifest that supports both IPv4 and IPv6 and runs over SCTP:

```
# inetadm -l svc:/network/echo:sctp_stream
    SCOPE    NAME=VALUE      name="echo"
             endpoint_type="stream"
             proto="sctp6"
             isrpc=FALSE
             wait=FALSE
             exec="/usr/lib/inet/in.echod -s"
             user="root"
    default  bind_addr=""
    default  bind_fail_max=-1
    default  bind_fail_interval=-1
    default  max_con_rate=-1
    default  max_copies=-1
    default  con_rate_offline=-1
    default  failrate_cnt=40
    default  failrate_interval=60
    default  inherit_env=TRUE
    default  tcp_trace=FALSE
    default  tcp_wrappers=FALSE
```

To change the value of the `proto` field, use the following syntax:

```
# inetadm -m FMRI proto="transport-protocols"
```

All servers that are provided with Oracle Solaris software require only one profile entry that specifies `proto` as `tcp6`, `udp6`, or `sctp6`. However, the remote shell server (`shell`) and the

remote execution server (`exec`) now are composed of a single service instance, which requires a `proto` value containing both the `tcp` and `tcp6only` values. For example, to set the `proto` value for `shell`, you would issue the following command:

```
# inetadm -m network/shell:default proto="tcp,tcp6only"
```

See IPv6 extensions to the Socket API in *Programming Interfaces Guide* for more details on writing IPv6-enabled servers that use sockets.

Considerations When Configuring a Service for IPv6

When you add or modify a service for IPv6, keep in mind the following caveats:

- You need to specify the `proto` value as `tcp6`, `sctp6`, or `udp6` to enable both IPv4 or IPv6 connections. If you specify the value for `proto` as `tcp`, `sctp`, or `udp`, the service uses only IPv4.
- Though you can add a service instance that uses one-to-many style SCTP sockets for `inetd`, this is not recommended. `inetd` does not work with one-to-many style SCTP sockets.
- If a service requires two entries because its `wait-status` or `exec` properties differ, then you must create two instances/services from the original service.

IPv6 Neighbor Discovery Protocol

IPv6 introduces the Neighbor Discovery protocol, as described in RFC 2461, Neighbor Discovery for IP Version 6 (IPv6) (`http://www.ietf.org/rfc/rfc2461.txt?number=2461`). For an overview of major Neighbor Discovery features, refer to "IPv6 Neighbor Discovery Protocol Overview" on page 59.

This section discusses the following features of the Neighbor Discovery protocol:

ICMP Messages From Neighbor Discovery

Neighbor Discovery defines five new Internet Control Message Protocol (ICMP) messages. The messages serve the following purposes:

- **Router solicitation** – When an interface becomes enabled, hosts can send router solicitation messages. The solicitations request routers to generate router advertisements immediately, rather than at their next scheduled time.
- **Router advertisement** – Routers advertise their presence, various link parameters, and various Internet parameters. Routers advertise either periodically, or in response to a router solicitation message. Router advertisements contain prefixes that are used for on-link determination or address configuration, a suggested hop-limit value, and so on.
- **Neighbor solicitation** – Nodes send neighbor solicitation messages to determine the link-layer address of a neighbor. Neighbor solicitation messages are also sent to verify that a neighbor is still reachable by a cached link-layer address. Neighbor solicitations are also used for duplicate address detection.
- **Neighbor advertisement** – A node sends neighbor advertisement messages in response to a neighbor solicitation message. The node can also send unsolicited neighbor advertisements to announce a link-layer address change.
- **Redirect** – Routers use redirect messages to inform hosts of a better first hop for a destination, or that the destination is on the same link.

Autoconfiguration Process

This section provides an overview of the typical steps that are performed by an interface during autoconfiguration. Autoconfiguration is performed only on multicast-capable links.

1. A multicast-capable interface is enabled, for example, during system startup of a node.
2. The node begins the autoconfiguration process by generating a link-local address for the interface.

   The link-local address is formed from the Media Access Control (MAC) address of the interface.
3. The node sends a neighbor solicitation message that contains the tentative link-local address as the target.

   The purpose of the message is to verify that the prospective address is not already in use by another node on the link. After verification, the link-local address can be assigned to an interface.

   a. If another node already uses the proposed address, that node returns a neighbor advertisement stating that the address is already in use.

   b. If another node is also attempting to use the same address, the node also sends a neighbor solicitation for the target.

      The number of neighbor solicitation transmissions or retransmissions, and the delay between consecutive solicitations, are link specific. You can set these parameters, if necessary.

4. If a node determines that its prospective link-local address is not unique, autoconfiguration stops. At that point, you must manually configure the link-local address of the interface.

   To simplify recovery, you can supply an alternate interface ID that overrides the default identifier. Then, the autoconfiguration mechanism can resume by using the new, presumably unique, interface ID.

5. When a node determines that its prospective link-local address is unique, the node assigns the address to the interface.

   At this point, the node has IP-level connectivity with neighboring nodes. The remaining autoconfiguration steps are performed only by hosts.

Obtaining a Router Advertisement

The next phase of autoconfiguration involves obtaining a router advertisement or determining that no routers are present. If routers are present, the routers send router advertisements that specify what type of autoconfiguration a host should perform.

Routers send router advertisements periodically. However, the delay between successive advertisements is generally longer than a host that performs autoconfiguration can wait. To quickly obtain an advertisement, a host sends one or more router solicitations to the all-routers multicast group.

Prefix Configuration Variables

Router advertisements also contain prefix variables with information that stateless address autoconfiguration uses to generate prefixes. The Stateless Address Autoconfiguration field in router advertisements are processed independently. One option field that contains prefix information, the Address Autoconfiguration flag, indicates whether the option even applies to stateless autoconfiguration. If the option field does apply, additional option fields contain a subnet prefix with lifetime values. These values indicate the length of time that addresses created from the prefix remain preferred and valid.

Because routers periodically generate router advertisements, hosts continually receive new advertisements. IPv6-enabled hosts process the information that is contained in each advertisement. Hosts add to the information. They also refresh the information that is received in previous advertisements.

Address Uniqueness

For security reasons, all addresses must be tested for uniqueness prior to their assignment to an interface. The situation is different for addresses that are created through stateless autoconfiguration. The uniqueness of an address is determined primarily by the portion of the address that is formed from an interface ID. Thus, if a node has already verified the uniqueness of a link-local address, additional addresses need not be tested individually. The addresses must

be created from the same interface ID. In contrast, all addresses that are obtained manually should be tested individually for uniqueness. System administrators at some sites believe that the overhead of performing duplicate address detection outweighs its benefits. For these sites, the use of duplicate address detection can be disabled by setting a per-interface configuration flag.

To accelerate the autoconfiguration process, a host can generate its link-local address, and verify its uniqueness, while the host waits for a router advertisement. A router might delay a response to a router solicitation for a few seconds. Consequently, the total time necessary to complete autoconfiguration can be significantly longer if the two steps are done serially.

Neighbor Solicitation and Unreachability

Neighbor Discovery uses *neighbor solicitation* messages to determine if more than one node is assigned the same unicast address. *Neighbor unreachability detection* detects the failure of a neighbor or the failure of the forward path to the neighbor. This detection requires positive confirmation that packets that are sent to a neighbor are actually reaching that neighbor. Neighbor unreachability detection also determines that packets are being processed properly by the node's IP layer.

Neighbor unreachability detection uses confirmation from two sources: upper-layer protocols and neighbor solicitation messages. When possible, upper-layer protocols provide a positive confirmation that a connection is making *forward progress*. For example, when new TCP acknowledgments are received, it is confirmed that previously sent data has been delivered correctly.

When a node does not get positive confirmation from upper-layer protocols, the node sends unicast neighbor solicitation messages. These messages solicit neighbor advertisements as reachability confirmation from the next hop. To reduce unnecessary network traffic, probe messages are sent only to neighbors to which the node is actively sending packets.

Duplicate Address Detection Algorithm

To ensure that all configured addresses are likely to be unique on a particular link, nodes run a *duplicate address detection* algorithm on addresses. The nodes must run the algorithm before assigning the addresses to an interface. The duplicate address detection algorithm is performed on all addresses.

The autoconfiguration process that is described in this section applies only to hosts, and not routers. Because host autoconfiguration uses information that is advertised by routers, routers need to be configured by some other means. However, routers generate link-local addresses by

using the mechanism that is described in this chapter. In addition, routers are expected to successfully pass the duplicate address detection algorithm on all addresses prior to assigning the address to an interface.

Proxy Advertisements

A router that accepts packets on behalf of a target address can issue non-override neighbor advertisements. The router can accept packets for a target address that is unable to respond to neighbor solicitations. Currently, the use of proxy is not specified. However, proxy advertising can potentially be used to handle cases such as mobile nodes that have moved off-link. Note that the use of proxy is not intended as a general mechanism to handle nodes that do not implement this protocol.

Inbound Load Balancing

Nodes with replicated interfaces might need to load balance the reception of incoming packets across multiple network interfaces on the same link. Such nodes have multiple link-local addresses assigned to the same interface. For example, a single network driver can represent multiple network interface cards as a single logical interface that has multiple link-local addresses.

Load balancing is handled by allowing routers to omit the source link-local address from router advertisement packets. Consequently, neighbors must use neighbor solicitation messages to learn link-local addresses of routers. Returned neighbor advertisement messages can then contain link-local addresses that differ, depending on which issued the solicitation.

Link-Local Address Change

A node that knows its link-local address has been changed can send out multicast unsolicited, neighbor advertisement packets. The node can send multicast packets to all nodes to update cached link-local addresses that have become invalid. The sending of unsolicited advertisements is a performance enhancement only. The detection algorithm for neighbor unreachability ensures that all nodes reliably discover the new address, though the delay might be somewhat longer.

Comparison of Neighbor Discovery to ARP and Related IPv4 Protocols

The functionality of the IPv6 Neighbor Discovery protocol corresponds to a combination of the IPv4 protocols: Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP) Router Discovery, and ICMP Redirect. IPv4 does not have a generally agreed on protocol or mechanism for neighbor unreachability detection. However, host requirements do specify some possible algorithms for dead gateway detection. Dead gateway detection is a subset of the problems that neighbor unreachability detection solves.

The following list compares the Neighbor Discovery protocol to the related set of IPv4 protocols.

- Router discovery is part of the base IPv6 protocol set. IPv6 hosts do not need to `snoop` the routing protocols to find a router. IPv4 uses ARP, ICMP router discovery, and ICMP redirect for router discovery.
- IPv6 router advertisements carry link-local addresses. No additional packet exchange is needed to resolve the router's link-local address.
- Router advertisements carry site prefixes for a link. A separate mechanism is not needed to configure the netmask, as is the case with IPv4.
- Router advertisements enable address autoconfiguration. Autoconfiguration is not implemented in IPv4.
- Neighbor Discovery enables IPv6 routers to advertise an MTU for hosts to use on the link. Consequently, all nodes use the same MTU value on links that lack a well-defined MTU. IPv4 hosts on the same network might have different MTUs.
- Unlike IPv4 broadcast addresses, IPv6 address resolution multicasts are spread over 4 billion (2^32) multicast addresses, greatly reducing address resolution-related interrupts on nodes other than the target. Moreover, non-IPv6 machines should not be interrupted at all.
- IPv6 redirects contain the link-local address of the new first hop. Separate address resolution is not needed on receiving a redirect.
- Multiple site prefixes can be associated with the same IPv6 network. By default, hosts learn all local site prefixes from router advertisements. However, routers can be configured to omit some or all prefixes from router advertisements. In such instances, hosts assume that destinations are on remote networks. Consequently, hosts send the traffic to routers. A router can then issue redirects, as appropriate.
- Unlike IPv4, the recipient of an IPv6 redirect message assumes that the new next-hop is on the local network. In IPv4, a host ignores redirect messages that specify a next-hop that is not on the local network, according to the network mask. The IPv6 redirect mechanism is analogous to the XRedirect facility in IPv4. The redirect mechanism is useful on non-broadcast and shared media links. On these networks, nodes should not check for all prefixes for local link destinations.

- IPv6 neighbor unreachability detection improves packet delivery in the presence of failing routers. This capability improves packet delivery over partially failing or partitioned links. This capability also improves packet delivery over nodes that change their link-local addresses. For example, mobile nodes can move off the local network without losing any connectivity because of stale ARP caches. IPv4 has no corresponding method for neighbor unreachability detection.
- Unlike ARP, Neighbor Discovery detects half-link failures by using neighbor unreachability detection. Neighbor Discovery avoids sending traffic to neighbors when two-way connectivity is absent.
- By using link-local addresses to uniquely identify routers, IPv6 hosts can maintain the router associations. The ability to identify routers is required for router advertisements and for redirect messages. Hosts need to maintain router associations if the site uses new global prefixes. IPv4 does not have a comparable method for identifying routers.
- Because Neighbor Discovery messages have a hop limit of 255 upon receipt, the protocol is immune to spoofing attacks originating from off-link nodes. In contrast, IPv4 off-link nodes can send ICMP redirect messages. IPv4 off-link nodes can also send router advertisement messages.
- By placing address resolution at the ICMP layer, Neighbor Discovery becomes more media independent than ARP. Consequently, standard IP authentication and security mechanisms can be used.

IPv6 Routing

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

IPv6 also includes simple routing extensions that support powerful new routing capabilities. The following list describes the new routing capabilities:

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

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

To make address sequences a general function, IPv6 hosts are required, in most instances, to reverse routes in a packet that a host receives. The packet must be successfully authenticated by

using the IPv6 authentication header. The packet must contain address sequences in order to return the packet to its originator. This technique forces IPv6 host implementations to support the handling and reversal of source routes. The handling and reversal of source routes is the key that enables providers to work with hosts that implement the new IPv6 capabilities such as provider selection and extended addresses.

Router Advertisement

On multicast-capable links and point-to-point links, each router periodically sends to the multicast group a router advertisement packet that announces its availability. A host receives router advertisements from all routers, building a list of default routers. Routers generate router advertisements frequently enough so that hosts learn of their presence within a few minutes. However, routers do not advertise frequently enough to rely on an absence of advertisements to detect router failure. A separate detection algorithm that determines neighbor unreachability provides failure detection.

Router Advertisement Prefixes

Router advertisements contain a list of subnet prefixes that is used to determine if a host is on the same link (on-link) as the router. The list of prefixes is also used for autonomous address configuration. Flags that are associated with the prefixes specify the intended uses of a particular prefix. Hosts use the advertised on-link prefixes to build and maintain a list that is used to decide when a packet's destination is on-link or beyond a router. A destination can be on-link even though the destination is not covered by any advertised on-link prefix. In such instances, a router can send a redirect. The redirect informs the sender that the destination is a neighbor.

Router advertisements, and per-prefix flags, enable routers to inform hosts how to perform stateless address autoconfiguration.

Router Advertisement Messages

Router advertisement messages also contain Internet parameters, such as the hop limit, that hosts should use in outgoing packets. Optionally, router advertisement messages also contain link parameters, such as the link MTU. This feature enables the centralized administration of critical parameters. The parameters can be set on routers and automatically propagated to all hosts that are attached.

Nodes accomplish address resolution by sending to the multicast group a neighbor solicitation that asks the target node to return its link-layer address. Multicast neighbor solicitation messages are sent to the solicited-node multicast address of the target address. The target returns its link-layer address in a unicast neighbor advertisement message. A single

request-response pair of packets is sufficient for both the initiator and the target to resolve each other's link-layer addresses. The initiator includes its link-layer address in the neighbor solicitation.

IPv6 Extensions to Oracle Solaris Name Services

This section describes naming changes that were introduced by the implementation of IPv6. You can store IPv6 addresses in any of the Oracle Solaris naming services, NIS, LDAP, DNS, and files. You can also use NIS over IPv6 RPC transports to retrieve any NIS data.

DNS Extensions for IPv6

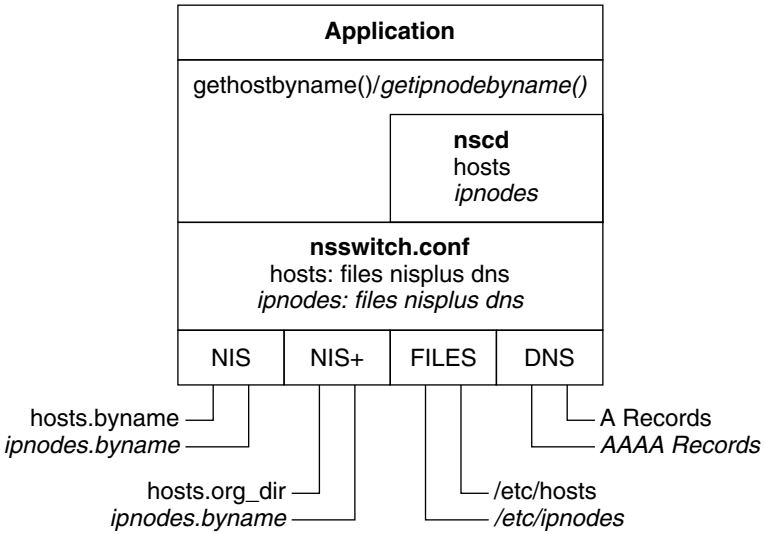
An IPv6-specific resource record, the AAAA resource record, has been specified by in RFC 1886 *DNS Extensions to Support IP Version 6*. This AAAA record maps a host name into a 128 bit IPv6 address. The PTR record is still used with IPv6 to map IP addresses into host names. The 32 four bit nibbles of the 128 bit address are reversed for an IPv6 address. Each nibble is converted to its corresponding hexadecimal ASCII value. Then, `ip6.int` is appended.

Changes to the `nsswitch.conf` File

IPv6 support has been added to the NIS, LDAP, and DNS name services. Consequently, the `nsswitch.conf` file has been modified to support IPv6 lookups.

The following diagram shows the new relationship between the `nsswitch.conf` file and the new name services databases for applications that use the `gethostbyname` and `getipnodebyname` commands. Items in italics are new. The `gethostbyname` command checks only for IPv4 addresses that are stored in `/etc/inet/hosts`. If the lookup fails, then the command checks the database that is specified in the `hosts` entry in the `nsswitch.conf` file.

FIGURE 10–5 Relationship Between `nsswitch.conf` and Name Services



For more information on name services, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

Changes to Name Service Commands

To support IPv6, you can look up IPv6 addresses with the existing name service commands. For example, the `ypmatch` command works with the new NIS maps. The `nslookup` command can look up the new AAAA records in DNS.

NFS and RPC IPv6 Support

NFS software and Remote Procedure Call (RPC) software support IPv6 in a seamless manner. Existing commands that are related to NFS services have not changed. Most RPC applications also run on IPv6 without any change. Some advanced RPC applications with transport knowledge might require updates.

IPv6 Over ATM Support

Oracle Solaris supports IPv6 over ATM, permanent virtual circuits (PVC), and static switched virtual circuits (SVC).

PART II

DHCP

This part contains conceptual information about the Dynamic Host Configuration Protocol (DHCP), and tasks for planning, configuring, administering, and troubleshooting the DHCP service.

◆ ◆ ◆ CHAPTER 11

About DHCP (Overview)

This chapter introduces the Dynamic Host Configuration Protocol (DHCP), and explains the concepts that underlie the protocol. This chapter also describes the advantages of using DHCP in your network.

This chapter contains the following information:

About the DHCP Protocol

The DHCP protocol enables host systems in a TCP/IP network to be configured automatically for the network as the systems boot. DHCP uses a client-server mechanism. Servers store and manage configuration information for clients and provide that information upon a client's request. The information includes the client's IP address and information about network services that are available to the client.

DHCP evolved from an earlier protocol, BOOTP, which was designed for booting over a TCP/IP network. DHCP uses the same format as BOOTP for messages between the client and server. However, unlike BOOTP messages, DHCP messages can include network configuration data for the client.

A primary benefit of DHCP is its ability to manage IP address assignments through leases. *Leases* allow IP addresses to be reclaimed when they are not in use. The reclaimed IP addresses can be reassigned to other clients. A site that uses DHCP can use a smaller pool of IP addresses than would be needed if all clients were assigned a permanent IP address.

Advantages of Using DHCP

DHCP relieves you of some of the time-consuming tasks involved in setting up a TCP/IP network and in the daily management of that network. Note that in the Oracle Solaris implementation, DHCP works only with IPv4.

DHCP offers the following advantages:

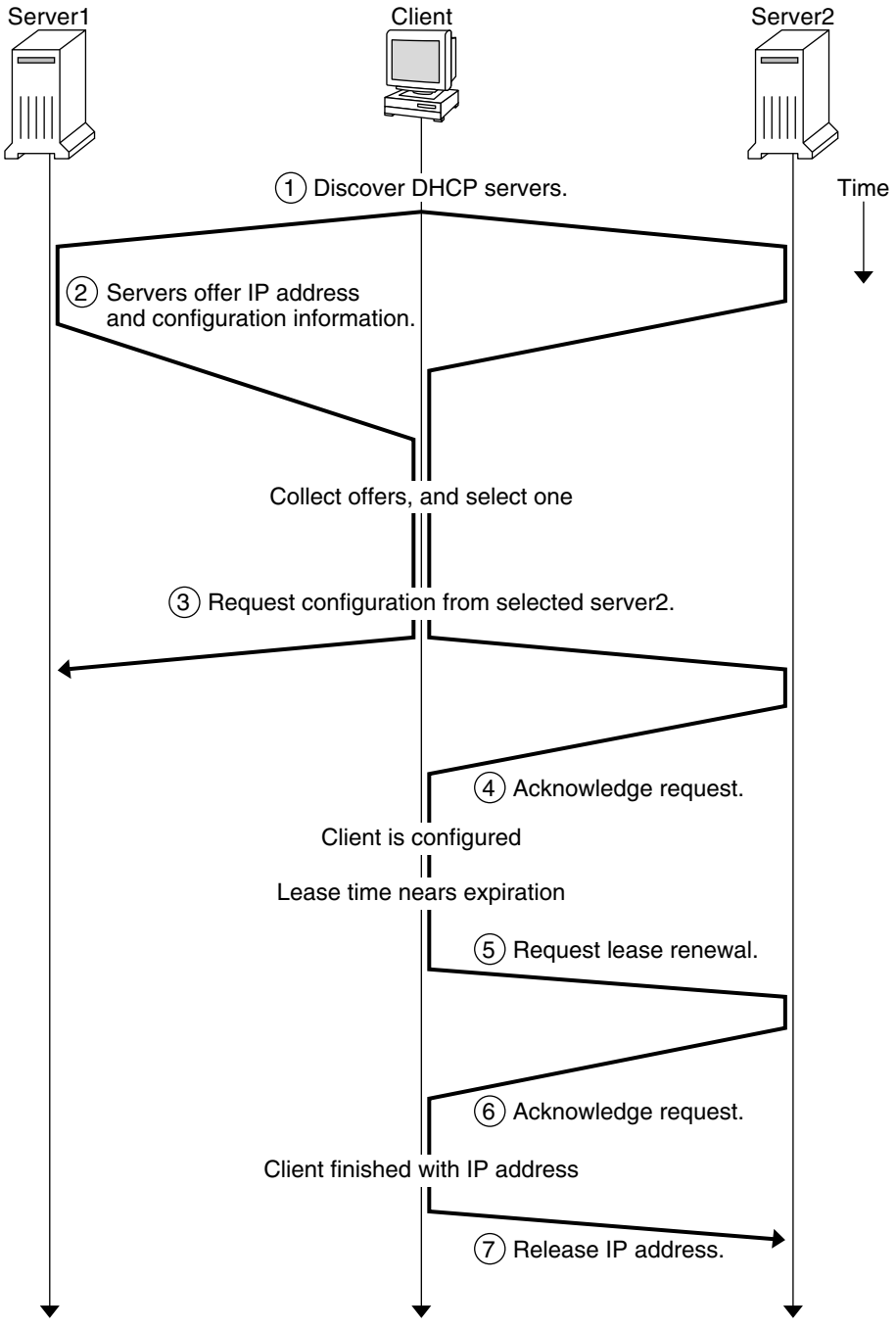
- **IP address management** – A primary advantage of DHCP is easier management of IP addresses. In a network without DHCP, you must manually assign IP addresses. You must be careful to assign unique IP addresses to each client and to configure each client individually. If a client moves to a different network, you must make manual modifications for that client. When DHCP is enabled, the DHCP server manages and assigns IP addresses without administrator intervention. Clients can move to other subnets without manual reconfiguration because they obtain, from a DHCP server, new client information appropriate for the new network.
- **Centralized network client configuration** – You can create a tailored configuration for certain clients, or for certain types of clients. The configuration information is stored in one place, in the DHCP data store. You do not need to log in to a client to change its configuration. You can make changes for multiple clients just by changing the information in the data store.
- **Support of BOOTP clients** – Both BOOTP servers and DHCP servers listen and respond to broadcasts from clients. The DHCP server can respond to requests from BOOTP clients as well as DHCP clients. BOOTP clients receive an IP address and the information needed to boot from a server.
- **Support of local clients and remote clients** – BOOTP provides for the relaying of messages from one network to another network. DHCP takes advantage of the BOOTP relay feature in several ways. Most network routers can be configured to act as BOOTP relay agents to pass BOOTP requests to servers that are not on the client's network. DHCP requests can be relayed in the same manner because, to the router, DHCP requests are indistinguishable from BOOTP requests. The DHCP server can also be configured to behave as a BOOTP relay agent, if a router that supports BOOTP relay is not available.
- **Network booting** – Clients can use DHCP to obtain the information that is needed to boot from a server on the network, instead of using RARP (Reverse Address Resolution Protocol) and the `bootparams` file. The DHCP server can give a client all the information that the client needs to function, including IP address, boot server, and network configuration information. Because DHCP requests can be relayed across subnets, you can deploy fewer boot servers in your network when you use DHCP network booting. RARP booting requires that each subnet have a boot server.
- **Large network support** – Networks with millions of DHCP clients can use DHCP. The DHCP server uses multithreading to process many client requests simultaneously. The server also supports data stores that are optimized to handle large amounts of data. Data store access is handled by separate processing modules. This data store approach enables you to add support for any database that you require.

How DHCP Works

You must first install and configure the DHCP server. During configuration, you specify information about the network that clients need to operate on the network. After this information is in place, clients are able to request and receive network information.

The sequence of events for DHCP service is shown in the following diagram. The numbers in circles correlate to the numbered items in the description following the diagram.

FIGURE 11–1 Sequence of Events for DHCP Service



The preceding diagram shows the following steps:

1. The client discovers a DHCP server by broadcasting a *discover message* to the limited broadcast address (`255.255.255.255`) on the local subnet. If a router is present and configured to behave as a BOOTP relay agent, the request is passed to other DHCP servers on different subnets. The client's *broadcast* includes its unique ID, which, in the DHCP implementation in Oracle Solaris, is derived from the client's Media Access Control (MAC) address. On an Ethernet network, the MAC address is the same as the Ethernet address.

   DHCP servers that receive the discover message can determine the client's network by looking at the following information:

   - Which network interface did the request come in on? The server determines either that the client is on the network to which the interface is connected, or that the client is using a BOOTP relay agent connected to that network.
   - Does the request include the IP address of a BOOTP relay agent? When a request passes through a relay agent, the relay agent inserts its address in the request header. When the server detects a *relay agent address*, the server knows that the network portion of the address indicates the client's network address because the relay agent must be connected to the client's network.
   - Is the client's network subnetted? The server consults the `netmasks` table to find the subnet mask used on the network indicated by the relay agent's address or by the address of the network interface that received the request. Once the server knows the subnet mask used, it can determine which portion of the network address is the host portion, and then it can select an IP address appropriate for the client. See the `netmasks(4)` man page for information on `netmasks`.
2. After the DHCP servers determine the client's network, the servers select an appropriate IP address and verify that the address is not already in use. The DHCP servers then respond to the client by broadcasting an *offer message*. The offer message includes the selected IP address and information about services that can be configured for the client. Each server temporarily reserves the offered IP address until the client determines whether to use the IP address.
3. The client selects the best offer, based on the number and type of services offered. The client broadcasts a request that specifies the IP address of the server that made the best offer. The broadcast ensures that all the responding DHCP servers know that the client has chosen a server. The servers that are not chosen can cancel the reservations for the IP addresses that they had offered.
4. The selected server allocates the IP address for the client and stores the information in the DHCP data store. The server also sends an acknowledgement message (ACK) to the client. The *acknowledgement message* contains the network configuration parameters for the client. The client uses the `ping` utility to test the IP address to make sure no other system is using it. The client then continues booting to join the network.
5. The client monitors the lease time. When a set period of time has elapsed, the client sends a new message to the chosen server to increase the lease time.

6. The DHCP server that receives the request extends the lease time if the lease still adheres to the local lease policy set by the administrator. If the server does not respond within 20 seconds, the client broadcasts a request so that one of the other DHCP servers can extend the lease.
7. When the client no longer needs the IP address, the client notifies the server that the IP address is released. This notification can happen during an orderly shutdown and can also be done manually.

The DHCP Server

The DHCP server runs as a daemon in Oracle Solaris on a host system. The server has two basic functions:

- **Managing IP addresses** – The DHCP server controls a range of IP addresses and allocates them to clients, either permanently or for a defined period of time. The server uses a lease mechanism to determine how long a client can use a nonpermanent address. When the address is no longer in use, it is returned to the pool and can be reassigned. The server maintains information about the binding of IP addresses to clients in its DHCP network tables, ensuring that no address is used by more than one client.
- **Providing network configuration for clients** – The server assigns an IP address and provides other information for network configuration, such as a host name, broadcast address, network subnet mask, default gateway, name service, and potentially much more information. The network configuration information is obtained from the server's `dhcptab` database.

The DHCP server can also be configured to perform the following additional functions:

- **Responding to BOOTP client requests** – The server listens for broadcasts from BOOTP clients discovering a BOOTP server and provides them with an IP address and boot parameters. The information must have been configured statically by an administrator. The DHCP server can simultaneously perform as a BOOTP server and as a DHCP server.
- **Relaying requests** – The server relays BOOTP and DHCP requests to appropriate servers on other subnets. The server cannot provide DHCP or BOOTP service when configured as a BOOTP relay agent.
- **Providing network booting support for DHCP clients** – The server can provide DHCP clients with information needed to boot over the network: an IP address, boot parameters, and network configuration information. The server can also provide information that DHCP clients need to boot and install over a wide area network (WAN).
- **Updating DNS tables for clients that supply a host name** – For clients that provide a `Hostname` option and value in their requests for DHCP service, the server can attempt DNS updates on their behalf.

DHCP Server Management

As superuser, you can start, stop, and configure the DHCP server with DHCP Manager or with command-line utilities described in "DHCP Command-Line Utilities" on page 235. Generally, the DHCP server is configured to start automatically when the system boots, and to stop when the system is shut down. You should not need to start and stop the server manually under normal conditions.

DHCP Data Store

All the data used by the DHCP server is maintained in a data store. The data store might consist of plain text files or binary-format files. While configuring the DHCP service, you choose the type of data store to be used. The section "Choosing the DHCP Data Store" on page 248 describes the differences between the types of data stores. You can convert a data store from one format to another by using DHCP Manager or the `dhcpconfig` command.

You can also move data from one DHCP server's data store to another server's data store. You can use export and import utilities that work with the data stores, even if the servers are using different data store formats. You can export and import the entire content of a data store, or just some of the data within it, using DHCP Manager or the `dhcpconfig` command.

Note – Any database or file format can be used for DHCP data storage if you develop your own code module to provide an interface between DHCP (server and management tools) and the database. .

Within the DHCP data store are two types of tables. You can view and manage the contents if these tables by using either DHCP Manager or the command-line utilities. The data tables are as follows:

- `dhcptab` **table** – Table of configuration information that can be passed to clients.
- **DHCP network tables** – Tables containing information about the DHCP and BOOTP clients that reside on the network specified in the table name. For example, the network `192.168.32.0` would have a table whose name includes `192_168_32_0`.

The `dhcptab` Table

The `dhcptab` table contains all the information that clients can obtain from the DHCP server. The DHCP server scans the `dhcptab` table each time it starts. The file name of the `dhcptab` table varies according to the data store used.

The DHCP protocol defines a number of standard items of information that can be passed to clients. These items are referred to as parameters, symbols, or options. Options are defined in the DHCP protocol by numeric codes and text labels, but without values. Some commonly used standard options are shown in the following table.

TABLE 11–1 Sample DHCP Standard Options

Code	Label	Description
1	`Subnet`	Subnet mask IP address
3	`Router`	IP address for the router
6	`DNSserv`	IP address for the DNS server
12	`Hostname`	Text string for the client host name
15	`DNSdmain`	DNS domain name

Some options are automatically assigned values when you provide information during server configuration. You can also explicitly assign values to other options at a later time. Options and their values are passed to the client to provide configuration information. For example, the option/value pair, `DNSdmain=Georgia.Peach.COM`, sets the client's DNS domain name to `Georgia.Peach.COM`.

Options can be grouped with other options in containers known as *macros*, which makes it easier to pass information to a client. Some macros are created automatically during server configuration and contain options that were assigned values during configuration. Macros can also contain other macros.

The format of the `dhcptab` table is described in the `dhcptab(4)` man page. In DHCP Manager, all the information shown in the Options and Macros tabs comes from the `dhcptab` table. See "About DHCP Options" on page 238 for more information about options. See "About DHCP Macros" on page 239 for more information about macros.

Note that the `dhcptab` table should not be edited manually. You should use either the `dhtadm` command or DHCP Manager to create, delete, or modify options and macros.

DHCP Network Tables

A DHCP network table maps client identifiers to IP addresses and the configuration parameters associated with each address. The format of the network tables is described in the `dhcp_network(4)` man page. In DHCP Manager, all the information shown in the Addresses tab comes from the network tables.

DHCP Manager

DHCP Manager is a graphical user interface (GUI) tool you can use to perform all management duties associated with the DHCP service. You can use it to manage the server as well as the data the server uses. You must be superuser when you run DHCP Manager.

You can use DHCP Manager with the server in the following ways:

- Configuring and unconfiguring the DHCP server
- Starting, stopping, and restarting the DHCP server
- Disabling and enabling DHCP service
- Customizing DHCP server settings

DHCP Manager enables you to manage the IP addresses, network configuration macros, and network configuration options in the following ways:

- Adding and deleting networks under DHCP management
- Viewing, adding, modifying, deleting, and releasing IP addresses under DHCP management
- Viewing, adding, modifying, and deleting network configuration macros
- Viewing, adding, modifying, and deleting nonstandard network configuration options

DHCP Manager allows you to manage the DHCP data stores in the following ways:

- Convert data to a new data store format
- Move DHCP data from one DHCP server to another by exporting it from the first server and importing it on the second server

DHCP Manager includes extensive online help for procedures you can perform with the tool. For more information, see "About DHCP Manager" on page 268.

DHCP Command-Line Utilities

All DHCP management functions can be performed by using command-line utilities. You can run the utilities if you are logged in as superuser or as a user assigned to the DHCP Management profile. See "Setting Up User Access to DHCP Commands" on page 271.

The following table lists the utilities and describes the purpose of each utility.

TABLE 11–2 DHCP Command-Line Utilities

Command	Description and Purpose	Man Page Links
`in.dhcpd`	The DHCP service daemon. Command-line arguments enable you to set several runtime options.	`in.dhcpd(1M)`

TABLE 11–2 DHCP Command-Line Utilities *(Continued)*

Command	Description and Purpose	Man Page Links
`dhcpconfig`	Used to configure and unconfigure a DHCP server. This utility enables you to perform many of the functions of DHCP Manager from the command line. This utility is primarily intended for use in scripts for sites that want to automate some configuration functions. `dhcpconfig` collects information from the server system's network topology files to create useful information for the initial configuration.	`dhcpconfig(1M)`
`dhtadm`	Used to add, delete, and modify configuration options and macros for DHCP clients. This utility lets you edit the `dhcptab` table indirectly, which ensures the correct format of the `dhcptab` table. You should not directly edit the `dhcptab` table.	`dhtadm(1M)`
`pntadm`	Used to manage the DHCP network tables. You can use this utility to perform the following tasks: ■ Add and remove IP addresses and networks under DHCP management. ■ Modify the network configuration for specified IP addresses. ■ Display information about IP addresses and networks under DHCP management.	`pntadm(1M)`

Role-Based Access Control for DHCP Commands

Security for the `dhcpconfig`, `dhtadm`, and `pntadm` commands is determined by role-based access control (RBAC) settings. By default, the commands can be run only by superuser. If you want to use the commands under another user name, you must assign the user name to the DHCP Management profile as described in "Setting Up User Access to DHCP Commands" on page 271.

DHCP Server Configuration

You configure the DHCP server the first time you run DHCP Manager on the system where you want to run the DHCP server.

DHCP Manager server configuration dialog boxes prompt you for essential information needed to enable and run the DHCP server on one network. Some default values are obtained from existing system files. If you have not configured the system for the network, there are no default values. DHCP Manager prompts for the following information:

- Role of the server, either as the DHCP server or as the BOOTP relay agent
- Data store type (files, binary files, or something specific to your site)
- Data store configuration parameters for the data store type you selected
- Name service to use to update host records, if any (`/etc/hosts`, or DNS)
- Length of lease time and whether clients should be able to renew leases
- DNS domain name and IP addresses of DNS servers
- Network address and subnet mask for the first network you want to configure for DHCP service
- Network type, either local area network (LAN) or point-to-point network
- Router discovery or the IP address of a particular router
- NIS domain name and IP address of NIS servers

You can also configure the DHCP server using the `dhcpconfig` command. This utility automatically gathers information from existing system files to provide a useful initial configuration. Therefore, you must ensure that the files are correct before running `dhcpconfig`. See the `dhcpconfig(1M)` man page for information about the files that `dhcpconfig` uses to obtain information.

IP Address Allocation

The DHCP server supports the following types of IP address allocation:

- **Manual allocation** – The server provides a specific IP address that you choose for a specific DHCP client. The address cannot be reclaimed or assigned to another client.
- **Automatic, or permanent, allocation** – The server provides an IP address that has no expiration time, making it permanently associated with the client until you change the assignment or the client releases the address.
- **Dynamic allocation** – The server provides an IP address to a requesting client, with a lease for a specific period of time. When the lease expires, the address is taken back by the server and can be assigned to another client. The period of time is determined by the lease time configured for the server.

Network Configuration Information

You determine what information to provide to DHCP clients. When you configure the DHCP server, you provide essential information about the network. Later, you can add more information that you want to provide to clients.

The DHCP server stores network configuration information in the `dhcptab` table, in the form of option/value pairs and macros. Options are keywords for network data that you want to supply to clients. Values are assigned to options and passed to clients in DHCP messages. For example, the NIS server address is passed by way of an option called `NISservs`. The `NISservs` option has a value that is equal to a list of IP addresses, which is assigned by the DHCP server. Macros provide a convenient way to group together any number of options that you want to supply to clients. You can use DHCP Manager to create macros to group options and to assign values to the options. If you prefer a command-line tool, you can use `dhtadm`, the DHCP configuration table management utility, to work with options and macros.

About DHCP Options

In DHCP, an *option* is a piece of network information to be passed to a client. The DHCP literature also refers to options as *symbols* or *tags*. An option is defined by a numeric code and a text label. An option receives a value when it is used in the DHCP service.

The DHCP protocol defines a large number of standard options for commonly specified network data: `Subnet`, `Router`, `Broadcst`, `Hostname`, and `LeaseTim` are a few examples. A complete list of standard options is shown in the `dhcp_inittab(4)` man page. You cannot modify the standard option keywords in any way. However, you can assign values to the options that are relevant to your network when you include the options in macros.

You can create new options for data that is not represented by the standard options. Options you create must be classified in one of three categories:

- **Extended** – Reserved for options that have become standard DHCP options but are not yet included in the DHCP server implementation. You might use an extended option if you know of a standard option that you want to use, but you do not want to upgrade your DHCP server.
- **Site** – Reserved for options that are unique to your site. You create these options.
- **Vendor** – Reserved for options that should apply only to clients of a particular class, such as a hardware or vendor platform. The DHCP implementation includes a number of vendor options for Oracle Solaris clients. For example, the option `SrootIP4` is used to specify the IP address of a server that a client that boots from the network should use for its root (/) file system.

Chapter 14, "Administering DHCP (Tasks)," includes procedures for creating, modifying, and deleting DHCP options.

About DHCP Macros

In the DHCP service, a *macro* is a collection of network configuration options and the values that you assign to them. Macros are created to group options together to be passed to specific clients or types of clients. For example, a macro intended for all clients of a particular subnet might contain option/value pairs for subnet mask, router IP address, broadcast address, and lease time.

Macro Processing by the DHCP Server

When the DHCP server processes a macro, it places the network options and values defined in the macro in a DHCP message to a client. The server processes some macros automatically for clients of a particular type.

For the server to process a macro automatically, the name of the macro must comply with one of the categories shown in the following table.

TABLE 11–3 DHCP Macro Categories for Automatic Processing

Macro Category	Description
Client class	The macro name matches a class of client, indicated by the client machine type, operating system, or both. For example, if a server has a macro named `SUNW.Sun-Blade-100`, any client whose hardware implementation is `SUNW,Sun-Blade-100` automatically receives the values in the `SUNW.Sun-Blade-100` macro.
Network address	The macro name matches a DHCP-managed network IP address. For example, if a server has a macro named `10.53.224.0`, any client connected to the `10.53.224.0` network automatically receives the values in the `10.53.224.0` macro.
Client ID	The macro name matches some unique identifier for the client, usually derived from an Ethernet or MAC address. For example, if a server has a macro named `08002011DF32`, the client with the client ID `08002011DF32` (derived from the Ethernet address `8:0:20:11:DF:32`) automatically receives the values in the macro named `08002011DF32`.

A macro with a name that does not use one of the categories listed in Table 11–3 can be processed only if one of the following is true:

- The macro is mapped to an IP address.
- The macro is included in another macro that is processed automatically.
- The macro is included in another macro that is mapped to an IP address.

Note – When you configure a server, a macro that is named to match the server's name is created by default. This server macro is *not* processed automatically for any client because it is not named with one of the name types that cause automatic processing. When you later create IP addresses on the server, the IP addresses are mapped to use the server macro by default.

Order of Macro Processing

When a DHCP client requests DHCP services, the DHCP server determines which macros match the client. The server processes the macros, using the macro categories to determine the order of processing. The most general category is processed first, and the most specific category is processed last. The macros are processed in the following order:

1. Client class macros – The most general category
2. Network address macros – More specific than Client class
3. Macros mapped to IP addresses – More specific than Network address
4. Client ID macros – The most specific category, pertaining to one client

A macro that is included in another macro is processed as part of the container macro.

If the same option is included in more than one macro, the value for that option in the macro with the most specific category is used because it is processed last. For example, if a Network address macro contains the lease time option with a value of 24 hours, and a Client ID macro contains the lease time option with a value of 8 hours, the client receives a lease time of 8 hours.

Size Limit for DHCP Macros

The sum total of the values assigned to all the options in a macro must not exceed 255 bytes, including the option codes and length information. This limit is dictated by the DHCP protocol.

The macros that are most likely to be impacted by this limit are macros that are used to pass paths to files on Oracle Solaris installation servers. Generally, you should pass the minimum amount of vendor information needed. You should use short path names in options that require path names. If you create symbolic links to long paths, you can pass the shorter link names.

The DHCP Client

The term “client” is sometimes used to refer to a physical machine that is performing a client role on the network. However, the DHCP client described in this document is a software entity. The DHCP client is a daemon (`dhcpagent`) that runs in Oracle Solaris on a system that is configured to receive its network configuration from a DHCP server. DHCP clients from other vendors can also use the services of the DHCP server. However, this document describes only the DHCP client.

See Chapter 15, "Configuring and Administering the DHCP Client," for detailed information about the DHCP client.

◆ ◆ ◆ CHAPTER 12

Planning for DHCP Service (Tasks)

You can use the DHCP service in a network that you are creating or in a network that exists. If you are setting up a network, see Chapter 1, "Planning an IPv4 Addressing Scheme (Tasks)," before you attempt to set up the DHCP service. If the network already exists, continue in this chapter.

This chapter describes what you need to do before you set up the DHCP service on your network. The information is intended for use with DHCP Manager, although you can also use the command-line utility `dhcpconfig` to set up the DHCP service.

This chapter contains the following information:

Preparing Your Network for the DHCP Service (Task Map)

Before you set up your network to use DHCP, you must collect information to help you make decisions for configuring one or more servers. Use the task map in the following table to identify the tasks for preparing your network for DHCP. The table lists the tasks, descriptions of what each task accomplishes, and the sections that detail the steps to perform the individual tasks.

Task	Description	For Instructions
Map your network topology.	Determine and locate the services that are available on the network.	"Mapping Your Network Topology" on page 244

Task	Description	For Instructions
Determine the number of DHCP servers you need.	Use the expected number of DHCP clients as a basis for determining the number of DHCP servers you need.	"Determining the Number of DHCP Servers" on page 245
Update system files and `netmasks` table.	Reflect the network topology accurately.	"Updating System Files and Netmask Tables" on page 246

Mapping Your Network Topology

If you have not already done so, you should map the physical structure of your network. Indicate the location of routers and clients, and the location of servers that provide network services. This map of your network topology can help you determine which server to use for the DHCP service. The map can also help you determine the configuration information that the DHCP server can provide to clients.

See Chapter 1, "Planning an IPv4 Addressing Scheme (Tasks)," for more information about planning your network.

The DHCP configuration process can gather some network information from the server's system and network files. "Updating System Files and Netmask Tables" on page 246 discusses these files. However, you might want to give clients other service information, which you must enter into the server's macros. As you examine your network topology, record the IP addresses of any servers you want your clients to know about. The following servers, for example, might provide services on your network. The DHCP configuration does not discover these servers.

- Time server
- Log server
- Print server
- Install server
- Boot server
- Web proxy server
- Swap server
- X Window font server
- Trivial File Transfer Protocol (TFTP) server

Network Topology to Avoid

In some IP network environments, several local area networks (LANs) share the same network hardware media. The networks may use multiple network hardware interfaces or multiple logical interfaces. DHCP does not work well in this kind of shared media network. When multiple LANs run across the same physical network, a DHCP client's request arrives on all network hardware interfaces. This effect makes the client appear to be attached to all of the IP networks simultaneously.

DHCP must be able to determine the address of a client's network in order to assign an appropriate IP address to the client. If more than one network is present on the hardware media, the server cannot determine the client's network. The server cannot assign an IP address without knowing the network number.

You can use DHCP on only one of the networks. If one network does not suit your DHCP needs, you must reconfigure the networks. You should consider the following suggestions:

- Use a variable length subnet mask (VLSM) on your subnets to make better use of the IP address space you have. You may not need to run multiple networks on the same physical network. See the `netmasks(4)` man page for information about implementing variable length subnetting. For more detailed information about Classless Inter-Domain Routing (CIDR) and VLSM, see `http://www.ietf.org/rfc/rfc1519.txt`.
- Configure the ports on your switches to assign devices to different physical LANs. This technique preserves the mapping of one LAN to one IP network, required for DHCP. See the documentation for the switch for information about port configuration.

Determining the Number of DHCP Servers

The data store option that you choose has a direct effect on the number of servers you must have to support your DHCP clients. The following table shows the maximum number of DHCP and BOOTP clients that can be supported by one DHCP server for each data store.

TABLE 12–1 Estimated Maximum Number of Clients Supported by One DHCP Server

Data Store Type	Maximum Number of Clients Supported
Text files	10,000
Binary files	100,000

This maximum number is a general guideline, not an absolute number. A DHCP server's client capacity depends greatly on the number of transactions per second that the server must process. Lease times and usage patterns have a significant impact on the transaction rate. For example, suppose leases are set to 12 hours and users turn their systems off at night. If many users turn on their systems at the same time in the morning, the server must handle transaction peaks as many clients request leases simultaneously. The DHCP server can support fewer clients in such an environment. The DHCP server can support more clients in an environment with longer leases, or an environment that consists of constantly connected devices such as cable modems.

The section "Choosing the DHCP Data Store" on page 248 compares the types of data stores.

Updating System Files and Netmask Tables

During DHCP configuration, the DHCP tools scan various system files on your server for information that can be used to configure the server.

You must be sure the information in the system files is current before you run DHCP Manager or `dhcpconfig` to configure your server. If you notice errors after you configure the server, use DHCP Manager or `dhtadm` to modify the macros on the server.

The following table lists some of the information gathered during DHCP server configuration, and the sources for the information. Be sure this information is set correctly on the server before you configure DHCP on the server. If you make changes to the system files after you configure the server, you should reconfigure the service to reflect these changes.

TABLE 12–2 Information Used for DHCP Configuration

Information	Source	Comments
Time zone	System date, time zone settings	The date and time zone are initially set during Oracle Solaris installation. You can change the date by using the `date` command. You can change the time zone by editing the `/etc/default/init` file to set the `TZ` environment variable. See the `TIMEZONE(4)` man page for more information.
DNS parameters	`/etc/resolv.conf`	The DHCP server uses the `/etc/resolv.conf` file to obtain DNS parameters such as the DNS domain name and DNS server addresses. See *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* or the `resolv.conf(4)` man page for more information about `resolv.conf`.
NIS parameters	System domain name, `nsswitch.conf`, NIS	The DHCP server uses the `domainname` command to obtain the domain name of the server system. The `nsswitch.conf` file tells the server where to look for domain-based information. If the server system is an NIS client, the DHCP server performs a query to get NIS server IP addresses. See the `nsswitch.conf(4)` man page for more information.

TABLE 12–2 Information Used for DHCP Configuration *(Continued)*

Information	Source	Comments
Default router	System routing tables, user prompt	The DHCP server searches the network routing tables to find the default router for clients that are attached to the local network. For clients not on the same network, the DHCP server must prompt you for the information.
Subnet mask	Network interface, `netmasks` table	The DHCP server looks to its own network interfaces to determine the netmask and broadcast address for local clients. If the request was forwarded by a relay agent, the server obtains the subnet mask in the `netmasks` table on the relay agent's network.
Broadcast address	Network interface, `netmasks` table	For the local network, the DHCP server obtains the broadcast address by querying the network interface. For remote networks, the server uses the BOOTP relay agent's IP address and the remote network's netmask to calculate the broadcast address for the network.

Making Decisions for Your DHCP Server Configuration (Task Map)

This section discusses some of the decisions to make before you configure the first DHCP server on your network. The following table guides you in the decisions you need to configure your network to use DHCP, and links each task to the section that describes the steps to perform each task.

Task	Description	For Instructions
Select a server for DHCP.	Determine if a server meets the system requirements to run the DHCP service.	"Selecting a Host to Run the DHCP Service" on page 248
Choose a data store.	Compare the data store types to determine the best data store for your site.	"Choosing the DHCP Data Store" on page 248
Set a lease policy.	Learn about IP address leases to help you determine appropriate lease policy for your site.	"Setting a Lease Policy" on page 249
Select a router address or router discovery.	Determine whether DHCP clients use router discovery or a specific router.	"Determining Routers for DHCP Clients" on page 250

Selecting a Host to Run the DHCP Service

With your network topology in mind, you can use the following system requirements to select a host on which to set up a DHCP server.

The host must meet the following requirements:

- The host must be accessible to all the networks that have clients that plan to use DHCP, either directly on the network or through a BOOTP relay agent.
- The host must be configured to use routing.
- The host must have a correctly configured `netmasks` table that reflects your network topology.

Choosing the DHCP Data Store

You can choose to store the DHCP data in text files or binary files. The following table summarizes the features of each type of data store, and indicates the environment in which to use each data store type.

TABLE 12–3 Comparison of DHCP Data Stores

Data Store Type	Performance	Maintenance	Sharing	Environment
Binary files	High performance, high capacity	Low maintenance, no database servers required. Contents must be viewed with DHCP Manager or `dhtadm` and `pntadm`. Regular file backups suggested.	Data stores cannot be shared among DHCP servers.	Midsize to large environments with many networks with thousands of clients per network. Useful for small to medium ISPs.
Text files	Moderate performance, low capacity	Low maintenance, no database servers required. ASCII format is readable without DHCP Manager, `dhtadm`, or `pntadm`. Regular file backups suggested.	Data store can be shared among DHCP servers if DHCP data is stored on one file system that is exported through an NFS mount point.	Small environments with less than 10,000 clients, with a few hundred to a thousand clients per network.

Traditional NIS is not offered as a data store option because NIS does not support fast incremental updates. If your network uses NIS, you should use text files or binary files for your data store.

Setting a Lease Policy

A *lease* specifies the amount of time the DHCP server permits a DHCP client to use a particular IP address. During the initial server configuration, you must specify a site-wide lease policy. The *lease policy* indicates the lease time and specifies whether clients can renew their leases. The server uses the information that you supply to set option values in the default macros that the server creates during configuration. You can set different lease policies for specific clients or type of clients, by setting options in configuration macros you create.

The *lease time* is specified as a number of hours, days, or weeks for which the lease is valid. When a client is assigned an IP address, or renegotiates a lease on an IP address, the lease expiration date and time is calculated. The number of hours in the lease time is added to the timestamp on the client's DHCP acknowledgement. For example, suppose the timestamp of the DHCP acknowledgment is September 16, 2005 9:15 A.M., and the lease time is 24 hours. The lease expiration time in this example is September 17, 2005 9:15 A.M. The lease expiration time is stored in the client's DHCP network record, viewable in DHCP Manager or with the `pntadm`utility.

The lease time value should be relatively small so that expired addresses are reclaimed quickly. The lease time value also should be large enough to outlast DHCP service disruptions. Clients should be able to function while the system that runs the DHCP service is repaired. A general guideline is to specify a time that is two times the predicted downtime of a system. For example, if you need four hours to obtain and replace a defective part and reboot the system, specify a lease time of eight hours.

The lease negotiation option determines whether a client can renegotiate its lease with the server before the lease expires. If lease negotiation is allowed, the client tracks the time that remains in its lease. When half of the lease time has passed, the client requests the DHCP server to extend its lease to the original lease time. You should disable lease negotiation in environments where there are more systems than IP addresses. The time limit is then enforced on the use of IP addresses. If there are enough IP addresses, you should enable lease negotiation to avoid forcing clients to take down their network interfaces when leases expire. If you make clients obtain new leases, the clients' TCP connections such as NFS and telnet sessions might be interrupted. You can enable lease negotiation for all clients during the server configuration. You can enable lease negotiation for particular clients or particular types of clients through the use of the `LeaseNeg` option in configuration macros.

Note – Systems that provide services on the network should retain their IP addresses. Such systems should not be subject to short-term leases. You can use DHCP with such systems if you assign reserved manual IP addresses to those systems, rather than IP addresses with permanent leases. You can then detect when the system's IP address is no longer in use.

Determining Routers for DHCP Clients

Host systems use routers for any network communication beyond their local network. The hosts must know the IP addresses of these routers.

When you configure a DHCP server, you must provide DHCP clients with router addresses in one of two ways. One way is to provide specific IP addresses for routers. However, the preferred method is to specify that clients should find routers with the router discovery protocol.

If clients on your network can perform router discovery, you should use the router discovery protocol, even if there is only one router. Router discovery enables a client to adapt easily to router changes in the network. For example, suppose that a router fails and is replaced by a router with a new address. Clients can discover the new address automatically without having to obtain a new network configuration to get the new router address.

Making Decisions for IP Address Management (Task Map)

As part of the DHCP service setup, you determine several aspects of the IP addresses that the server is to manage. If your network needs more than one DHCP server, you can assign responsibility for some IP addresses to each server. You must decide how to divide responsibility for the addresses. The following table is a task map that describes tasks to manage IP addresses when you use DHCP on the network. The table also contains links to the appropriate sections that detail how to perform each task.

Task	Description	For Information
Specify which addresses that the server should manage.	Determine how many addresses you want the DHCP server to manage, and what those addresses are.	"Number and Ranges of IP Addresses" on page 251
Decide if the server should automatically generate host names for clients.	Learn how client host names are generated so that you can decide whether to generate host names.	"Client Host Name Generation" on page 251
Determine what configuration macro to assign to clients.	Learn about client configuration macros so that you can select an appropriate macro for clients.	"Default Client Configuration Macros" on page 251
Determine lease types to use.	Learn about lease types to help you determine what type is best for your DHCP clients.	"Dynamic and Permanent Lease Types" on page 252

Number and Ranges of IP Addresses

During the initial server configuration, DHCP Manager allows you to add one block, or range, of IP addresses under DHCP management by specifying the total number of addresses and the first address in the block. DHCP Manager adds a list of contiguous addresses from this information. If you have several blocks of noncontiguous addresses, you can add the others by running DHCP Manager's Address Wizard again, after the initial configuration.

Before you configure your IP addresses, know how many addresses are in the initial block of addresses you want to add and the IP address of the first address in the range.

Client Host Name Generation

The dynamic nature of DHCP means that an IP address is not permanently associated with the host name of the system that is using it. The DHCP management tools can generate a client name to associate with each IP address if you select this option. The client names consist of a prefix, or root name, plus a dash and a number assigned by the server. For example, if the root name is `charlie`, the client names are `charlie-1`, `charlie-2`, `charlie-3`, and so on.

By default, generated client names begin with the name of the DHCP server that manages them. This strategy is useful in environments that have more than one DHCP server because you can quickly see in the DHCP network tables which clients any given DHCP server manages. However, you can change the root name to any name you choose.

Before you configure your IP addresses, decide if you want the DHCP management tools to generate client names, and if so, what root name to use for the names.

The generated client names can be mapped to IP addresses in `/etc/inet/hosts` or DNS if you specify to register host names during DHCP configuration. See "Client Host Name Registration" on page 282 for more information.

Default Client Configuration Macros

In DHCP, a *macro* is a collection of network configuration options and their assigned values. The DHCP server uses macros to determine what network configuration information to send to a DHCP client.

When you configure the DHCP server, the management tools gather information from system files and directly from you through prompts or command-line options you specify. With this information, the management tools create the following macros:

- **Network address macro** — The network address macro is named to match the IP address of the client network. For example, if the network is `192.68.0.0`, the network address macro is also named `192.68.0.0`. The macro contains information needed by any client that is part of the network, such as subnet mask, network broadcast address, default router or router discovery token, and NIS domain and server if the server uses NIS. Other options that are applicable to your network might be included. The network address macro is automatically processed for all clients located on that network, as described in "Order of Macro Processing" on page 240.
- **Locale macro** — The locale macro is named `Locale`. The macro contains the offset (in seconds) from Coordinated Universal Time (UTC) to specify the time zone. The locale macro is not automatically processed, but is included in the server macro.
- **Server macro** — The server macro is named to match the server's host name. For example, if the server is named `pineola`, the server macro is also named `pineola`. The server macro contains information about the lease policy, time server, DNS domain, and DNS server, and possibly other information that the configuration program was able to obtain from system files. The server macro includes the locale macro, so the DHCP server processes the locale macro as part of the server macro.

  When you configure IP addresses for the first network, you must select a client configuration macro to be used for all DHCP clients that use the addresses you are configuring. The macro that you select is mapped to the IP addresses. By default, the server macro is selected because the macro contains information needed by all clients that use this server.

Clients receive the options contained in the network address macro before the options in the macro that is mapped to IP addresses. This processing order causes the options in the server macro to take precedence over any conflicting options in the network address macro. See "Order of Macro Processing" on page 240 for more information about the order in which macros are processed.

Dynamic and Permanent Lease Types

The *lease type* determines whether the lease policy applies to the IP addresses you are configuring. During initial server configuration, DHCP Manager allows you to select either dynamic or permanent leases for the addresses you are adding. If you configure the DHCP server with the `dhcpconfig` command, leases are dynamic.

When an IP address has a *dynamic lease*, the DHCP server can manage the address. The DHCP server can allocate the IP address to a client, extend the lease time, detect when the address is no longer in use, and reclaim the address. When an IP address has a *permanent lease*, the DHCP

server can only allocate the address. The client then owns the address until explicitly releasing the address. When the address is released, the server can assign the address to another client. The address is not subject to the lease policy as long as the address is configured with a permanent lease type.

When you configure a range of IP addresses, the lease type you select applies to all the addresses in the range. To get the most benefit from DHCP, you should use dynamic leases for most of the addresses. You can later modify individual addresses to make them permanent, if necessary. However, the total number of permanent leases should be kept to a minimum.

Reserved IP Addresses and Lease Type

IP addresses can be reserved by manually assigning them to particular clients. A reserved address can be associated with a permanent lease or a dynamic lease. When a reserved address is assigned a permanent lease, the following statements are true:

- The address can be allocated only to the client that is bound to the address.
- The DHCP server cannot allocate the address to another client.
- The address cannot be reclaimed by the DHCP server.

If a reserved address is assigned a dynamic lease, the address can be allocated only to the client that is bound to the address. However, the client must track lease time and negotiate for a lease extension as if the address were not reserved. This strategy enables you to track when the client is using the address by looking at the network table.

You cannot create reserved addresses for all the IP addresses during the initial configuration. Reserved addresses are intended to be used sparingly for individual addresses.

Planning for Multiple DHCP Servers

If you want to configure more than one DHCP server to manage your IP addresses, consider the following guidelines:

- Divide the pool of IP addresses so that each server is responsible for a range of addresses, and there is no overlap of responsibility.
- Choose text files and specify a shared directory for the absolute path to the data store. The binary files data store cannot be shared.
- Configure each server separately so that address ownership is allocated correctly and so that server-based macros can be automatically created.
- Set up the servers to scan the options and macros in the `dhcptab` table at specified intervals so that the servers are using the latest information. You can use DHCP Manager to schedule automatic reading of `dhcptab` as described in "Customizing Performance Options for the DHCP Server" on page 283.

- Be sure all clients can access all DHCP servers so that the servers can support one another. A client that has a valid IP address lease might try to verify its configuration or extend the lease when the server that owns the client's address is not reachable. Another server can respond to the client after the client has attempted to contact the primary server for 20 seconds. If a client requests a specific IP address, and the server that owns the address is not available, one of the other servers handles the request. In this case, the client does not receive the requested address. The client receives an IP address that is owned by the responding DHCP server.

Planning DHCP Configuration of Your Remote Networks

After the initial DHCP configuration, you can place IP addresses in remote networks under DHCP management. However, because the system files are not local to the server, DHCP Manager and `dhcpconfig` cannot look up information to provide default values, so you must provide the information. Before you try to configure a remote network, be sure you know the following information:

- The remote network's IP address.
- The subnet mask of the remote network. This information can be obtained from the `netmasks` table in the name service. If the network uses local files, look in `/etc/netmasks` on a system in the network. If the network uses NIS, use the command `ypcat -k netmasks.byaddr`. Make sure the `netmasks` table contains all the topology information for all the subnets you want to manage.
- The network type. The clients connect to the network through either a local area network (LAN) connection or a Point-to-Point Protocol (PPP).
- Routing information. Can the clients use router discovery? If not, you must determine the IP address of a router they can use.
- NIS domain and NIS servers, if applicable.

See "Adding DHCP Networks" on page 288 for the procedure for adding DHCP networks.

Selecting the Tool for Configuring DHCP

After you gather information and plan for DHCP service, you are ready to configure a DHCP server. You can use the DHCP Manager or the command-line utility `dhcpconfig` to configure a server. DHCP Manager lets you select options and specify data that is then used to create the `dhcptab` and network tables used by the DHCP server. The `dhcpconfig` utility requires you to use command-line options to specify data.

DHCP Manager Features

DHCP Manager, a Java™ technology-based GUI tool, provides a DHCP Configuration Wizard. The configuration wizard starts automatically the first time you run DHCP Manager on a

system that is not configured as a DHCP server. The DHCP Configuration Wizard provides a series of dialog boxes that prompt you for the essential information required to configure a server: data store format, lease policy, DNS/NIS servers and domains, and router addresses. Some of the information is obtained by the wizard from system files, and you only need to confirm that the information is correct, or to correct information, if necessary.

When you progress through the dialog boxes and approve the information, the DHCP server daemon starts on the server system. You are then prompted to start the Add Addresses Wizard to configure IP addresses for the network. Only the server's network is configured for DHCP initially, and other server options are given default values. You can run DHCP Manager again after the initial configuration is complete to add networks and modify other server options.

See "Configuring and Unconfiguring a DHCP Server Using DHCP Manager" on page 257 for more information about the DHCP Configuration Wizard. See "About DHCP Manager" on page 268 for more detailed information about DHCP Manager.

`dhcpconfig` Features

The `dhcpconfig` utility supports options that enable you to configure and unconfigure a DHCP server, as well as convert to a new data store and import/export data to and from other DHCP servers. When you use the `dhcpconfig` utility to configure a DHCP server, the utility obtains information from the system files discussed in "Updating System Files and Netmask Tables" on page 246. You cannot view and confirm the information obtained from system files as you can with DHCP Manager. So, it is important that the system files be updated before you run `dhcpconfig`. You can also use command-line options to override the values `dhcpconfig` would obtain by default from system files. The `dhcpconfig` command can be used in scripts. See the `dhcpconfig(1M)` man page for more information.

Comparison of DHCP Manager and `dhcpconfig`

The following table summarizes the differences between the two server configuration tools.

TABLE 12–4 Comparison of DHCP Manager and the `dhcpconfig` Command

Feature	DHCP Manager	`dhcpconfig` With Options
Network information that is gathered from system.	Enables you to view the information gathered from system files, and to change it if needed.	You can specify the network information with command-line options.

TABLE 12–4 Comparison of DHCP Manager and the `dhcpconfig` Command *(Continued)*

Feature	DHCP Manager	`dhcpconfig` With Options
Speed of configuration.	Speeds the configuration process by omitting prompts for nonessential server options, using default values instead. You can change nonessential options after initial configuration.	Fastest configuration process, but you might need to specify values for many options.

Chapter 13, "Configuring the DHCP Service (Tasks)," includes procedures you can use to configure your server with either DHCP Manager or the `dhcpconfig` utility.

◆ ◆ ◆ CHAPTER 13

Configuring the DHCP Service (Tasks)

When you configure the DHCP service on your network, you configure and start the first DHCP server. Other DHCP servers can be added later and can access the same data from a shared location if the data store supports shared data. This chapter describes tasks that enable you to configure the DHCP server and place networks and their associated IP addresses under DHCP management. This chapter also explains how to unconfigure a DHCP server.

Each task includes a procedure to help you perform the task in DHCP Manager and a procedure for the equivalent task with the `dhcpconfig` utility. This chapter contains the following information:

- "Configuring and Unconfiguring a DHCP Server Using DHCP Manager" on page 257
- "Configuring and Unconfiguring a DHCP Server Using `dhcpconfig` Commands" on page 264

If you experience problems configuring the DHCP service, see Chapter 16, "Troubleshooting DHCP (Reference)."

After you configure the DHCP service, see Chapter 14, "Administering DHCP (Tasks)," for information about managing the DHCP service.

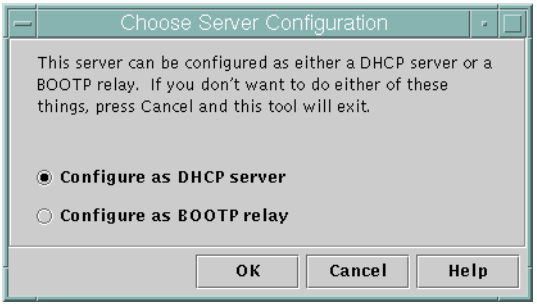
Configuring and Unconfiguring a DHCP Server Using DHCP Manager

This section includes procedures to help you configure and unconfigure a DHCP server with DHCP Manager. Note that you must be running an X Window system such as CDE or GNOME to use DHCP Manager.

DHCP Manager can be run as superuser with the `/usr/sadm/admin/bin/dhcpmgr` command. See "About DHCP Manager" on page 268 for general information about the utility. See "How to Start and Stop the DHCP Service (DHCP Manager)" on page 272 for more detailed information about running DHCP Manager.

When you run DHCP Manager on a server that is not configured for DHCP, the following screen is displayed. You can specify whether you want to configure a DHCP server or a BOOTP relay agent.

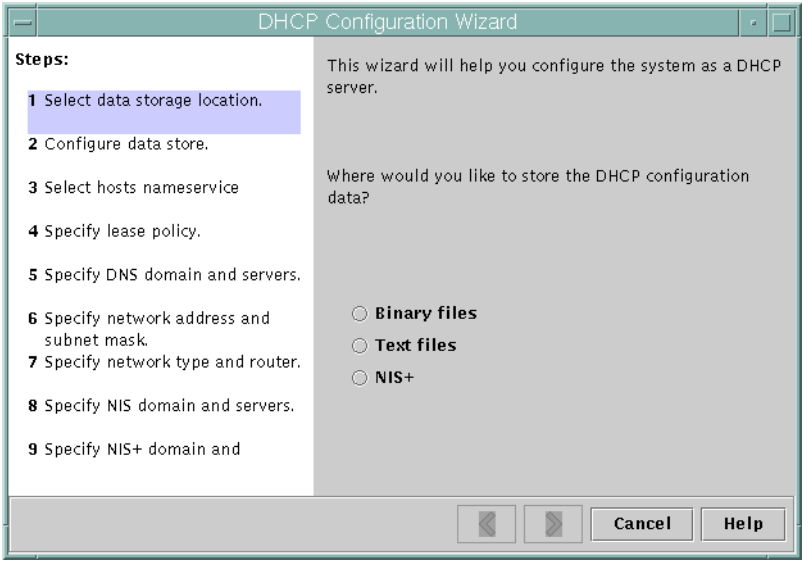
FIGURE 13–1 Choose Server Configuration Dialog Box in DHCP Manager



Configuring DHCP Servers

When you configure a DHCP server, DHCP Manager starts the DHCP Configuration Wizard, which prompts you for information that is needed to configure the server. The initial screen of the wizard is shown in the following figure.

FIGURE 13–2 DHCP Configuration Wizard's Initial Screen



When you finish answering the wizard prompts, DHCP Manager creates the items that are listed in the following table.

TABLE 13–1 Items Created During DHCP Server Configuration

Item	Description	Contents
Service configuration file, `/etc/inet/dhcpsvc.conf`	Records keywords and values for server configuration options.	Data store type and location, and options that are used with `in.dhcpd` to start the DHCP daemon when the system boots. Do not edit this file manually. You must use `dhcpmgr` or `dhcpconfig` to modify DHCP configuration information.
`dhcptab` table	DHCP Manager creates a `dhcptab` table if the table does not already exist.	Macros and options with assigned values.
Locale macro (optional), which is named `Locale`	Contains the local time zone's offset in seconds from Universal time (UTC).	`UTCoffst` option with assigned number of seconds.
Server macro, which is named to match the server's node name	Contains options whose values are determined by input from the administrator who configured the DHCP server. Options apply to all clients that use addresses owned by the server.	The `Locale` macro, plus the following options: ■ `Timeserv`, set to point to the server's primary IP address. ■ `LeaseTim`, set to the number of seconds for the leases. ■ `LeaseNeg`, if you selected negotiable leases. ■ `DNSdmain` and `DNSserv`, if DNS is configured. ■ `Hostname`, which *must not* be assigned a value. The presence of this option indicates that the host name must be obtained from the name service.
Network address macro, whose name is the same as the network address of client's network	Contains options whose values are determined by input from the administrator who configured the DHCP server. Options apply to all clients that reside on the network specified by the macro name.	The following options: ■ `Subnet`, set to the subnet mask for the local subnet ■ `Router`, set to the IP address of a router, or `RDiscvyF`, to cause the client to use router discovery ■ `Broadcst`, set to the broadcast IP address. This option is present only if the network is not a Point-to-Point network. ■ `MTU`, for the maximum transmission unit ■ `NISdmain` and `NISservs`, if NIS is configured
Network table for the network	An empty table is created until you create IP addresses for the network.	No content until you add IP addresses.

▼ How to Configure a DHCP Server (DHCP Manager)

Before You Begin Make sure that you have read Chapter 12, "Planning for DHCP Service (Tasks)," before you configure your DHCP server. In particular, you should use the guidelines in "Making Decisions for Your DHCP Server Configuration (Task Map)" on page 247 to help you perform the following tasks:

- Select the system that you want to use as a DHCP server.
- Make decisions about your data store, lease policy, and router information.

1. **Start DHCP Manager.**

   ```
   #/usr/sadm/admin/bin/dhcpmgr &
   ```

2. **Choose the option Configure as DHCP Server.**

   The DHCP Configuration Wizard starts, to help you configure your server.

3. **Select options, or type requested information, based on the decisions you made in the planning phase.**

   If you have difficulty, click Help in the wizard window to open your web browser and display help for the DHCP Configuration Wizard.

4. **Click Finish to complete the server configuration when you have finished specifying the requested information.**

5. **At the Start Address Wizard prompt, click Yes to configure IP addresses for the server.**

   The Add Addresses to Network wizard enables you to specify which addresses to place under the control of DHCP.

6. **Answer the prompts according to decisions you made in the planning phase.**

   See "Making Decisions for IP Address Management (Task Map)" on page 250 for more information. If you have difficulty, click Help in the wizard window to open your web browser and display help for the Add Addresses to Network wizard.

7. **Review your selections, and then click Finish to add the IP addresses to the network table.**

   The network table is updated with records for each address in the range you specified.

See Also You can add more networks to the DHCP server with the Network Wizard, as explained in "Adding DHCP Networks" on page 288.

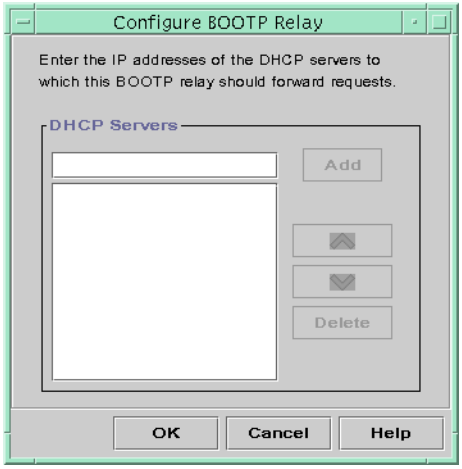
Configuring BOOTP Relay Agents

When you configure a BOOTP relay agent, DHCP Manager takes the following actions:

- Prompts you for the IP address for one or more DHCP servers to which requests should be relayed
- Stores settings needed for BOOTP relay service

The following figure shows the screen displayed when you choose to configure a BOOTP relay agent.

FIGURE 13–3 Configure BOOTP Relay Dialog Box in DHCP Manager



▼ How to Configure a BOOTP Relay Agent (DHCP Manager)

Before You Begin Make sure that you have read Chapter 12, "Planning for DHCP Service (Tasks)," before you configure your BOOTP relay agent. In particular, you should see "Selecting a Host to Run the DHCP Service" on page 248 for help in selecting the system to use.

1 Start the DHCP Manager.

```
#/usr/sadm/admin/bin/dhcpmgr &
```

If the system has not been configured as a DHCP server or BOOTP relay agent, the DHCP Configuration Wizard starts. If the system has already been configured as a DHCP server, you must first unconfigure the server. See "Unconfiguring DHCP Servers and BOOTP Relay Agents" on page 262.

2 **Select Configure as BOOTP Relay.**

The Configure BOOTP Relay dialog box opens.

3 **Type the IP address or host name of one or more DHCP servers, and click Add.**

The specified DHCP servers must be configured to handle BOOTP or DHCP requests received by this BOOTP relay agent.

4 **Click OK to exit the dialog box.**

Notice that DHCP Manager offers only the File menu to exit the application and the Service menu to manage the server. The disabled menu options are useful only on a DHCP server.

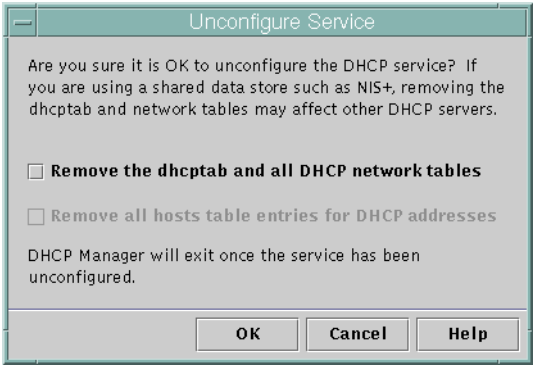
Unconfiguring DHCP Servers and BOOTP Relay Agents

When you unconfigure a DHCP server or a BOOTP relay agent, DHCP Manager takes the following actions:

- Stops the DHCP daemon (`in.dhpcd`) process
- Removes the `/etc/inet/dhcpsvc.conf` file, which records information about daemon startup and the data store location

The following figure shows the screen that is displayed when you choose to unconfigure a DHCP server.

FIGURE 13–4 Unconfigure Service Dialog Box in DHCP Manager



DHCP Data on an Unconfigured Server

When you unconfigure a DHCP server, you must decide what to do with the `dhcptab` table and the DHCP network tables. If the data is shared among servers, you should not remove the

`dhcptab` and DHCP network tables. If the tables are removed, DHCP would become unusable across your network. Data can be shared on exported local file systems. The file `/etc/inet/dhcpsvc.conf` records the data store used and its location.

You can unconfigure a DHCP server but leave the data intact by not selecting any of the options to remove data. If you unconfigure the server and leave the data intact, you disable the DHCP server.

If you want another DHCP server to take ownership of the IP addresses, you must move the DHCP data to the other DHCP server. You must move the data before you unconfigure the current server. See "Moving Configuration Data Between DHCP Servers (Task Map)" on page 333 for more information.

If you are certain you want to remove the data, you can select an option to remove the `dhcptab` and network tables. If you had generated client names for the DHCP addresses, you can also elect to remove those entries from the hosts table. Client name entries can be removed from DNS, `/etc/inet/hosts`.

Before you unconfigure a BOOTP relay agent, be sure that no clients rely on this agent to forward requests to a DHCP server.

How to Unconfigure a DHCP Server or a BOOTP Relay Agent (DHCP Manager)

1. **Start DHCP Manager.**

   ```
   #/usr/sadm/admin/bin/dhcpmgr &
   ```

2. **From the Service menu, choose Unconfigure.**

   The Unconfigure Service dialog box is displayed. If the server is a BOOTP relay agent, the dialog box enables you to confirm your intention to unconfigure the relay agent. If the server is a DHCP server, you must decide what to do with the DHCP data and make selections in the dialog box. See Figure 13–4.

3. **(Optional) Select options to remove data.**

   If the server uses shared data in files shared through NFS, do not select any options to remove the data. If the server does not use shared data, select one option or both options to remove the data.

   See "DHCP Data on an Unconfigured Server" on page 262 for more information about removing data.

4. **Click OK to unconfigure the server.**

   The Unconfigure Service dialog box and DHCP Manager are closed.

Configuring and Unconfiguring a DHCP Server Using dhcpconfig Commands

This section includes procedures to help you configure and unconfigure a DHCP server or a BOOTP relay agent by using dhcpconfig with command-line options.

▼ How to Configure a DHCP Server (dhcpconfig -D)

Before You Begin Make sure that you have read Chapter 12, "Planning for DHCP Service (Tasks)," before you configure your DHCP server. In particular, you should use the guidelines in "Making Decisions for Your DHCP Server Configuration (Task Map)" on page 247 to help you perform the following tasks:

- Select the system that you want to use as a DHCP server.
- Make decisions about your data store, lease policy, and router information.

1 Log in to the system on which you want to configure the DHCP server.

2 Configure the DHCP server by typing a command of the following format:

#/usr/sbin/dhcpconfig -D -r *datastore* **-p** *location*

datastore is one of the following: SUNWfiles, or SUNWbinfiles.

location is the data-store-dependent location where you want to store the DHCP data. For SUNWfiles and SUNWbinfiles, the location must be an absolute path name.

For example, you might type a command similar to the following:

```
dhcpconfig -D -r SUNWbinfiles -p /var/dhcp
```

The dhcpconfig utility uses the host's system files and network files to determine values used to configure the DHCP server. See the dhcpconfig(1M) man page for information about additional options to the dhcpconfig command that enable you to override the default values.

3 Add one or more networks to the DHCP service.

See "How to Add a DHCP Network (dhcpconfig)" on page 290 for the procedure to add a network.

▼ How to Configure a BOOTP Relay Agent (dhcpconfig -R)

Before You Begin Select the system that you want to use as a BOOTP relay agent, using the requirements listed in "Selecting a Host to Run the DHCP Service" on page 248.

1. **Log in to the server that you want to configure as a BOOTP relay agent.**

2. **Configure the BOOTP relay agent by typing a command of the following format:**

   ```
   # /usr/sbin/dhcpconfig -R server-addresses
   ```

   Specify one or more IP addresses of DHCP servers to which you want requests to be forwarded. If you specify more than one address, separate the addresses with commas.

   For example, you might type a command similar to the following:

   ```
   /usr/sbin/dhcpconfig -R 192.168.1.18,192.168.42.132
   ```

▼ How to Unconfigure a DHCP Server or a BOOTP Relay Agent (dhcpconfig -U)

1. **Log in to the DHCP server or the BOOTP relay agent system that you want to unconfigure.**

2. **Unconfigure the DHCP server or the BOOTP relay agent:**

   ```
   # /usr/sbin/dhcpconfig -U
   ```

   If the server does not use shared data, you can also use the `-x` option to remove the `dhcptab` and network tables. If the server uses shared data, do not use the `-x` option. The `-h` option can be used to remove host names from the host table. See the `dhcpconfig(1M)` man page for more information about `dhcpconfig` options.

   See "DHCP Data on an Unconfigured Server" on page 262 for more information about removing data.

◆ ◆ ◆ CHAPTER 14

Administering DHCP (Tasks)

This chapter describes tasks that you might find useful when you administer the DHCP service. The chapter includes tasks for the server, BOOTP relay agent, and client. Each task includes a procedure to help you perform the task in DHCP Manager and a procedure for the equivalent task with DHCP command-line utilities. DHCP command-line utilities are more fully documented in man pages.

You should have already completed the initial configuration of your DHCP service and initial network before you use this chapter. Chapter 13, "Configuring the DHCP Service (Tasks)," discusses DHCP configuration.

This chapter contains the following information:

About DHCP Manager

DHCP Manager is a graphical user interface (GUI) tool that you can use to perform administration tasks on the DHCP service.

DHCP Manager Window

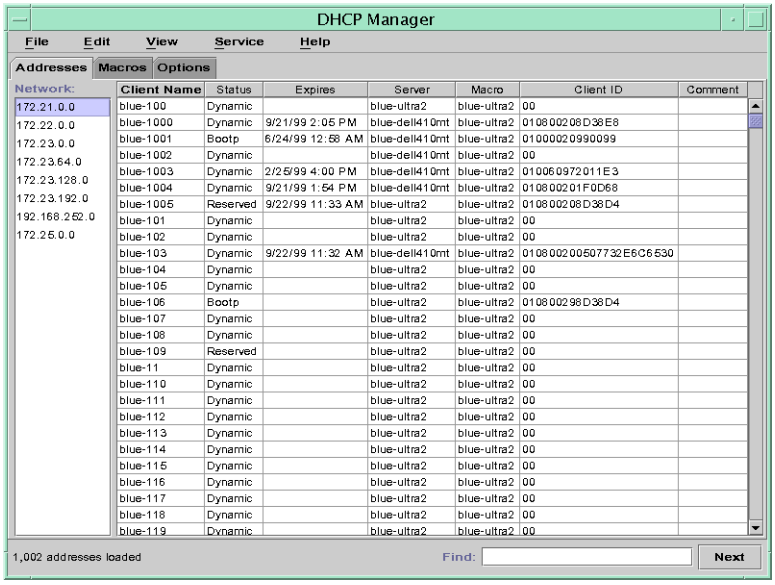
The DHCP Manager window's appearance depends on how the DHCP server is configured on the system on which DHCP Manager is running.

DHCP Manager uses a tab-based window when the system is configured as a DHCP server. You select a tab for the type of information you want to work with. DHCP Manager features the following tabs:

- **Addresses**tab – Lists all networks and IP addresses placed under DHCP management. From the Addresses tab, you can work with networks and IP addresses. You can add or delete items individually or in blocks. You can also modify the properties of individual networks or IP addresses or simultaneously make the same property modifications for a block of addresses. When you start DHCP Manager, the Addresses tab opens first.
- **Macros** tab – Lists all available macros in the DHCP configuration table (`dhcptab`) and the options contained within the macros. From the Macros tab, you can create or delete macros. You can also modify macros by adding options and providing values for the options.
- **Options** tab – Lists all options that have been defined for this DHCP server. Options that are listed on this tab are not the standard options defined in the DHCP protocol. The options are extensions to the standard options, and have a class of Extended, Vendor, or Site. Standard options cannot be changed in any way so those options are not listed here.

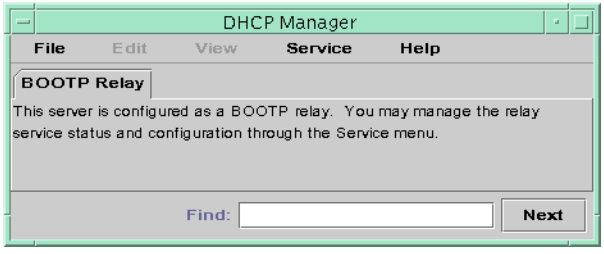
The following figure shows how the DHCP Manager window might look when you start DHCP Manager on a DHCP server.

FIGURE 14–1 DHCP Manager on a DHCP Server System



When the server is configured as a BOOTP relay agent, the DHCP Manager window does not show these tabs. The BOOTP relay agent does not need the same information. You can only modify the BOOTP relay agent's properties and stop or start the DHCP daemon with DHCP Manager. The following figure shows how DHCP Manager might look on a system that is configured as a BOOTP relay agent.

FIGURE 14–2 DHCP Manager on a BOOTP Relay Agent



DHCP Manager Menus

DHCP Manager menus include the following items:

- **File** – Exit DHCP Manager.
- **Edit** – Perform management tasks for networks, addresses, macros, and options.

- **View** – Change the look of the tab currently selected.
- **Service** – Manage the DHCP daemon and data store.
- **Help** – Open your web browser and display help for DHCP Manager.

When DHCP Manager runs on a BOOTP relay agent, the Edit and View menus are disabled.

All DHCP management tasks are accomplished through the Edit and Service menus.

You use the commands in the Edit menu to create, delete, and modify items in the selected tab. Items can include networks, addresses, macros, and options. When the Addresses tab is selected, the Edit menu also lists wizards. Wizards are sets of dialogs that help you create networks and multiple IP addresses.

The Service menu lists commands that enable you to manage the DHCP daemon. From the Service menu, you can perform the following tasks:

- Start and stop the DHCP daemon.
- Enable and disable the DHCP daemon.
- Modify the server configuration.
- Unconfigure the server.
- Convert the data store.
- Export and import data on the server.

Starting and Stopping DHCP Manager

You must run DHCP Manager on a DHCP server system as superuser. If you must run DHCP Manager remotely, you can send the display to your system by using the X Window remote display feature.

▼ How to Start and Stop DHCP Manager

1 (Optional) If you are logged in to the DHCP server system remotely, display DHCP Manager on your local system as follows.

a. Type the following on the local system:

```
# xhost +server-name
```

b. Type the following on the remote DHCP server system:

```
# DISPLAY=local-hostname;export DISPLAY
```

2 Start DHCP Manager.

```
# /usr/sadm/admin/bin/dhcpmgr &
```

The DHCP Manager window opens. If the server is configured as a DHCP server, the window displays the Addresses tab. If the server is configured as a BOOTP relay agent, the window displays with no tabs.

3 **To stop DHCP Manager, choose Exit from the File menu.**

The DHCP Manager window closes.

Setting Up User Access to DHCP Commands

By default, only root or superuser can execute `dhcpconfig`, `dhtadm`, and `pntadm` commands. If you want non root users to use the commands, you can set up role-based access control (RBAC) for those commands.

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

You might also find the following man pages helpful: `rbac(5)`, `exec_attr(4)`, and `user_attr(4)`.

The following procedure explains how to assign the DHCP Management profile, which enables the user to execute the DHCP commands.

▼ How to Grant Users Access to DHCP Commands

- **Edit the file `/etc/user_attr` to add an entry of the following form. Add one entry for each user or role that should manage the DHCP service.**

  username`::::type=normal;profiles=DHCP Management`

  For example, for user `ram`, you would add the following entry:

  ```
  ram::::type=normal;profiles=DHCP Management
  ```

Starting and Stopping the DHCP Service

This section describes starting and stopping the DHCP service by using DHCP Manager and the `dhcpconfig` command. The DHCP service can also be started and stopped by using the Service Management Facility (SMF) commands. See "DHCP Service and the Service Management Facility" on page 273 for more information about using SMF commands with the DHCP service.

Starting and stopping the DHCP service encompasses several degrees of action you can take to affect the operation of the DHCP daemon. You must understand what each action means in order to select the correct procedure to obtain the result that you want. The terms for the actions are as follows:

- **Start, stop, and restart commands** affect the daemon only for the current session. For example, if you stop the DHCP service, the daemon terminates but restarts when you reboot the system. DHCP data tables are not affected when you stop the service. You can use DHCP Manager or SMF commands to temporarily start and stop the DHCP service without enabling and disabling the service.
- **Enable and disable commands** affect the daemon for current and future sessions. If you disable the DHCP service, the currently running daemon terminates and does not start when you reboot the server. You must enable the DHCP daemon for automatic startup at system boot to occur. DHCP data tables are not affected. You can use DHCP Manager, the `dhcpconfig` command, or SMF commands to enable and disable the DHCP service.
- The **unconfigure command** shuts down the daemon, prevents the daemon from starting on system reboot, and enables you to remove the DHCP data tables. You can use DHCP Manager or the `dhcpconfig` command to unconfigure the DHCP service. Unconfiguration is described in Chapter 13, "Configuring the DHCP Service (Tasks)."

Note – If a server has multiple network interfaces but you do not want to provide DHCP services on all the networks, see "Specifying Network Interfaces for DHCP Monitoring" on page 286.

The following procedures help you start, stop, enable, and disable the DHCP service.

▼ How to Start and Stop the DHCP Service (DHCP Manager)

1. **Start DHCP Manager.**

   ```
   # /usr/sadm/admin/bin/dhcpmgr &
   ```

2. **Select one of the following:**

   - **Choose Start from the Service menu to start the DHCP service.**
   - **Choose Stop from the Service menu to stop the DHCP service.**

     The DHCP daemon stops until it is restarted, or the system reboots.
   - **Choose Restart from the Service menu to stop and immediately restart the DHCP service.**

▼ How to Enable and Disable the DHCP Service (DHCP Manager)

- **In DHCP Manager, choose one of the following:**
  - **Choose Enable from the Service menu to configure the DHCP daemon for automatic startup when the system boots.**

    The DHCP service starts immediately when it is enabled.
  - **Choose Disable from the Service menu to prevent the DHCP daemon from automatically starting when the system boots.**

    The DHCP service immediately stops when it is disabled.

▼ How to Enable and Disable the DHCP Service (`dhcpconfig -S`)

1. **Log in to the DHCP server system.**
2. **Choose one of the following:**
   - **To enable the DHCP service, type the following command:**

     ```
     # /usr/sbin/dhcpconfig -S -e
     ```
   - **To disable the DHCP service, type the following command:**

     ```
     # /usr/sbin/dhcpconfig -S -d
     ```

DHCP Service and the Service Management Facility

The Service Management Facility (SMF) is described in Chapter 11, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*. The SMF `svcadm` command can be used to enable and start the DHCP server, and disable and stop the DHCP server. However, you cannot use SMF commands to modify the DHCP service options that the DHCP tools allow you to set. In particular, service options that are stored in the `/etc/dhcp/dhcpsvc.conf` file cannot be set by using the SMF tools.

The following table maps DHCP commands to the equivalent SMF commands.

TABLE 14–1 SMF Commands For DHCP Server Tasks

Task	DHCP Command	SMF Command
Enable DHCP service	`dhcpconfig -S -e`	`svcadm enable svc:/network/dhcp-server`
Disable DHCP service	`dhcpconfig -S -d`	`svcadm disable svc:/network/dhcp-server`
Start DHCP service for current session only	None	`svcadm enable -t svc:/network/dhcp-server`
Stop DHCP service for current session	None	`svcadm disable -t svc:/network/dhcp-server`
Restart DHCP service	`dhcpconfig -S -r`	`svcadm restart svc:/network/dhcp-server`

Modifying DHCP Service Options (Task Map)

You can change values for some additional features of the DHCP service, which might not have been offered during the initial configuration with DHCP Manager. To change service options, you can use the Modify Service Options dialog box in DHCP Manager. Or you can specify options with the `dhcpconfig` command.

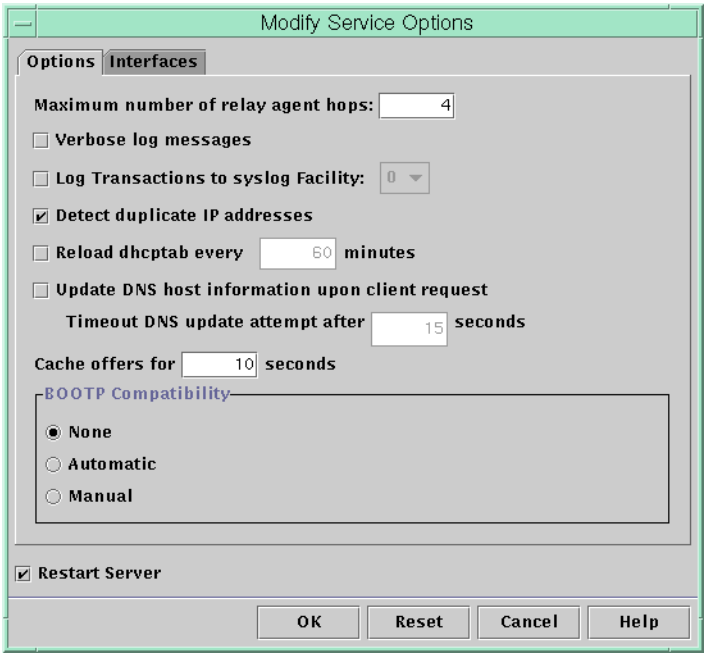
The following table is a map that describes tasks to modify DHCP service options. The table also includes links to the procedures to accomplish each task.

Task	Description	For Instructions
Change logging options.	Enable or disable logging, and select a `syslog` facility to use for logging DHCP transactions.	"How to Generate Verbose DHCP Log Messages (DHCP Manager)" on page 277 "How to Generate Verbose DHCP Log Messages (Command Line)" on page 278 "How to Enable and Disable DHCP Transaction Logging (DHCP Manager)" on page 278 "How to Enable and Disable DHCP Transaction Logging (Command Line)" on page 279 "How to Log DHCP Transactions to a Separate `syslog` File" on page 279

Task	Description	For Instructions
Change DNS update options.	Enable or disable server's capability to dynamically add DNS entries for clients that supply a host name. Determine the maximum time the server should spend attempting to update DNS.	"How to Enable Dynamic DNS Updating for DHCP Clients" on page 281
Enable or disable duplicate IP address detection.	Enable or disable the DHCP server's capability to determine that an IP address is not already in use before offering the address to a client.	"How to Customize DHCP Performance Options (DHCP Manager)" on page 284 "How to Customize DHCP Performance Options (Command Line)" on page 284
Change options for the DHCP server's reading of configuration information.	Enable or disable the automatic reading of `dhcptab` at specified intervals, or change the interval between reads.	"How to Customize DHCP Performance Options (DHCP Manager)" on page 284 "How to Customize DHCP Performance Options (Command Line)" on page 284
Change the number of relay agent hops.	Increase or decrease the number of networks a request can travel through before being dropped by the DHCP daemon.	"How to Customize DHCP Performance Options (DHCP Manager)" on page 284 "How to Customize DHCP Performance Options (Command Line)" on page 284
Change the length of time an IP address offer is cached.	Increase or decrease the number of seconds that the DHCP service reserves an offered IP address before offering the address to a new client.	"How to Customize DHCP Performance Options (DHCP Manager)" on page 284 "How to Customize DHCP Performance Options (Command Line)" on page 284

The following figure shows DHCP Manager's Modify Service Options dialog box.

FIGURE 14–3 Modify Service Options Dialog Box in DHCP Manager



Changing DHCP Logging Options

The DHCP service can log DHCP service messages and DHCP transactions to `syslog`. See the `syslogd(1M)` and `syslog.conf(4)` man pages for more information about `syslog`.

DHCP service messages logged to `syslog` include the following:

- Error messages, which notify you of conditions that prevent the DHCP service from fulfilling a request by a client or by you.
- Warnings and notices, which notify you of conditions that are abnormal, but do not prevent the DHCP service from fulfilling a request.

You can increase the amount of information that is reported by using the verbose option for the DHCP daemon. Verbose message output can help you troubleshoot DHCP problems. See "How to Generate Verbose DHCP Log Messages (DHCP Manager)" on page 277.

Another useful troubleshooting technique is transaction logging. Transactions provide information about every interchange between a DHCP server or BOOTP relay and clients. DHCP transactions include the following message types:

- ASSIGN – IP address assignment
- ACK – Server acknowledges that the client accepts the offered IP address, and sends configuration parameters

- EXTEND – Lease extension
- RELEASE – IP address release
- DECLINE – Client is declining address assignment
- INFORM – Client is requesting network configuration parameters but not an IP address
- NAK – Server does not acknowledge a client's request to use a previously used IP address
- ICMP_ECHO – Server detects potential IP address is already in use by another host

BOOTP relay transactions include the following message types:

- RELAY-CLNT – Message is being relayed from the DHCP client to a DHCP server
- RELAY–SRVR – Message is being relayed from the DHCP server to the DHCP client

DHCP transaction logging is disabled by default. When enabled, DHCP transaction logging uses the `local0` facility in `syslog` by default. DHCP transaction messages are generated with a `syslog` severity level of *notice*. This security level causes DHCP transactions to be logged to the file where other system notices are logged. However, because the `local` facility is used, the DHCP transaction messages can be logged separately from other notices. To log the transaction messages separately, you must edit the `syslog.conf` file to specify a separate log file. See the `syslog.conf(4)` man page for more information about the `syslog.conf` file.

You can disable or enable transaction logging, and you can specify a different `syslog` facility, from `local0` through `local7`, as explained in "How to Enable and Disable DHCP Transaction Logging (DHCP Manager)" on page 278. In the server system's `syslog.conf` file, you can also instruct `syslogd` to store the DHCP transaction messages in a separate file. See "How to Log DHCP Transactions to a Separate `syslog` File" on page 279 for more information.

How to Generate Verbose DHCP Log Messages (DHCP Manager)

1. **In DHCP Manager, choose Modify from the Service menu.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

   The Modify Service Options dialog box opens and displays the Options tab. See Figure 14–3.

2. **Select Verbose Log Messages.**

3. **Select Restart Server.**

   The Restart Server option is near the bottom of the dialog box.

4 **Click OK.**

The daemon runs in verbose mode for this session and each subsequent session until you reset this option. Verbose mode can reduce daemon efficiency because of the time that is taken to display messages.

▼ How to Generate Verbose DHCP Log Messages (Command Line)

- **Type the following command to set verbose mode:**

```
# /usr/sbin/dhcpconfig -P VERBOSE=true
```

The next time the DHCP server starts, the server runs in verbose mode until you turn off verbose mode.

To turn off verbose mode, type the following command:

```
# /usr/sbin/dhcpconfig -P VERBOSE=
```

This command sets the `VERBOSE` keyword to no value, which causes the keyword to be removed from the server's configuration file.

Verbose mode can reduce daemon efficiency because of the time that is taken to display messages.

▼ How to Enable and Disable DHCP Transaction Logging (DHCP Manager)

This procedure enables and disables transaction logging for all subsequent DHCP server sessions.

1 **In DHCP Manager, choose Modify from the Service menu.**

See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2 **Select Log Transactions to Syslog Facility.**

To disable transaction logging, deselect this option.

3 **(Optional) Select a local facility from 0 to 7 to use for logging DHCP transactions.**

By default, DHCP transactions are logged to the location where system notices are logged, which depends on how `syslogd` is configured. If you want the DHCP transactions to be logged to a file separate from other system notices, see "How to Log DHCP Transactions to a Separate `syslog` File" on page 279.

Message files can quickly become very large when transaction logging is enabled.

4 **Select Restart Server.**

5 **Click OK.**

The daemon logs transactions to the selected `syslog` facility for this session and each subsequent session until you disable logging.

How to Enable and Disable DHCP Transaction Logging (Command Line)

- **Choose one of the following steps:**
  - **To enable DHCP transaction logging, type the following command:**

    `# /usr/sbin/dhcpconfig -P LOGGING_FACILITY=`*syslog-local-facility*

    syslog-local-facility is a number from 0 through 7. If you omit this option, 0 is used.

    By default, DHCP transactions are logged to the location where system notices are logged, which depends on how `syslogd` is configured. If you want the DHCP transactions to be logged to a file separate from other system notices, see "How to Log DHCP Transactions to a Separate `syslog` File" on page 279.

    Message files can quickly become very large when transaction logging is enabled.

  - **To disable DHCP transaction logging, type the following command:**

    `# /usr/sbin/dhcpconfig -P LOGGING_FACILITY=`

    Note that you supply no value for the parameter.

How to Log DHCP Transactions to a Separate `syslog` File

- **Edit the `/etc/syslog.conf` file on the server system to add a line of the following format:**

  `local`*n*`.notice` *path-to-logfile*

  n is the `syslog` facility number you specified for transaction logging, and *path-to-logfile* is the complete path to the file to use for logging transactions.

  For example, you might add the following line:

  ```
  local0.notice /var/log/dhcpsrvc
  ```

  See the `syslog.conf(4)` man page for more information about the `syslog.conf` file.

Enabling Dynamic DNS Updates by a DHCP Server

DNS provides name-to-address and address-to-name services for the Internet. Once a DNS mapping is made, a system can be reached through its host name or its IP address. The system is also reachable from outside its domain.

The DHCP service can use DNS in two ways:

- The DHCP server can look up the host name that is mapped to an IP address that the server is assigning to the client. The server then returns the client's host name along with the client's other configuration information.
- The DHCP server can attempt to make a DNS mapping on a client's behalf, if the DHCP server is configured to update DNS. The client can supply its own host name when requesting DHCP service. If configured to make DNS updates, the DHCP server attempts to update DNS with the client's suggested host name. If the DNS update is successful, the DHCP server returns the requested host name to the client. If the DNS update is not successful, the DHCP server returns a different host name to the client.

You can enable the DHCP service to update the DNS service for DHCP clients that supply their own host names. For the DNS update feature to work, the DNS server, the DHCP server, and the DHCP client must be set up correctly. In addition, the requested host name must not be in use by another system in the domain.

The DHCP server's DNS update feature works if the following statements are true:

- The DNS server supports RFC 2136.
- The DNS software is based on BIND v8.2.2, patch level 5 or later, whether on the DHCP server system or the DNS server system.
- The DNS server is configured to accept dynamic DNS updates from the DHCP server.
- The DHCP server is configured to make dynamic DNS updates.
- DNS support is configured for the DHCP client's network on the DHCP server.
- The DHCP client is configured to supply a requested host name in its DHCP request message.
- The requested host name corresponds to a DHCP-owned address. The host name could also have no corresponding address.

How to Enable Dynamic DNS Updating for DHCP Clients

Note – Be aware that dynamic DNS updates are a *security risk*.

By default, the Oracle Solaris DNS daemon (`in.named`) does not allow dynamic updates. Authorization for dynamic DNS updates is granted in the `named.conf` configuration file on the DNS server system. No other security is provided. You must carefully weigh the convenience of this facility for users against the security risk created when you enable dynamic DNS updates.

1. **On the DNS server, edit the `/etc/named.conf` file.**

2. **Find the `zone` section for the appropriate domain in the `named.conf` file.**

3. **Add the DHCP server's IP addresses to the `allow-update` keyword.**

   If the `allow-update` keyword does not exist, insert the keyword.

   For example, if the DHCP server resides at addresses `10.0.0.1` and `10.0.0.2`, a `named.conf` file for the `dhcp.domain.com` zone should be modified as follows:

```
zone "dhcp.domain.com" in {
             type master;
             file "db.dhcp";
             allow-update { 10.0.0.1; 10.0.0.2; };
};

zone "10.IN-ADDR.ARPA" in {
             type master;
             file "db.10";
             allow-update { 10.0.0.1; 10.0.0.2; };
};
```

   Note that `allow-update` for both zones must be enabled to allow the DHCP server to update both A and PTR records on the DNS server.

4. **On the DHCP server, start DHCP Manager.**

```
# /usr/sadm/admin/bin/dhcpmgr &
```

   See "How to Start and Stop DHCP Manager" on page 270 for more detailed information.

5. **Choose Modify from the Service menu.**

   The Modify Service Options dialog box opens.

6. **Select Update DNS Host Information Upon Client Request.**

7 **Specify the number of seconds to wait for a response from the DNS server before timing out, then click OK.**

The default value of 15 seconds should be adequate. If you have time out problems, you can increase the value later.

8 **Click the Macros tab, and ensure that the correct DNS domain is specified.**

The `DNSdmain` option must be passed with the correct domain name to any client that expects dynamic DNS update support. By default, `DNSdmain` is specified in the server macro, which is used as the configuration macro bound to each IP address.

9 **Set up the DHCP client to specify its host name when requesting DHCP service.**

If you use the DHCP client, see "How to Enable a DHCPv4 Client to Request a Specific Host Name" on page 353. If your client is not a DHCP client, see the documentation for your DHCP client for information about how to specify a host name.

Client Host Name Registration

If you let the DHCP server generate host names for the IP addresses that you place in the DHCP service, the DHCP server can register those host names in DNS name services. Host name registration cannot be done in NIS because NIS does not provide a protocol to allow programs to update and propagate NIS maps.

Note – The DHCP server can update DNS with generated host names only if the DNS server and the DHCP server are running on the same system.

If a DHCP client provides its host name and the DNS server is configured to allow dynamic updates from the DHCP server, the DHCP server can update DNS on the client's behalf. Dynamic updates can be done even if the DNS and DHCP servers are running on different systems. See "Enabling Dynamic DNS Updates by a DHCP Server" on page 280 for more information about enabling this feature.

The following table summarizes client host name registration for DHCP client systems with the various name services.

TABLE 14–2 Client Host Name Registration in Name Services

	Who Registers Host Name	
Name Service	**DHCP-Generated Host Name**	**DHCP Client-Supplied Host Name**
NIS	NIS Administrator	NIS Administrator
`/etc/hosts`	DHCP tools	DHCP tools

TABLE 14–2 Client Host Name Registration in Name Services *(Continued)*

	Who Registers Host Name	
Name Service	**DHCP-Generated Host Name**	**DHCP Client-Supplied Host Name**
DNS	DHCP tools, if the DNS server runs on the same system as the DHCP server	DHCP server, if configured for dynamic DNS updates
	DNS Administrator, if the DNS server runs on a different system	DNS Administrator, if DHCP server is not configured for dynamic DNS updates

DHCP clients can request particular host names in DHCP requests if configured to do so as described in "How to Enable a DHCPv4 Client to Request a Specific Host Name" on page 353. Refer to the vendor documentation for other DHCP clients to determine if the capability is supported.

Customizing Performance Options for the DHCP Server

You can change options that affect the performance of the DHCP server. These options are described in the following table.

TABLE 14–3 Options Affecting DHCP Server Performance

Server Option	Description	Keyword
Maximum number of BOOTP relay agent hops	If a request has traveled through more than a given number of BOOTP relay agents, the request is dropped. The default maximum number of relay agent hops is four. This number is likely to be sufficient for most networks. A network might need more than four hops if DHCP requests pass through several BOOTP relay agents before reaching a DHCP server.	`RELAY_HOPS=`*integer*
Detect duplicate addresses	By default, the server pings an IP address before offering the address to a client. A lack of response to the ping verifies that the address is not already in use. You can disable this feature to decrease the time that the server takes to make an offer. However, disabling the feature creates the risk of having duplicate IP addresses in use.	`ICMP_VERIFY=TRUE/FALSE`
Reload `dhcptab` automatically at specified intervals	The server can be set to automatically read the `dhcptab` at the interval, in minutes, that you specify. If your network configuration information does not change frequently, and you do not have multiple DHCP servers, you do not need to reload the `dhcptab` automatically. Also, note that DHCP Manager gives you the option to have the server reload the `dhcptab` after you make a change to the data.	`RESCAN_INTERVAL=`*min*

TABLE 14–3 Options Affecting DHCP Server Performance *(Continued)*

Server Option	Description	Keyword
Cache offers of IP addresses for specified intervals	After a server offers an IP address to a client, the offer is cached. While the offer is cached, the server does not offer the address again. You can change the number of seconds for which the offer is cached. The default is 10 seconds. On slow networks, you might need to increase the offer time.	`OFFER_CACHE_TIMEOUT=`*sec*

The following procedures describe how to change these options.

▼ How to Customize DHCP Performance Options (DHCP Manager)

1. **In DHCP Manager, choose Modify from the Service menu.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Change the desired options.**

   See Table 14–3 for information about the options.

3. **Select Restart Server.**

4. **Click OK.**

▼ How to Customize DHCP Performance Options (Command Line)

If you change options with this procedure, the changed options are used only after the DHCP server is restarted.

- **Modify one or more performance options:**

  `# /usr/sbin/dhcpconfig -P` *keyword=value,keyword=value...*

  keyword=value can be any of the following keywords:

  `RELAY_HOPS=`*integer* — Specifies the maximum number of relay agent hops that can occur before the daemon drops the DHCP or BOOTP datagram.

  `ICMP_VERIFY=TRUE/FALSE` — Enables or disables automatic duplicate IP address detection. Setting this keyword to `FALSE` is not recommended.

`RESCAN_INTERVAL=`*minutes*	Specifies the interval in minutes that the DHCP server should use to schedule the automatic rereading of the `dhcptab` information.
`OFFER_CACHE_TIMEOUT=`*seconds*	Specifies the number of seconds the DHCP server should cache the offers that are extended to discovering DHCP clients. The default setting is 10 seconds.

Example 14–1 Setting DHCP Performance Options

The following is an example of how to specify all the command options.

```
# dhcpconfig -P RELAY_HOPS=2,ICMP_VERIFY=TRUE,\
RESCAN_INTERVAL=30,OFFER_CACHE_TIMEOUT=20
```

Adding, Modifying, and Removing DHCP Networks (Task Map)

When you configure a DHCP server, you must also configure at least one network in order to use the DHCP service. You can add more networks at any time.

The following table is a map that describes additional tasks that you can perform when working with DHCP networks after their initial configuration. The task map includes links to procedures for carrying out the tasks.

Task	Description	For Instructions
Enable or disable the DHCP service on server network interfaces	The default behavior is to monitor all network interfaces for DHCP requests. If you do not want all interfaces to accept DHCP requests, you can remove an interface from the list of monitored interfaces.	"How to Specify Network Interfaces for DHCP Monitoring (DHCP Manager)" on page 287
Add a new network to the DHCP service.	Places a network under DHCP management, for the purpose of managing IP addresses on the network.	"How to Add a DHCP Network (DHCP Manager)" on page 289 "How to Add a DHCP Network (`dhcpconfig`)" on page 290
Change parameters of a DHCP-managed network.	Modifies the information that is passed to clients of a particular network.	"How to Modify the Configuration of a DHCP Network (DHCP Manager)" on page 291 "How to Modify the Configuration of a DHCP Network (`dhtadm`)" on page 292

Task	Description	For Instructions
Delete a network from the DHCP service.	Removes a network so that IP addresses on the network are no longer managed by DHCP.	"How to Remove a DHCP Network (DHCP Manager)" on page 293 "How to Remove a DHCP Network (`pntadm`)" on page 294

Specifying Network Interfaces for DHCP Monitoring

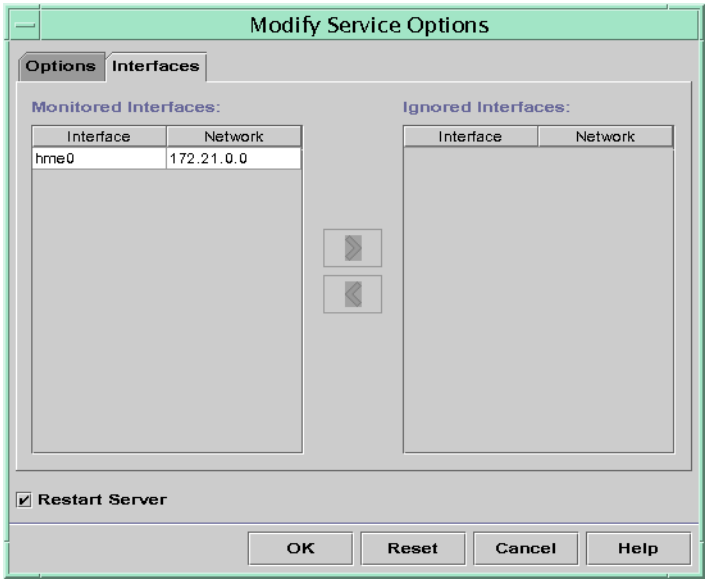
By default, both `dhcpconfig` and DHCP Manager's Configuration Wizard configure the DHCP server to monitor all the server system's network interfaces. If you add a new network interface to the server system, the DHCP server automatically monitors the new interface when you boot the system. You can then add any networks to be monitored through the network interface.

However, you can also specify which network interfaces should be monitored, and which interfaces should be ignored. You might want to ignore an interface if you do not want to offer DHCP service on that network.

If you specify that any interface should be ignored, and then install a new interface, the DHCP server ignores the new interface. You must add the new interface to the server's list of monitored interfaces. You can specify interfaces with DHCP Manager or the `dhcpconfig` utility.

This section includes procedures that enable you to specify which network interfaces DHCP should monitor or ignore. The DHCP Manager procedure uses the Interfaces tab of the DHCP Manager's Modify Service Options dialog box, which is shown in the following figure.

FIGURE 14–4 Interfaces Tab of Modify Service Options Dialog Box in DHCP Manager



How to Specify Network Interfaces for DHCP Monitoring (DHCP Manager)

1. **In DHCP Manager, choose Modify from the Service menu.**

   The Modify Service Options dialog box is displayed.

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Select the Interfaces tab.**

3. **Select the appropriate network interface.**

4. **Click the arrow buttons to move the interface to the appropriate list.**

   For example, to ignore an interface, select the interface in the Monitored Interfaces list, and then click the right arrow button. The interface is then shown in the Ignored Interfaces list.

5. **Select Restart Server, and click OK.**

   The changes you make persist across reboots.

▼ How to Specify Network Interfaces for DHCP Monitoring (`dhcpconfig`)

- **Type the following command on the DHCP server system:**

  ```
  # /usr/sbin/dhcpconfig -P INTERFACES=int,int,...
  ```

  int, int,... is a list of interfaces to monitor. The interface names must be separated by commas.

  For example, you would use the following command to monitor only `ge0` and `ge1`:

  ```
  #/usr/sbin/dhcpconfig -P INTERFACES=ge0,ge1
  ```

  Interfaces that you want to ignore should be omitted from the `dhcpconfig` command line.

  The changes you make with this command persist across reboots.

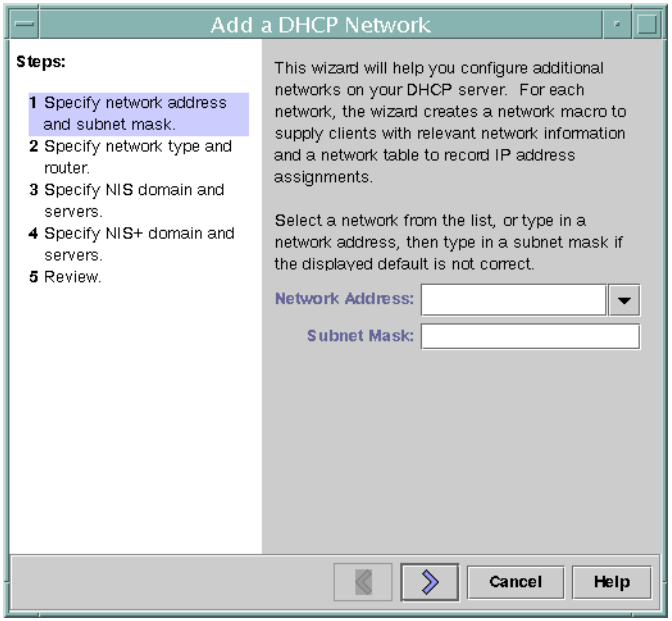
Adding DHCP Networks

When you use DHCP Manager to configure the server, the first network is also configured at the same time. The first network is usually the local network on the server system's primary interface. If you want to configure additional networks, use the DHCP Network Wizard in DHCP Manager.

If you use the `dhcpconfig -D` command to configure the server, you must separately configure all networks that you want to use the DHCP service. See "How to Add a DHCP Network (`dhcpconfig`)" on page 290 for more information.

The following figure shows the initial dialog box for the DHCP Network Wizard in DHCP Manager.

FIGURE 14–5 DHCP Manager's Network Wizard



When you configure a new network, DHCP Manager creates the following components:

- A network table in the data store. The new network is shown in the network list within the Addresses tab of DHCP Manager.
- A network macro that contains information needed by clients that reside on this network. The network macro's name matches the IP address of the network. The network macro is added to the `dhcptab` table in the data store.

▼ How to Add a DHCP Network (DHCP Manager)

1. **In DHCP Manager, click the Addresses tab.**

   Any networks already configured for DHCP service are listed.

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Choose Network Wizard from the Edit menu.**

3 **Select options, or type requested information. Use the decisions that you made during the planning phase to determine what information to specify.**

Planning is described in "Planning DHCP Configuration of Your Remote Networks" on page 254.

If you have difficulty with the wizard, click Help in the wizard window. Your web browser displays help for the DHCP Network Wizard.

4 **Click Finish to complete the network configuration when you have finished specifying the requested information.**

The Network Wizard creates an empty network table, which is listed in the left pane of the window.

The Network Wizard also creates a network macro whose name matches the IP address of the network.

5 **(Optional) Select the Macros tab and select the network macro to view the macro's contents.**

You can confirm that the information that you provided in the wizard has been inserted as values for options in the network macro.

See Also You must add addresses for the network before the network's IP addresses can be managed under DHCP. See "Adding IP Addresses to the DHCP Service" on page 301 for more information.

If you leave the network table empty, the DHCP server can still provide configuration information to clients. See "Setting Up DHCP Clients to Receive Information Only (Task Map)" on page 330 for more information.

▼ How to Add a DHCP Network (`dhcpconfig`)

- **Type the following command on the DHCP server system:**

```
# /usr/sbin/dhcpconfig -N network-address
```

network-address is the IP address of the network you want to add to the DHCP service. See the `dhcpconfig`(1M) man page for suboptions you can use with the `-N` option.

If you do not use suboptions, `dhcpconfig` uses network files to obtain information about the network.

See Also You must add addresses for the network before the network's IP addresses can be managed under DHCP. See "Adding IP Addresses to the DHCP Service" on page 301 for more information.

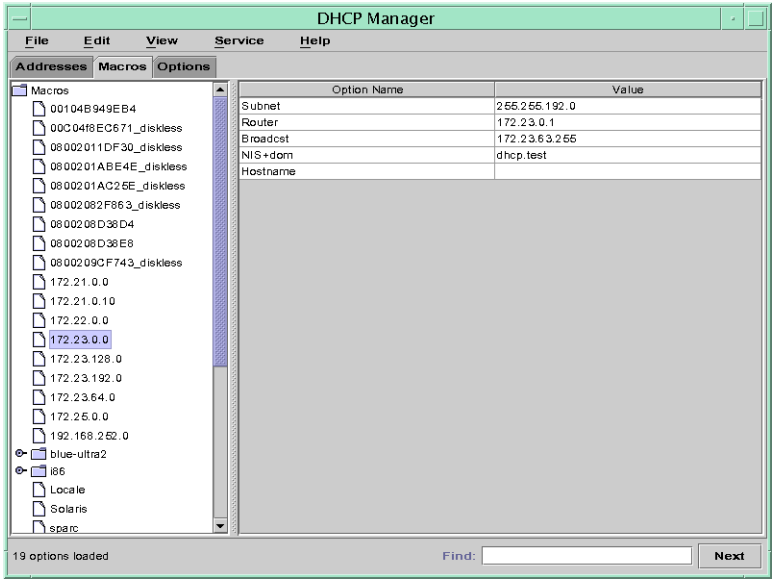
If you leave the network table empty, the DHCP server can still provide configuration information to clients. See "Setting Up DHCP Clients to Receive Information Only (Task Map)" on page 330 for more information.

Modifying DHCP Network Configurations

After you add a network to the DHCP service, you can modify the configuration information that you originally supplied. The configuration information is stored in the network macro used to pass information to clients on the network. You must modify the network macro to change the network configuration.

The following figure shows the Macros tab of DHCP Manager.

FIGURE 14–6 DHCP Manager's Macros Tab



▼ How to Modify the Configuration of a DHCP Network (DHCP Manager)

1 **In DHCP Manager, select the Macros tab.**

All macros that are defined for this DHCP server are listed in the left pane.

See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2 **Select the network macro whose name matches the network configuration that you are changing.**

The network macro name is the network IP address.

3 **Choose Properties from the Edit menu.**

The Macro Properties dialog box displays a table of the options included in the macro.

4 **Select the option that you want to modify.**

The option name and its value are displayed in text fields near the top of the dialog box.

5 **(Optional) Modify the option name, or choose the Select button to display a list of option names.**

The Select Option dialog box displays a list of all DHCP standard options, with a brief description of each option.

6 **(Optional) Select an option name in the Select Option dialog box, and click OK.**

The new option name is displayed in the Option Name field.

7 **Type the new value for the option, and click Modify.**

8 **(Optional) You can also add options to the network macro by choosing Select in the dialog box.**

See "Modifying DHCP Macros" on page 314 for more general information about modifying macros.

9 **Select Notify DHCP Server of Change, and click OK.**

This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

▼ How to Modify the Configuration of a DHCP Network (`dhtadm`)

1 **Determine which macro includes information for all clients of the network.**

The network macro's name matches the network IP address.

If you don't know which macro includes this information, you can display the `dhcptab` table to list all macros by using the command `dhtadm -P`.

2 **Type a command of the following format to change the value of the option you want to change:**

```
# dhtadm -M -m macro-name -e 'symbol=value' -g
```

See the `dhtadm`(1M) man page for more information about `dhtadm` command-line options.

Example 14–2 Using the `dhtadm` Command to Modify a DHCP Macro

For example, to change the `10.25.62.0` macro's lease time to 57600 seconds and the NIS domain to `sem.example.com`, you would type the following commands:

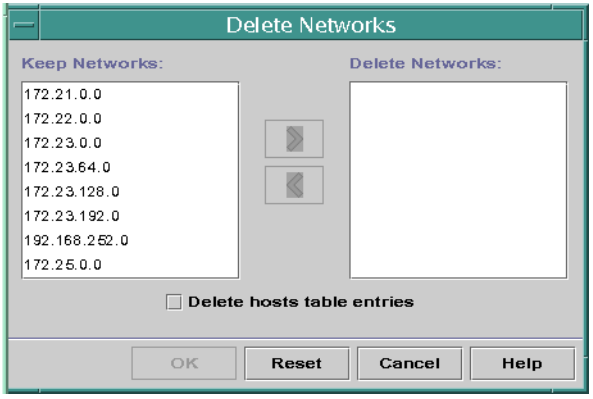
```
# dhtadm -M -m 10.25.62.0 -e 'LeaseTim=57600' -g

# dhtadm -M -m 10.25.62.0 -e 'NISdmain=sem.example.com' -g
```

The `-g` option causes the DHCP daemon to reread the `dhcptab` table and put the changes into effect.

Removing DHCP Networks

DHCP Manager enables you to remove multiple networks at once. You have the option to automatically remove the hosts table entries associated with the DHCP-managed IP addresses on those networks as well. The following figure shows DHCP Manager's Delete Networks dialog box.

FIGURE 14–7 Delete Networks Dialog Box in DHCP Manager



The `pntadm` command requires you to delete each IP address entry from a network before you delete that network. You can delete only one network at a time.

▼ How to Remove a DHCP Network (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2 **Choose Delete Networks from the Edit menu.**

The Delete Networks dialog box opens.

3 **In the Keep Networks list, select the networks that you want to delete.**

Press the Control key while you click with the mouse to select multiple networks. Press the Shift key while you click to select a range of networks.

4 **Click the right arrow button to move the selected networks to the Delete Networks list.**

5 **If you want to remove the host table entries for this network's DHCP addresses, select Delete Host Table Entries.**

Note that deleting host table entries does not delete the host registrations at the DNS server for these addresses. Entries are deleted only in the local name service.

6 **Click OK.**

▼ How to Remove a DHCP Network (`pntadm`)

Note that this procedure deletes the network's IP addresses from the DHCP network table before removing the network. The addresses are deleted to ensure that the host names are removed from the `hosts` file or database.

1 **Type a command following this format to remove an IP address and its host name from the name service:**

`# pntadm -D -y` *IP-address*

For example, to remove IP address `10.25.52.1`, you would type the following command:

```
# pntadm -D -y 10.25.52.1
```

The `-y` option specifies to delete the host name.

2 **Repeat the `pntadm -D -y` command for each address in the network.**

You might want to create a script to run the `pntadm` command if you are deleting many addresses.

3 **After all addresses are deleted, type the following command to delete the network from the DHCP service.**

`# pntadm -R` *network-IP-address*

For example, to remove network `10.25.52.0`, you would type the following command:

```
# pntadm -R 10.25.52.0
```

See the `pntadm`(1M) man page for more information about using the `pntadm` utility.

Supporting BOOTP Clients With the DHCP Service (Task Map)

To support BOOTP clients on your DHCP server, you must set up your DHCP server to be BOOTP compatible. If you want to specify which BOOTP clients can use your DHCP, you can register BOOTP clients in the DHCP server's network table. Alternatively, you can reserve a number of IP addresses for automatic allocation to BOOTP clients.

Note – BOOTP addresses are permanently assigned, whether or not you explicitly assign a permanent lease to the address.

The following table describes tasks that you might need to perform to support BOOTP clients. The task map contains links to the procedures used to carry out the tasks.

Task	Description	For Instructions
Set up automatic BOOTP support.	Provides IP address for any BOOTP client on a DHCP-managed network, or on a network connected by a relay agent to a DHCP-managed network. You must reserve a pool of addresses for exclusive use by BOOTP clients. This option might be more useful if the server must support a large number of BOOTP clients.	"How to Set Up Support of Any BOOTP Client (DHCP Manager)" on page 295
Set up manual BOOTP support.	Provides IP address for only those BOOTP clients that have been manually registered with the DHCP service. This option requires you to bind a client's ID to a particular IP address that has been marked for BOOTP clients. This option is useful for a small number of BOOTP clients, or when you want to restrict the BOOTP clients that can use the DHCP server.	"How to Set Up Support of Registered BOOTP Clients (DHCP Manager)" on page 296

▼ How to Set Up Support of Any BOOTP Client (DHCP Manager)

1 **In DHCP Manager, select Modify from the Service menu.**

The Modify Service Options dialog box opens.

See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2 **In the BOOTP Compatibility section of the dialog box, select Automatic.**

3 **Select Restart Server, and click OK.**

4 **Select the Addresses tab.**

5 **Select addresses that you want to reserve for BOOTP clients.**

Select a range of addresses by clicking the first address, pressing the Shift key, and clicking the last address. Select multiple nonconcurrent addresses by pressing the Control key while clicking each address.

6 **Select Properties from the Edit menu.**

The Modify Multiple Addresses dialog box opens.

7 **In the BOOTP section, select Assign All Addresses Only to BOOTP Clients.**

All other options should be set to Keep Current Settings.

8 **Click OK.**

Any BOOTP client can now obtain an address from this DHCP server.

▼ How to Set Up Support of Registered BOOTP Clients (DHCP Manager)

1 **In DHCP Manager, select Modify from the Service menu.**

The Modify Service Options dialog box opens.

See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2 **In the BOOTP Compatibility section of the dialog box, select Manual.**

3 **Select Restart Server, and click OK.**

4 **Select the Addresses tab.**

5 **Select an address that you want to assign to a particular BOOTP client.**

6. **Choose Properties from the Edit menu.**

   The Address Properties dialog box opens.

7. **In the Address Properties dialog box, select the Lease tab.**

8. **In the Client ID field, type the client's identifier.**

   For a BOOTP Oracle Solaris client on an Ethernet network, the client ID is a string that is derived from the client's hexadecimal Ethernet address. The client ID includes a prefix that indicates the Address Resolution Protocol (ARP) type for Ethernet (01). For example, a BOOTP client with the Ethernet address `8:0:20:94:12:1e` would use the client ID `0108002094121E`.

   Tip – As superuser on an Oracle Solaris client system, type the following command to obtain the Ethernet address for the interface:

   ```
   # ipadm show-addr
   ```

9. **Select Reserved to reserve the IP address for this client.**

10. **Select Assign Only to BOOTP Clients, and click OK.**

    In the Addresses tab, BOOTP is displayed in the Status field, and the client ID you specified is listed in the Client ID field.

Working With IP Addresses in the DHCP Service (Task Map)

You can use DHCP Manager or the `pntadm` command to add IP addresses, modify address properties, and remove addresses from the DHCP service. Before you work with IP addresses, you should refer to Table 14–4 to become familiar with IP address properties. The table provides information for users of DHCP Manager and `pntadm`.

Note – Table 14–4 includes examples of using `pntadm` to specify IP address properties while adding and modifying IP addresses. Refer also to the `pntadm(1M)` man page for more information about `pntadm`.

The following task map lists tasks that you must perform to add, modify, or remove IP addresses. The task map also contains links to the procedures used to carry out the tasks.

Task	Description	For Instructions
Add single or multiple IP addresses to the DHCP service.	Adds IP addresses on networks that are already managed by the DHCP service by using DHCP Manager.	"How to Add a Single IP Address (DHCP Manager)" on page 303 "How to Duplicate an Existing IP Address (DHCP Manager)" on page 303 "How to Add Multiple IP Addresses (DHCP Manager)" on page 304 "How to Add IP Addresses (`pntadm`)" on page 304
Change properties of an IP address.	Changes any of the IP address properties described in Table 14–4.	"How to Modify IP Address Properties (DHCP Manager)" on page 306 "How to Modify IP Address Properties (`pntadm`)" on page 306
Remove IP addresses from the DHCP service.	Prevents the use of specified IP addresses by DHCP.	"How to Mark IP Addresses as Unusable (DHCP Manager)" on page 307 "How to Mark IP Addresses as Unusable (`pntadm`)" on page 308 "How to Delete IP Addresses From DHCP Service (DHCP Manager)" on page 309 "How to Delete IP Addresses From the DHCP Service (`pntadm`)" on page 309
Assign a consistent IP address to a DHCP client.	Sets up a client to receive the same IP address each time the client requests its configuration.	"How to Assign a Consistent IP Address to a DHCP Client (DHCP Manager)" on page 311 "How to Assign a Consistent IP Address to a DHCP Client (`pntadm`)" on page 311

The following table lists and describes the properties of IP addresses.

TABLE 14–4 IP Address Properties

Property	Description	How to Specify in `pntadm` Command
Network address	The address of the network that contains the IP address that you are working with. The network address is displayed in the Networks list within the Addresses tab in DHCP Manager.	The network address must be the last argument on the `pntadm` command line used to create, modify, or delete an IP address. For example, to add an IP address to network `10.21.0.0`, you would type: **`pntadm -A`** *ip-address options* **`10.21.0.0`**

TABLE 14–4 IP Address Properties *(Continued)*

Property	Description	How to Specify in `pntadm` Command
IP address	The address you are working with, whether you are creating, modifying, or deleting the address. The IP address is displayed in the first column of the DHCP Manager's Addresses tab.	The IP address must accompany the `-A`, `-M`, and `-D` options to the `pntadm` command. For example, to modify IP address `10.21.5.12`, you would type: **`pntadm -M 10.21.5.12`** *options* **`10.21.0.0`**
Client name	The host name mapped to the IP address in the hosts table. This name can be automatically generated by DHCP Manager when addresses are created. If you create a single address, you can supply the name.	Specify the client name with the `-h` option. For example, to specify client name `carrot12` for `10.21.5.12`, you would type: **`pntadm -M 10.21.5.12 -h carrot12 10.21.0.0`**
Owned by server	The DHCP server that manages the IP address and responds to the DHCP client's request for IP address allocation.	Specify the owning server name with the `-s` option. For example to specify server `blue2` to own `10.21.5.12`, you would type: **`pntadm -M 10.21.5.12 -s blue2 10.21.0.0`**
Configuration macro	The macro that the DHCP server uses to obtain network configuration options from the `dhcptab` table. Several macros are created automatically when you configure a server, and when you add networks. See "About DHCP Macros" on page 239 for more information about macros. When addresses are created, a server macro is also created. The server macro is assigned as the configuration macro for each address.	Specify the macro name with the `-m` option. For example, to assign the server macro `blue2` to address `10.21.5.12`, you would type: **`pntadm -M 10.21.5.12 -m blue2 10.21.0.0`**
Client ID	A text string that is unique within the DHCP service. If the client ID is listed as 00, the address is not allocated to any client. If you specify a client ID when modifying the properties of an IP address, the address is bound exclusively to that client. The client ID is determined by the vendor of the DHCP client. If your client is not an DHCP client, consult your DHCP client documentation for more information.	Specify the client ID with the `-i` option. For example, to assign client ID `08002094121E` to address `10.21.5.12`, you would type: **`pntadm -M 10.21.5.12 -i 0108002094121E 10.21.0.0`**

TABLE 14–4 IP Address Properties *(Continued)*

Property	Description	How to Specify in `pntadm` Command
	For DHCP clients, the client ID is derived from the client's hexadecimal hardware address. The client ID includes a prefix that represents the ARP code for the type of network, such as 01 for Ethernet. The ARP codes are assigned by the Internet Assigned Numbers Authority (IANA) in the ARP Parameters section of the Assigned Numbers standard at `http://www.iana.com/numbers.html` For example, an Oracle Solaris client with the hexadecimal Ethernet address `8:0:20:94:12:1e` uses the client ID `0108002094121E`. The client ID is listed in DHCP Manager and `pntadm` when a client is currently using an address. **Tip:** As superuser on the Oracle Solaris client system, type the following command to obtain the Ethernet address for the interface: `dladm show-phys -m`	
Reserved	The setting that specifies the address is reserved exclusively for the client indicated by the client ID, and the DHCP server cannot reclaim the address. If you choose this option, you manually assign the address to the client.	Specify that the address is reserved, or manual, with the `-f` option. For example, to specify that IP address `10.21.5.12` is reserved for a client, you would type: **`pntadm -M 10.21.5.12 -f MANUAL 10.21.0.0`**
Lease type or policy	The setting that determines how DHCP manages the use of IP addresses by clients. A lease is either dynamic or permanent. See "Dynamic and Permanent Lease Types" on page 252 for a complete explanation.	Specify that the address is permanently assigned with the `-f` option. Addresses are dynamically leased by default. For example, to specify that IP address `10.21.5.12` has a permanent lease, you would type: **`pntadm -M 10.21.5.12 -f PERMANENT 10.21.0.0`**
Lease expiration date	The date when the lease expires, applicable only when a dynamic lease is specified. The date is specified in *mm/dd/yyyy* format.	Specify a lease expiration date with the `-e` option. For example, to specify an expiration date of January 1, 2006, you would type: **`pntadm -M 10.21.5.12 -e 01/01/2006 10.21.0.0`**
BOOTP setting	The setting that marks the address as reserved for BOOTP clients. See "Supporting BOOTP Clients With the DHCP Service (Task Map)" on page 295 for more information about supporting BOOTP clients.	Reserve an address for BOOTP clients with the `-f` option. For example, to reserve IP address `10.21.5.12` for BOOTP clients, you would type: **`pntadm -M 10.21.5.12 -f BOOTP 10.21.0.0`**

TABLE 14–4 IP Address Properties *(Continued)*

Property	Description	How to Specify in `pntadm` Command
Unusable setting	The setting that marks the address to prevent assignment of the address to any client.	Mark an address as unusable with the `-f` option. For example, to mark IP address `10.21.5.12` as unusable, you would type: `pntadm -M 10.21.5.12 -f UNUSABLE 10.21.0.0`

Adding IP Addresses to the DHCP Service

Before you add IP addresses, you must add the network that owns the addresses to the DHCP service. See "Adding DHCP Networks" on page 288 for information about adding networks.

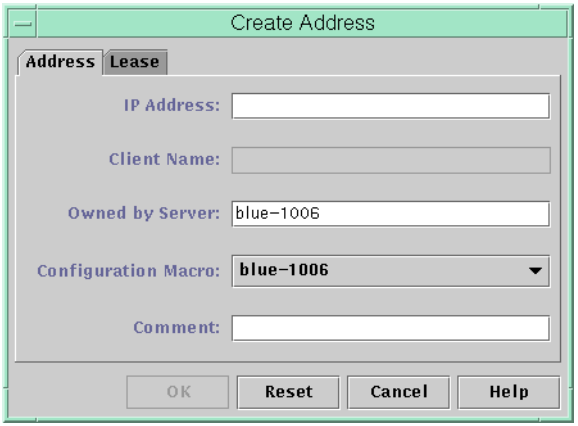
You can add addresses with DHCP Manager or the `pntadm` command.

On networks that are already managed by the DHCP service, you can add addresses in several ways with DHCP Manager:

- **Add a single IP address** – Place one new IP address under DHCP management.
- **Duplicate an existing IP address** – Copy the properties of an existing IP address managed by DHCP, and supply a new IP address and client name.
- **Add a range of multiple IP addresses** – Use the Address Wizard to place a series of IP addresses under DHCP management.

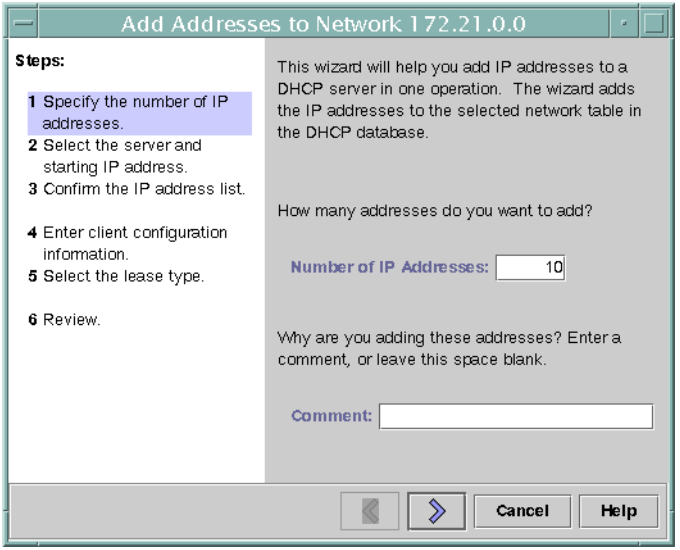
The following figure shows the Create Address dialog box. The Duplicate Address dialog box is identical to the Create Address dialog box, except that the text fields display the values for an existing address.

FIGURE 14–8 Create Address Dialog Box in DHCP Manager



The following figure shows the first dialog of the Add Addresses to Network wizard, used to add a range of IP addresses.

FIGURE 14–9 Add Addresses to Network Wizard in DHCP Manager



How to Add a Single IP Address (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Select the network where the new IP address is to be added.**

3. **Choose Create from the Edit menu.**

   The Create Address dialog box opens.

4. **Select or type values for the address settings on the Address and Lease tabs.**

   Select the Help button to open a web browser to display help for the dialog box. Also, see Table 14–4 for detailed information about the settings.

5. **Click OK.**

How to Duplicate an Existing IP Address (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Select the network where the new IP address is located.**

3. **Select the address with properties that you want to duplicate.**

4. **Choose Duplicate from the Edit menu.**

5. **Specify the new IP address in the IP Address field.**

6. **(Optional) Specify a new client name for the address.**

   You cannot use the same name that is used by the address that you are duplicating.

7. **(Optional) Modify other option values, if necessary.**

   Most other option values should remain the same.

8. **Click OK.**

▼ How to Add Multiple IP Addresses (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Select the network where the new IP addresses are to be added.**

3. **Choose Address Wizard from the Edit menu.**

   The Add Addresses to Network dialog box prompts you to provide values for the IP address properties. See Table 14–4 for more information about the properties, or select the Help button in the dialog box. "Making Decisions for IP Address Management (Task Map)" on page 250 includes more extensive information.

4. **Click the right arrow button as you finish each screen, and click Finish on the last screen.**

   The Addresses tab is updated with the new addresses.

▼ How to Add IP Addresses (`pntadm`)

- **Add IP addresses by typing a command of the following format:**

  `# pntadm -A` *ip-address options network-address*

  Refer to the `pntadm`(1M) man page for a list of options you can use with `pntadm -A`. In addition, Table 14–4 shows some sample `pntadm` commands that specify options.

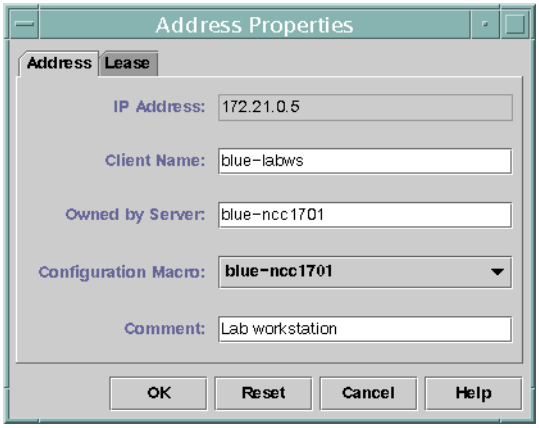
Note – You can write a script to add multiple addresses with `pntadm`. See Example 17–1 for an example.

Modifying IP Addresses in the DHCP Service

You can modify any of the address properties described in Table 14–4 by using DHCP Manager or the `pntadm -M` command. See the `pntadm`(1M) man page for more information about `pntadm -M`.

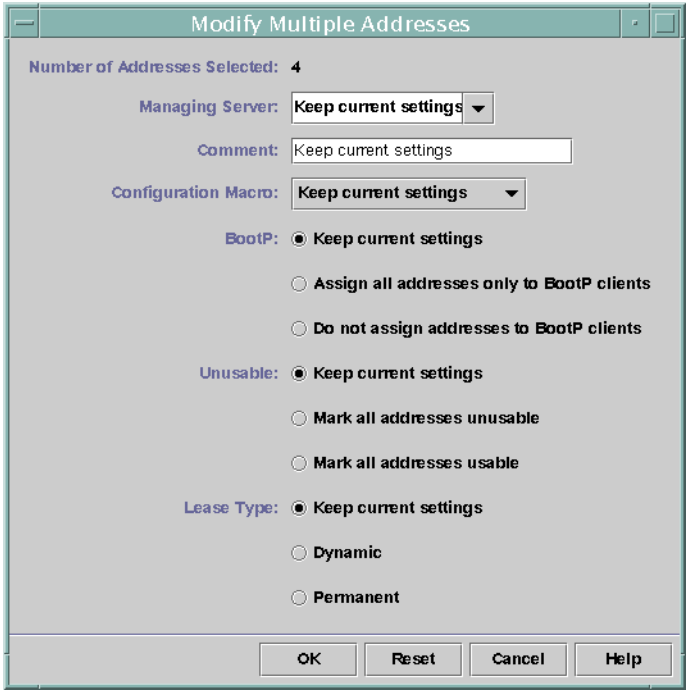
The following figure shows the Address Properties dialog box that you use to modify IP address properties.

FIGURE 14–10 Address Properties Dialog Box in DHCP Manager



The following figure shows the Modify Multiple Addresses dialog box that you use to modify multiple IP addresses.

FIGURE 14–11 Modify Multiple Addresses Dialog Box in DHCP Manager



▼ How to Modify IP Address Properties (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Select the IP address's network.**

3. **Select one or more IP addresses to modify.**

   If you want to modify more than one address, press the Control key while you click with the mouse to select multiple addresses. You can also press the Shift key while you click to select a block of addresses.

4. **Choose Properties from the Edit menu.**

   The Address Properties dialog box or the Modify Multiple Address dialog box opens.

5. **Change the appropriate properties.**

   Click the Help button, or refer to Table 14–4 for information about the properties.

6. **Click OK.**

▼ How to Modify IP Address Properties (`pntadm`)

- **Modify IP address properties by typing a command of the following format:**

  `# pntadm -M` *ip-address options network-address*

  Many options can be used with the `pntadm` command, which are documented in the `pntadm`(1M) man page.

  Table 14–4 shows some sample `pntadm` commands that specify options.

Removing IP Addresses From the DHCP Service

At times, you might want the DHCP service to stop managing a particular IP address or group of addresses. The method that you use to remove an address from DHCP depends on whether you want the change to be temporary or permanent.

- To temporarily prevent the use of addresses, you can mark the addresses as unusable in the Address Properties dialog box as described in "Marking IP Addresses as Unusable by the DHCP Service" on page 307.
- To permanently prevent the use of addresses by DHCP clients, delete the addresses from the DHCP network tables, as described in "Deleting IP Addresses From the DHCP Service" on page 308.

Marking IP Addresses as Unusable by the DHCP Service

You can use the `pntadm -M` command with the `-f UNUSABLE` option to mark addresses as unusable.

In DHCP Manager, you use the Address Properties dialog box, shown in Figure 14–10, to mark individual addresses. You use the Modify Multiple Addresses dialog box, show in Figure 14–11, to mark multiple addresses, as described in the following procedure.

▼ How to Mark IP Addresses as Unusable (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Select the IP address's network.**

3. **Select one or more IP addresses to mark as unusable.**

   If you want to mark more than one address as unusable, press the Control key while you click with the mouse to select multiple addresses. You can also press the Shift key while you click to select a block of addresses.

4. **Choose Properties from the Edit menu.**

   The Address Properties dialog box or the Modify Multiple Address dialog box opens.

5. **If you are modifying one address, select the Lease tab.**

6 **Select Address is Unusable.**

If you are editing multiple addresses, select Mark All Addresses Unusable.

7 **Click OK.**

▼ How to Mark IP Addresses as Unusable (`pntadm`)

- **Mark IP addresses as unusable by typing a command of the following format:**

  `# pntadm -M` *ip-address* `-f UNUSABLE` *network-address*

  For example, to mark address `10.64.3.3` as unusable, type:

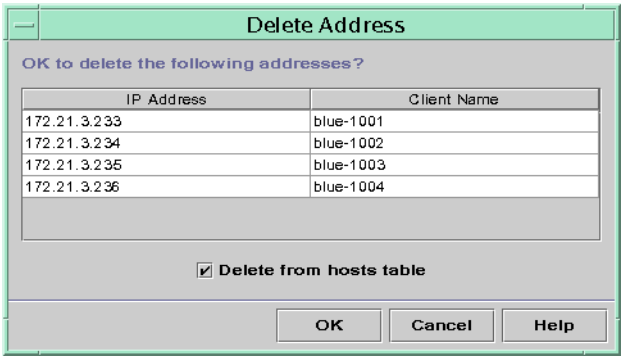
  ```
  pntadm -M 10.64.3.3 -f UNUSABLE 10.64.3.0
  ```

Deleting IP Addresses From the DHCP Service

You should delete IP addresses from the DHCP network tables if you no longer want the address to be managed by DHCP. You can use the `pntadm -D` command or DHCP Manager's Delete Address dialog box.

The following figure shows the Delete Address dialog box.

FIGURE 14–12 Delete Address Dialog Box in DHCP Manager



How to Delete IP Addresses From DHCP Service (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Select the IP address's network.**

3. **Select one or more IP addresses to delete.**

   If you want to delete more than one address, press the Control key while you click with the mouse to select multiple addresses. You can also press the Shift key while you click to select a block of addresses.

4. **Choose Delete from the Edit menu.**

   The Delete Address dialog box lists the address that you selected so that you can confirm the deletion.

5. **If you want to delete the host names from the hosts table, select Delete From Hosts Table.**

   If the host names were generated by DHCP Manager, you might want to delete the names from the hosts table.

6. **Click OK.**

How to Delete IP Addresses From the DHCP Service (`pntadm`)

- **Delete IP addresses by typing a command of the following format:**

  `# pntadm -D` *ip-address options network-address*

  If you include the `-y` option, the host name is deleted from the name service that maintains the host name.

  For example, to delete address `10.64.3.3` from network `10.64.3.0`, and delete the corresponding host name, type:

  ```
  pntadm -D 10.64.3.3 -y 10.64.3.0
  ```

Assigning a Reserved IP Address to a DHCP Client

The DHCP service attempts to provide the same IP address to a client that has previously obtained an address through DHCP. However, sometimes an address has already been reassigned to another client.

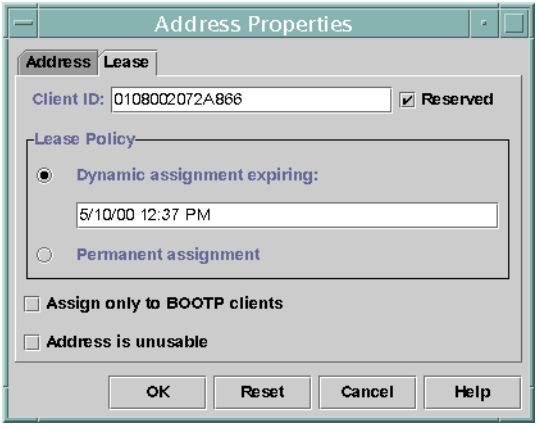
Routers, NIS servers, DNS servers, and other hosts that are critical to the network should not be DHCP clients. Hosts that provide services to the network should not rely on the network to obtain their IP addresses. Clients such as print servers or file servers should have consistent IP addresses as well. These clients can receive their network configurations and also be assigned a consistent IP address from the DHCP server.

You can set up the DHCP server to supply the same IP address to a client each time the client requests its configuration. You reserve the IP address for the client by manually assigning the client's ID to the address that you want the client to use. You can set up the reserved address to use either a dynamic lease or a permanent lease. If the client's address uses a dynamic lease, you can easily track the use of the address. If the client's address uses a permanent lease, you cannot track address use. Once a client obtains a permanent lease, the client does not contact the server again. The client can obtain updated configuration information only by releasing the IP address and restarting the DHCP lease negotiation.

You can use the `pntadm -M` command or DHCP Manager's Address Properties dialog box to set up lease properties.

The following figure shows the Lease tab of the Address Properties dialog box, which is used to modify the lease.

FIGURE 14–13 Address Properties Lease Tab in DHCP Manager



How to Assign a Consistent IP Address to a DHCP Client (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Select the appropriate network.**

3. **Double-click the IP address that you want to the client to use.**

   The Address Properties window opens.

4. **Select the Lease tab.**

5. **In the Client ID field, type the client ID.**

   The client ID is derived from the client's hardware address. See the Client ID entry in Table 14–4 for more information.

6. **Select the Reserved option to prevent the IP address from being reclaimed by the server.**

7. **In the Lease Policy area of the window, select Dynamic or Permanent assignment.**

   Select Dynamic if you want the client to negotiate to renew leases, which enables you to track when the address is used. Because you selected Reserved, the address cannot be reclaimed even when a dynamic lease is assigned. You do not need to specify an expiration date for this lease. The DHCP server calculates the expiration date by using the lease time.

   If you select Permanent, you cannot track the use of the IP address unless you enable transaction logging.

8. **Click OK.**

How to Assign a Consistent IP Address to a DHCP Client (`pntadm`)

- **Set the lease flags by typing a command of the following format:**

  `# pntadm -M` *ip-address* `-i` *client-id* `-f MANUAL+BOOTP` *network-address*

  For example, to enable the DHCP client whose MAC address is `08:00:20:94:12:1E` to always receive IP address `10.21.5.12`, you would type:

  ```
  pntadm -M 10.21.5.12 -i 0108002094121E -f MANUAL+BOOTP 10.21.0.0
  ```

Tip – Refer to the Client ID entry in Table 14–4 for more information about how to determine client identifiers.

Working With DHCP Macros (Task Map)

DHCP macros are containers of DHCP options. The DHCP service uses macros to gather options that should be passed to clients. DHCP Manager and the `dhcpconfig` utility create a number of macros automatically when you configure the server. See "About DHCP Macros" on page 239 for background information about macros. See Chapter 13, "Configuring the DHCP Service (Tasks)," for information about macros created by default.

You might find that when changes occur on your network, you need to make changes to the configuration information that is passed to clients. To change configuration information, you need to work with DHCP macros. You can view, create, modify, duplicate, and delete DHCP macros.

When you work with macros, you must know about DHCP standard options, which are described in the `dhcp_inittab(4)` man page.

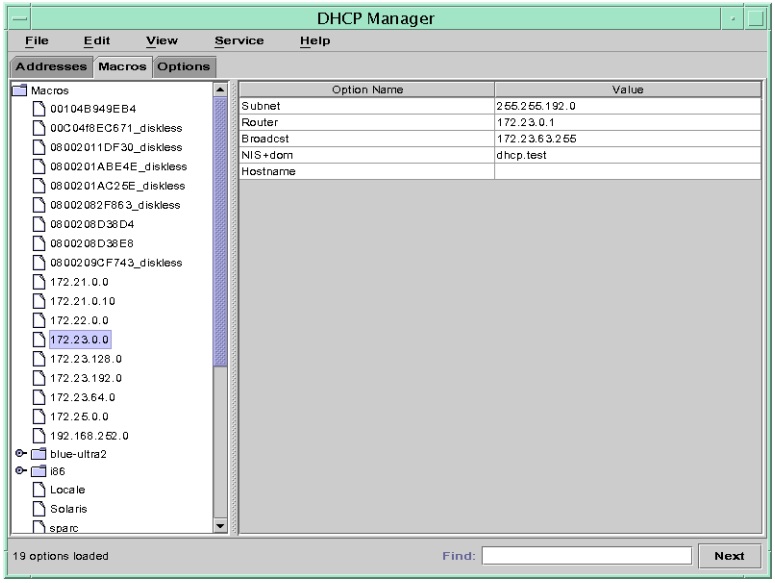
The following task map lists tasks to help you view, create, modify, and delete DHCP macros. The map also includes links to sections that detail how to accomplish each task.

Task	Description	For Instructions
View DHCP macros.	Display a list of all the macros that are defined on the DHCP server.	"How to View Macros Defined on a DHCP Server (DHCP Manager)" on page 314 "How to View Macros Defined on a DHCP Server (`dhtadm`)" on page 314
Create DHCP macros.	Create new macros to support DHCP clients.	"How to Create a DHCP Macro (DHCP Manager)" on page 319 "How to Create a DHCP Macro (`dhtadm`)" on page 320

Task	Description	For Instructions
Modify values that are passed in macros to DHCP clients.	Change macros by modifying existing options, adding options to macros, or removing options from macros.	"How to Change Values for Options in a DHCP Macro (DHCP Manager)" on page 315 "How to Change Values for Options in a DHCP Macro (`dhtadm`)" on page 316 "How to Add Options to a DHCP Macro (DHCP Manager)" on page 316 "How to Add Options to a DHCP Macro (`dhtadm`)" on page 317 "How to Delete Options From a DHCP Macro (DHCP Manager)" on page 317 "How to Delete Options From a DHCP Macro (`dhtadm`)" on page 318
Delete DHCP macros.	Remove DHCP macros that are no longer used.	"How to Delete a DHCP Macro (DHCP Manager)" on page 320 "How to Delete a DHCP Macro (`dhtadm`)" on page 321

The following figure shows the Macros tab in the DHCP Manager window.

FIGURE 14–14 DHCP Manager's Macros Tab



▼ How to View Macros Defined on a DHCP Server (DHCP Manager)

1. **In DHCP Manager, select the Macros tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

   The Macros area on the left side of the window displays, in alphabetical order, all the macros defined on the DHCP server. Macros preceded by a folder icon include references to other macros, whereas macros preceded by a document icon do not reference other macros.

2. **To open a macro folder, click the handle icon to the left of the folder icon.**

   The macros that are included in the selected macro are listed.

3. **To view the content of a macro, click the macro name.**

   Options and their assigned values are displayed.

▼ How to View Macros Defined on a DHCP Server (`dhtadm`)

- **Display the macros by typing the following command:**

  ```
  # dhtadm -P
  ```

  This command prints to standard output the formatted contents of the `dhcptab` table, including all macros and symbols defined on the DHCP server.

Modifying DHCP Macros

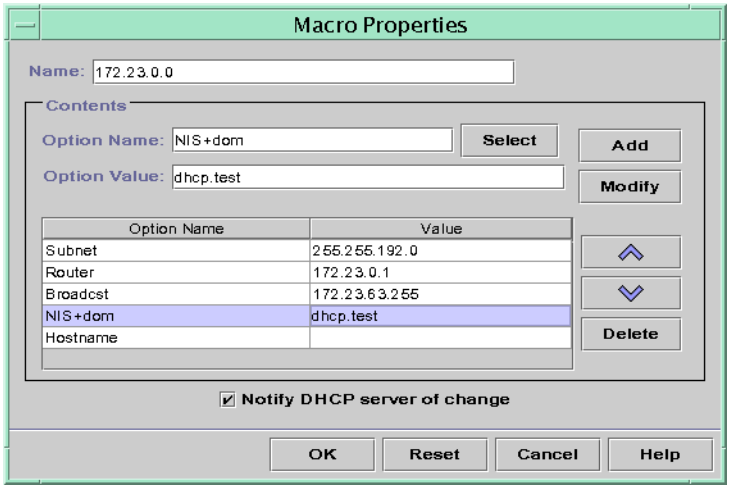
You might need to modify macros when some aspect of your network changes and one or more DHCP clients need to know about the change. For example, you might add a router or an NIS server, create a new subnet, or change the lease policy.

Before you modify a macro, determine the name of the DHCP option you want to change, add, or delete. The standard DHCP options are listed in the DHCP Manager help and in the `dhcp_inittab(4)` man page.

You can use the `dhtadm -M -m` command or DHCP Manager to modify macros. See the `dhtadm(1M)` man page for more information about `dhtadm`.

The following figure shows DHCP Manager's Macro Properties dialog box.

FIGURE 14–15 Macro Properties Dialog Box in DHCP Manager



How to Change Values for Options in a DHCP Macro (DHCP Manager)

1. **In DHCP Manager, select the Macros tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Select the macro that you want to change.**

3. **Choose Properties from the Edit menu.**

   The Macro Properties dialog box opens.

4. **In the table of Options, select the option that you want to change.**

   The option's name and its value are displayed in the Option Name and Option Value fields.

5. **In the Option Value field, select the old value and type the new value for the option.**

6. **Click Modify.**

   The new value is displayed in the options table.

7. **Select Notify DHCP Server of Change.**

   This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

8 **Click OK.**

How to Change Values for Options in a DHCP Macro (`dhtadm`)

- **Change option values by typing a command of the following format:**

  ```
  # dhtadm -M -m macroname -e 'option=value:option=value' -g
  ```

  For example, to change the lease time and the Universal Time Offset in the macro `bluenote`, you would type:

  ```
  # dhtadm -M -m bluenote -e 'LeaseTim=43200:UTCOffst=28800' -g
  ```

How to Add Options to a DHCP Macro (DHCP Manager)

1 **In DHCP Manager, select the Macros tab.**

  See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2 **Select the macro that you want to change.**

3 **Choose Properties from the Edit menu.**

  The Macro Properties dialog box opens.

4 **In the Option Name field, specify the name of an option by using one of the following methods:**

  - **Click the Select button next to the Option Name field to select an option to add to the macro.**

    The Select Option dialog box displays an alphabetized list of names of standard category options and descriptions. If you want to add an option that is not in the standard category, use the Category list to select a category.

    See "About DHCP Macros" on page 239 for more information about macro categories.

  - **Type `Include` if you want to include a reference to an existing macro in the new macro.**

5 **Type the value for the option in the Option Value field.**

  If you typed **`Include`** as the option name, you must specify the name of an existing macro in the Option Value field.

6. **Click Add.**

   The option is added to the bottom of the list of options in this macro. To change the option's position in the macro, select the option and click the arrow buttons to move the option up or down in the list.

7. **Select Notify DHCP Server of Change.**

   This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

8. **Click OK.**

How to Add Options to a DHCP Macro (`dhtadm`)

- **Add options to a macro by typing a command of the following format:**

  ```
  # dhtadm -M -m macroname -e 'option=value' -g
  ```

  For example, to add the ability to negotiate leases in the macro `bluenote`, you would type the following command:

  ```
  # dhtadm -M -m bluenote -e 'LeaseNeg=_NULL_VALUE' -g
  ```

  Note that if an option does not require a value, you must use `_NULL_VALUE` as the value for the option.

How to Delete Options From a DHCP Macro (DHCP Manager)

1. **In DHCP Manager, select the Macros tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Select the macro that you want to change.**

3. **Choose Properties from the Edit menu.**

   The Macro Properties dialog box opens.

4. **Select the option that you want to remove from the macro.**

5. **Click Delete.**

   The option is removed from the list of options for this macro.

6 **Select Notify DHCP Server of Change.**

This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

7 **Click OK.**

▼ How to Delete Options From a DHCP Macro (`dhtadm`)

- **Delete an option from a macro by typing a command of the following format:**

```
# dhtadm -M -m macroname -e 'option=' -g
```

For example, to remove the ability to negotiate leases in the macro `bluenote`, you would type the following command:

```
# dhtadm -M -m bluenote -e 'LeaseNeg=' -g
```

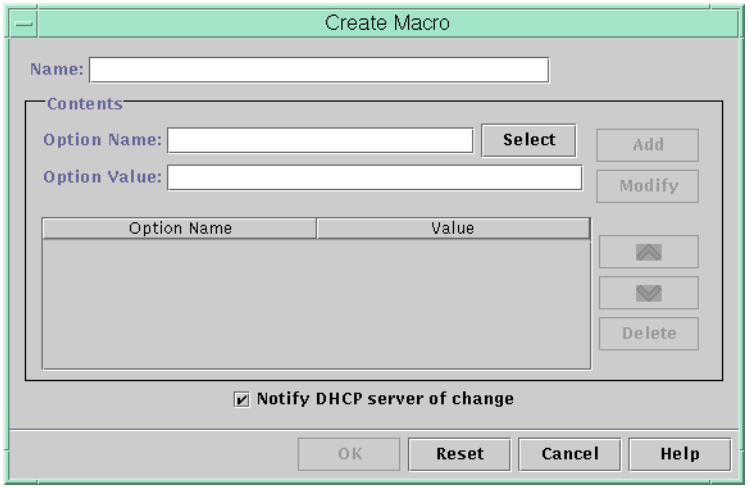
If an option is specified with no value, the option is removed from the macro.

Creating DHCP Macros

You might want to add new macros to your DHCP service to support clients with specific needs. You can use the `dhtadm -A -m` command or DHCP Manager's Create Macro dialog box to add macros. See the `dhtadm(1M)` man page for more information about the `dhtadm` command.

The following figure shows DHCP Manager's Create Macro dialog box.

FIGURE 14–16 Create Macro Dialog Box in DHCP Manager



How to Create a DHCP Macro (DHCP Manager)

1. **In DHCP Manager, select the Macros tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Choose Create from the Edit menu.**

   The Create Macro dialog box opens.

3. **Type a unique name for the macro.**

   The name can be up to 128 alphanumeric characters. If you use a name that matches a vendor class identifier, network address, or client ID, the macro is processed automatically for appropriate clients. If you use a different name, the macro is not processed automatically. The macro must be assigned to a specific IP address or included in another macro that is processed automatically. See "Macro Processing by the DHCP Server" on page 239 for more detailed information.

4. **Click the Select button, which is next to the Option Name field.**

   The Select Option dialog box displays an alphabetized list of names of standard category options and their descriptions. If you want to add an option that is not in the standard category, use the Category list. Select the category that you want from the Category list. See "About DHCP Options" on page 238 for more information about option categories.

5. **Select the option to add to the macro, and click OK.**

   The Macro Properties dialog box displays the selected option in the Option Name field.

6 **Type the value for the option in the Option Value field, and click Add.**

The option is added to the bottom of the list of options in this macro. To change the option's position in the macro, select the option and click the arrow buttons to move the option up or down in the list.

7 **Repeat Step 5 and Step 6 for each option you want to add to the macro.**

8 **Select Notify DHCP Server of Change when you are finished adding options.**

This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

9 **Click OK.**

▼ How to Create a DHCP Macro (`dhtadm`)

- **Create a macro by typing a command of the following format:**

```
# dhtadm -A -m macroname -d ':option=value:option=value:option=value:' -g
```

There is no limit to the number of *option=value* pairs that can be included in the argument to `-d`. The argument must begin and end with colons, with colons between each *option=value* pair. The complete string must be enclosed in quotation marks.

For example, to create the macro `bluenote`, type the following command:

```
# dhtadm -A -m bluenote -d ':Router=10.63.6.121\
:LeaseNeg=_NULL_VALUE:DNSserv=10.63.28.12:' -g
```

Note that if an option does not require a value, you must use `_NULL_VALUE` as the value for the option.

Deleting DHCP Macros

You might want to delete a macro from the DHCP service. For example, if you delete a network from the DHCP service, you can also delete the associated network macro.

You can use the `dhtadm -D -m` command or DHCP Manager to delete macros.

▼ How to Delete a DHCP Macro (DHCP Manager)

1 **In DHCP Manager, select the Macros tab.**

See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Select the macro to delete.**

   The Delete Macro dialog box prompts you to confirm that you want to delete the specified macro.

3. **Select Notify DHCP Server of Change.**

   This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

4. **Click OK.**

▼ How to Delete a DHCP Macro (`dhtadm`)

- **Delete a macro by typing a command of the following format:**

  ```
  # dhtadm -D -m macroname -g
  ```

  For example, to delete the macro `bluenote`, you would type the following command:

  ```
  # dhtadm -D -m bluenote -g
  ```

Working With DHCP Options (Task Map)

Options are keywords for network configuration parameters that the DHCP server can pass to clients. In the DHCP service, you cannot create, delete, or modify the standard DHCP options. The standard options are defined by the DHCP protocol, so the options cannot change. You can only perform tasks on options that you create for your site. For this reason, when you first set up your DHCP service, the Options tab in DHCP Manager is empty until you create options for your site.

If you create options on the DHCP server, you must also add information about the options on the DHCP client. For the DHCP client, you must edit the `/etc/dhcp/inittab` file to add entries for the new options. See the `dhcp_inittab(4)` man page for more information about this file.

If you have DHCP clients that are not Oracle Solaris clients, refer to the documentation for those clients for information about adding options or symbols. See "About DHCP Options" on page 238 for more information about options in DHCP.

You can use either DHCP Manager or the `dhtadm` command to create, modify, or delete options.

Tip – Options are called *symbols* in the DHCP literature. The `dhtadm` command and its related man page also refer to options as symbols.

The following task map lists tasks that you must perform to create, modify, and delete DHCP options. The task map contains links to procedures for the tasks.

Task	Description	For Instructions
Create DHCP options.	Add new options for information not covered by a standard DHCP option.	"How to Create DHCP Options (DHCP Manager)" on page 325 "How to Create DHCP Options (`dhtadm`)" on page 326 "Modifying the DHCP Client's Option Information" on page 329
Modify DHCP options.	Change properties of DHCP options you have created.	"How to Modify DHCP Option Properties (DHCP Manager)" on page 327 "How to Modify DHCP Option Properties (`dhtadm`)" on page 328
Delete DHCP options.	Remove DHCP options that you have created.	"How to Delete DHCP Options (DHCP Manager)" on page 329 "How to Delete DHCP Options (`dhtadm`)" on page 329

Before you create DHCP options, you should be familiar with the option properties listed in the following table.

TABLE 14–5 DHCP Option Properties

Option Property	Description
Category	The *category* of an option must be one of the following: ■ Vendor – Options specific to a client's vendor platform, either hardware or software. ■ Site – Options specific to your site. ■ Extend – Newer options that have been added to the DHCP protocol, but not yet implemented as standard options in DHCP.
Code	The *code* is a unique number that you assign to an option. The same code cannot be used for any other option within its option category. The code must be appropriate for the option category: ■ Vendor – Code values of 1–254 for each vendor class ■ Site – Code values of 128–254 ■ Extend – Code values of 77–127

TABLE 14–5 DHCP Option Properties *(Continued)*

Option Property	Description
Data type	The *data type* specifies what kind of data can be assigned as a value for the option. The valid data types are described in the following list. ■ ASCII – Text string value. ■ BOOLEAN – No value is associated with the Boolean data type. The presence of the option indicates that a condition is true, while the absence of the option indicates that a condition is false. For example, the `Hostname` option is Boolean. The presence of `Hostname` in a macro causes the DHCP server to look up the host name associated with the assigned address. ■ IP – One or more IP addresses, in dotted decimal format (*xxx.xxx.xxx.xxx*). ■ OCTET – Uninterpreted ASCII representation of binary data. For example, a client ID uses the octet data type. Valid characters are 0–9, A–F, and a–f. Two ASCII characters are needed to represent an 8-bit quantity. ■ UNUMBER8, UNUMBER16, UNUMBER32, UNUMBER64, SNUMBER8, SNUMBER16, SNUMBER32, or SNUMBER64 – Numeric value. An initial U or S indicates whether the number is unsigned or signed. The digits at the end indicate how many bits are in the number.
Granularity	The *granularity* specifies how many "instances" of the data type are needed to represent a complete option value. For example, a data type of IP and a granularity of 2 would mean that the option value must contain two IP addresses.
Maximum	The maximum number of values that can be specified for the option. For example, suppose the maximum is 2, the granularity is 2, and the data type is IP. In this case, the option value could contain a maximum of two pairs of IP addresses.

TABLE 14–5 DHCP Option Properties *(Continued)*

Option Property	Description
Vendor client classes	This option is available only when the option category is Vendor. Vendor client classes identify the client classes with which the Vendor option is associated. The class is an ASCII string that represents the client machine type or operating system. For example, the class string for some models of Sun workstations is `SUNW.Sun-Blade-100`. This type of option enables you to define configuration parameters that are passed to all clients of the same class, and *only* clients of that class. You can specify multiple client classes. Only those DHCP clients with a client class value that matches a class that you specify receive the options scoped by that class. The client class is determined by the vendor of the DHCP client. For DHCP clients that are not Oracle Solaris clients, refer to the vendor documentation for the DHCP client for the client class. For Oracle Solaris clients, the Vendor client class can be obtained by typing the `uname -i` command on the client. To specify the Vendor client class, substitute periods for any commas in the string returned by the `uname` command. For example, if the string `SUNW,Sun-Blade-100` is returned by the `uname -i` command, you should specify the Vendor client class as `SUNW.Sun-Blade-100`.

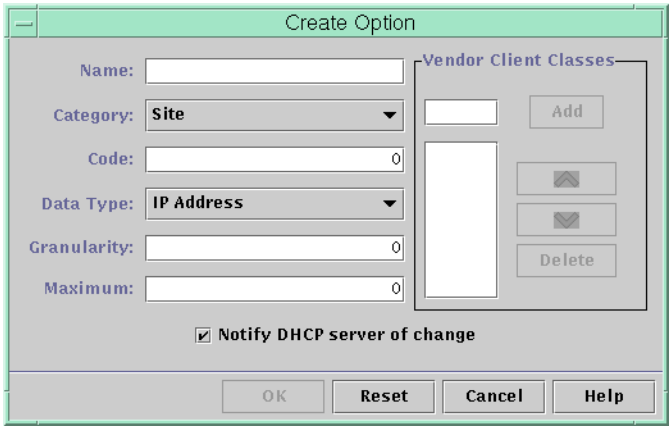
Creating DHCP Options

If you need to pass client information for which there is not already an existing option in the DHCP protocol, you can create an option. See the `dhcp_inittab(4)` man page for a list of all the options that are defined in DHCP before you create your own option.

You can use the `dhtadm -A -s` command or DHCP Manager's Create Option dialog box to create new options.

The following figure shows DHCP Manager's Create Option dialog box.

FIGURE 14–17 Create Option Dialog Box in DHCP Manager



How to Create DHCP Options (DHCP Manager)

1. **In DHCP Manager, select the Options tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Choose Create from the Edit menu.**

   The Create Options dialog box opens.

3. **Type a short descriptive name for the new option.**

   The name can contain up to 128 alphanumeric characters and spaces.

4. **Type or select values for each setting in the dialog box.**

   Refer to Table 14–5 for information about each setting, or view the DHCP Manager help.

5. **Select Notify DHCP Server of Change if you are finished creating options.**

   This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

6. **Click OK.**

   You can now add the option to macros, and assign a value to the option to pass to clients.

▼ How to Create DHCP Options (`dhtadm`)

- **Create a DHCP option by typing a command using the following format:**

```
# dhtadm -A -s option-name -d 'category,code,data-type,granularity,maximum' -g
```

option-name Is an alphanumeric string of 128 characters of less.

category Is one of the following: `Site`, `Extend`, or `Vendor=`*list-of-classes*. *list-of-classes* is a space-separated list of vendor client classes to which the option applies. See Table 14–5 for information about how to determine the vendor client class.

code Is a numeric value that is appropriate to the option category, as explained in Table 14–5.

data-type Is specified by a keyword that indicates the type of data that is passed with the option, as explained in Table 14–5.

granularity Is specified as a nonnegative number, as explained in Table 14–5.

maximum Is a nonnegative number, as explained in Table 14–5.

Example 14–3 Creating a DHCP Option With `dhtadm`

The following command would create an option called `NewOpt`, which is a Site category option. The option's code is 130. The option's value can be set to a single 8-bit unsigned integer.

```
# dhtadm -A -s NewOpt -d 'Site,130,UNUMBER8,1,1' -g
```

The following command would create an option called `NewServ`, which is a Vendor category option that applies to clients whose machine type is `SUNW,Sun-Blade-100` or `SUNW,Sun-Blade-1000`. The option's code is 200. The option's value can be set to one IP address.

```
# dhtadm -A -s NewServ -d 'Vendor=SUNW.Sun-Blade-100 \
SUNW.Sun-Blade-1000,200,IP,1,1' -g
```

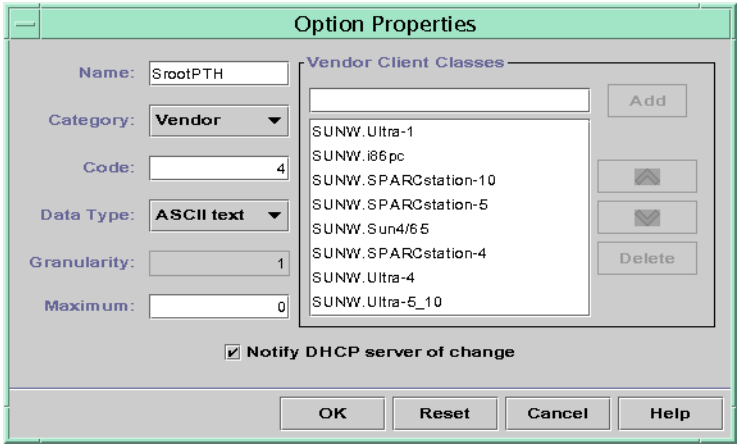
Modifying DHCP Options

If you have created options for your DHCP service, you can change the properties for these options. You can use the `dhtadm -M -s` command or DHCP Manager's Option Properties dialog box to modify options.

Note that you should modify the DHCP client's option information to reflect the same modification that you make to the DHCP service. See "Modifying the DHCP Client's Option Information" on page 329.

The following figure shows DHCP Manager's Option Properties dialog box.

FIGURE 14–18 Option Properties Dialog Box in DHCP Manager



How to Modify DHCP Option Properties (DHCP Manager)

1. **In DHCP Manager, select the Options tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Select the option that you want to modify.**

3. **Choose Properties from the Edit menu.**

   The Option Properties dialog box opens.

4. **Edit the properties as needed.**

   See Table 14–5 for information about the properties, or view the DHCP Manager help.

5. **Select Notify DHCP Server of Change when you are finished with options.**

   The change is made to the `dhcptab` table. The DHCP server is signaled to reread the `dhcptab` table to put the changes into effect.

6. **Click OK.**

▼ How to Modify DHCP Option Properties (`dhtadm`)

- **Modify an option by typing a command using the following format:**

```
# dhtadm -M -s option-name -d 'category,code,data-type,granularity,maximum' -g
```

option-name	Specifies the name of the option that you want to change.
category	Can be `Site`, `Extend`, or `Vendor=`*list-of-classes*. *list-of-classes* is a space-separated list of vendor client classes to which the option applies. For example, `SUNW.Sun-Blade-100 SUNW.Ultra-80 SUNWi86pc`.
code	Specifies a numeric value that is appropriate to the option category, as explained in Table 14–5.
data-type	Specifies a keyword that indicates the type of data that is passed with the option, as explained in Table 14–5.
granularity	Is a nonnegative number, as explained in Table 14–5.
maximum	Is a nonnegative number, as explained in as explained in Table 14–5.

Note that you must specify all of the DHCP option properties with the `-d` switch, not just the properties that you want to change.

Example 14–4 Modifying a DHCP Option With `dhtadm`

The following command would modify an option called `NewOpt`. The option is a Site category option. The option's code is 135. The option's value can be set to a single 8-bit unsigned integer.

```
# dhtadm -M -s NewOpt -d 'Site,135,UNUMBER8,1,1'
```

The following command would modify an option called `NewServ`, which is a Vendor category option. The option now applies to clients whose machine type is `SUNW,Sun-Blade-100` or `SUNW,i86pc`. The option's code is 200. The option's value can be set to one IP address.

```
# dhtadm -M -s NewServ -d 'Vendor=SUNW.Sun-Blade-100 \
SUNW.i86pc,200,IP,1,1' -g
```

Deleting DHCP Options

You cannot delete standard DHCP options. However, if you have defined options for your DHCP service, you can delete these options by using DHCP Manager or the `dhtadm` command.

▼ How to Delete DHCP Options (DHCP Manager)

1. **In DHCP Manager, select the Options tab.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Select the option that you want to delete.**

3. **Choose Delete from the Edit menu.**

   The Delete Option dialog box opens.

4. **Select Notify DHCP Server of Change if you are finished deleting options.**

   This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

5. **Click OK.**

▼ How to Delete DHCP Options (`dhtadm`)

- **Delete a DHCP option by typing a command using the following format:**

  ```
  # dhtadm -D -s option-name -g
  ```

Modifying the DHCP Client's Option Information

If you add a new DHCP option to your DHCP server, you must add a complementary entry to each DHCP client's option information. If you have a DHCP client that is not a DHCP client, refer to that client's documentation for information about adding options or symbols.

On an DHCP client, you must edit the `/etc/dhcp/inittab` file and add an entry for each option that you add to the DHCP server. If you later modify the option on the server, you must also modify the entry in the client's `/etc/dhcp/inittab` file.

Refer to the `dhcp_inittab(4)` man page for detailed information about the syntax of the `/etc/dhcp/inittab` file.

Note – If you added DHCP options to the `dhcptags` file in a previous Oracle Solaris release, you must add the options to the `/etc/dhcp/inittab` file. See "DHCP Option Information" on page 385 for more information.

Supporting Oracle Solaris Network Installation With the DHCP Service

You can use DHCP to install Oracle Solaris on certain client systems on your network. Only `sun4u`-based systems and x86 systems that meet the hardware requirements for running Oracle Solaris can use this feature. For information about using DHCP to automatically configure client systems for the network as they boot, see *Oracle Solaris 11 Express Automated Installer Guide*.

DHCP also supports Oracle Solaris client systems that boot and install remotely from servers across a wide area network (WAN) using HTTP. This method of remote booting and installing is called the *WAN boot installation* method. Using WAN boot, you can install Oracle Solaris on SPARC based systems over a large public network where the network infrastructure might be untrustworthy. You can use WAN boot with security features to protect data confidentiality and installation image integrity.

Before you can use DHCP for booting and installing client systems remotely by using WAN boot, the DHCP server must be configured to supply the following information to clients:

- The proxy server's IP address
- The location of the `wanboot-cgi` program

For further details, see *Oracle Solaris 11 Express Automated Installer Guide*

Setting Up DHCP Clients to Receive Information Only (Task Map)

In some networks, you might want the DHCP service to provide only configuration information to clients. Client systems that need information, not leases, can use the DHCP client to issue an `INFORM` message. The `INFORM` message asks the DHCP server to send the appropriate configuration information to the client.

You can set up the DHCP server to support clients that need information only. You need to create an empty network table that corresponds to the network that is hosting the clients. The table must exist so that the DHCP server can respond to clients from that network.

The following task map lists the tasks required to support information-only clients. The task map also includes links to procedures to help you carry out the tasks.

Task	Description	For Instructions
Create an empty network table.	Use DHCP Manager or the `pntadm` command to create a network table for the information-only clients' network.	"Adding DHCP Networks" on page 288
Create macros to contain information that is needed by clients.	Use DHCP Manager or the `dhtadm` command to create macros to pass the required information to clients.	"Creating DHCP Macros" on page 318
Have the DHCP client issue an `INFORM` message.	Use the `ipadm` command to make the DHCP client issue an `INFORM` message.	"DHCP Client Startup" on page 345 "`ipadm` Command Options Used With the DHCP Client" on page 350 `ipadm`(1M)

Converting to a New DHCP Data Store

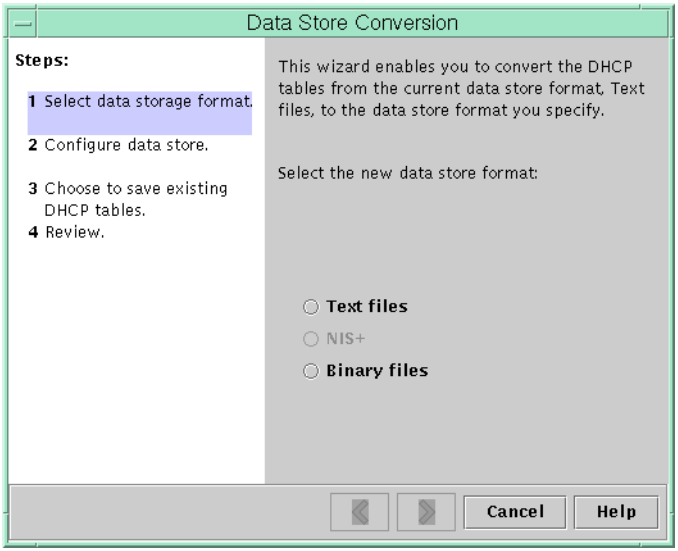
DHCP provides a utility to convert the DHCP configuration data from one data store to another data store. Several reasons might exist for converting to a new data store. For example, you might have more DHCP clients, requiring higher performance or higher capacity from the DHCP service. You also might want to share the DHCP server duties among multiple servers. See "Choosing the DHCP Data Store" on page 248 for a comparison of the relative benefits and drawbacks of each type of data store.

The conversion utility is also useful for sites that are converting from a Sun provided data store to a third-party data store. The conversion utility looks up entries in the existing data store and adds new entries that contain the same data to the new data store. Data store access is implemented in separate modules for each data store. This modular approach enables the conversion utility to convert DHCP data from any data store format to any other data store format. Each data store must have a module that the DHCP service can use.

The data store conversion can be accomplished with DHCP Manager through the Data Store Conversion wizard, or with the `dhcpconfig -C` command.

The initial dialog box of the Data Store Conversion wizard is shown in the following figure.

FIGURE 14–19 Data Store Conversion Wizard Dialog Box in DHCP Manager



Before the conversion begins, you must specify whether to save the old data store's tables (`dhcptab` and network tables). The conversion utility then stops the DHCP server, converts the data store, and restarts the server when the conversion has completed successfully. If you did not specify to save the old tables, the utility deletes the tables after determining the conversion is successful. The process of converting can be time-consuming. The conversion runs in the background with a meter to inform you of its progress.

▼ How to Convert the DHCP Data Store (DHCP Manager)

1. **In DHCP Manager, choose Convert Data Store from the Service menu.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

   The Data Store Conversion wizard opens.

2. **Answer the wizard's prompts.**

   If you have trouble providing the requested information, click Help to view detailed information about each dialog box.

3. **Review your selections, and then click Finish to convert the data store.**

   The DHCP server restarts when the conversion is complete. The server immediately uses the new data store.

▼ How to Convert the DHCP Data Store (`dhcpconfig -C`)

- **Convert the data store by typing a command of the following format:**

  ```
  # /usr/sbin/dhcpconfig -C -r resource -p path
  ```

  resource is the new data store type, such as `SUNWbinfiles`

  path is the path to the data, such as `/var/dhcp`

  Note that if you want to keep the original data in the old data store after the conversion, specify the `-k` option. For example, to convert your data store to `SUNWbinfiles` and save the old data store, you would type:

  ```
  # /usr/sbin/dhcpconfig -C -r SUNWbinfiles -p /var/dhcp -k
  ```

  See the `dhcpconfig(1M)` man page for more information about the `dhcpconfig` utility.

Moving Configuration Data Between DHCP Servers (Task Map)

DHCP Manager and the `dhcpconfig` utility enable you to move some or all the DHCP configuration data from one DHCP server to another server. You can move entire networks and all the IP addresses, macros, and options associated with the networks. Alternatively, you can select specific IP addresses, macros, and options to move. You can also copy macros and options without removing the macros and options from the first server.

You might want to move data if you are going to do any of the following tasks:

- Add a server to share DHCP duties.
- Replace the DHCP server's system.
- Change the path for the data store, while still using the same data store.

The following task map identifies the procedures that you must perform when you move DHCP configuration data. The map includes links to procedures to perform the tasks.

Task	Description	For Instructions
1. Export the data from the first server.	Select the data that you want to move to another server, and create a file of exported data.	"How to Export Data From a DHCP Server (DHCP Manager)" on page 335 "How to Export Data From a DHCP Server (`dhcpconfig -X`)" on page 336

Task	Description	For Instructions
2. Import the data to the second server.	Copy exported data to another DHCP server's data store.	"How to Import Data on a DHCP Server (DHCP Manager)" on page 337 "How to Import Data on a DHCP Server (`dhcpconfig -I`)" on page 337
3. Modify the imported data for the new server environment.	Change server-specific configuration data to match the new server's information.	"How to Modify Imported DHCP Data (DHCP Manager)" on page 337 "How to Modify Imported DHCP Data (`pntadm`, `dhtadm`)" on page 338

In DHCP Manager, you use the Export Data wizard and the Import Data wizard to move the data from one server to the other server. You then modify macros in the Macros tab. The following figures show the initial dialog boxes for the wizards.

FIGURE 14–20 Export Data Wizard Dialog Box in DHCP Manager

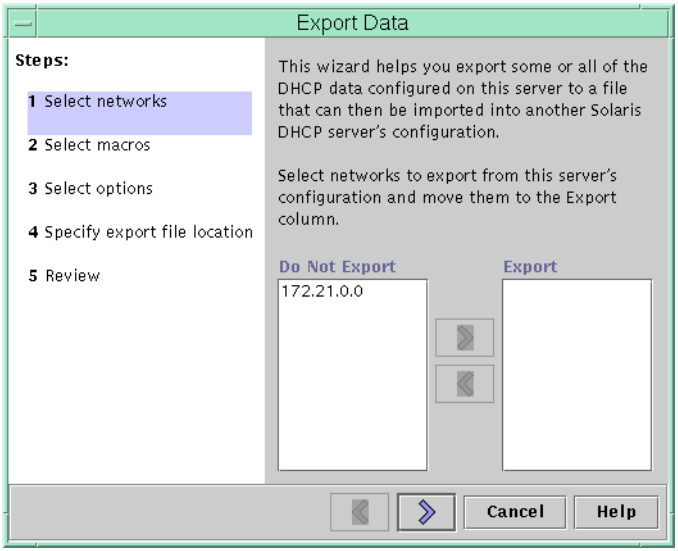
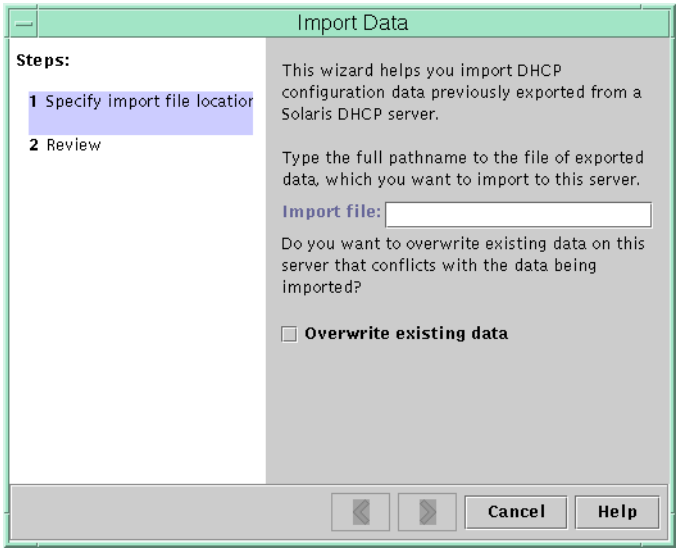


FIGURE 14–21 Import Data Wizard Dialog Box in DHCP Manager



▼ How to Export Data From a DHCP Server (DHCP Manager)

1. **Start DHCP Manager on the server from which you want to move or copy data.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Choose Export Data from the Service menu.**

   The Export Data wizard opens as shown in Figure 14–20.

3. **Answer the wizard's prompts.**

   If you have difficulty, click Help for detailed information about the prompts.

4. **Move the export file to a file system that is accessible to the DHCP server that must import the data.**

See Also Import the data as described in "How to Import Data on a DHCP Server (DHCP Manager)" on page 337.

▼ How to Export Data From a DHCP Server (`dhcpconfig -X`)

1 **Log in to the server from which you want to move or copy data.**

2 **Export the data.**

You can export all of the DHCP data, or specific parts of the data.

- **To export specific addresses, macros, and options, type a command that uses the following format:**

  `# dhcpconfig -X` *filename* `-a` *network-addresses* `-m` *macros* `-o` *options*

  filename is the full path name that you want to use to store the compressed exported data. You specify particular network addresses, DHCP macros, and DHCP options in comma-separated lists. The following example shows how to export specific networks, macros, and options.

```
# dhcpconfig -X /var/dhcp/0dhcp1065_data \
-a 10.63.0.0,10.62.0.0 \
-m 10.63.0.0,10.62.0.0,SUNW.Sun-Blade-100 -o Sterm
```

- **To export all DHCP data, type a command that uses the `ALL` keyword.**

  `# dhcpconfig -X` *filename* `-a ALL -m ALL -o ALL`

  filename is the full path name that you want to use to store the compressed exported data. The keyword `ALL` can be used with the command options to export all the network addresses, macros, or options. The following example shows how to use the `ALL` keyword.

```
# dhcpconfig -X /var/dhcp/dhcp1065_data -a ALL -m ALL -o ALL
```

Tip – You can omit the export of a particular kind of data by not specifying the `dhcpconfig` command option for that type of data. For example, if you do not specify the `-m` option, no DHCP macros are exported.

See the `dhcpconfig(1M)` man page for more information about the `dhcpconfig` command.

3 **Move the export file to a location that is accessible to the server that must import the data.**

See Also Import the data as described in "How to Import Data on a DHCP Server (`dhcpconfig -I`)" on page 337.

How to Import Data on a DHCP Server (DHCP Manager)

1. **Start DHCP Manager on the server to which you want to move data that you previously exported from a DHCP server.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Choose Import Data from the Service menu.**

   The Import Data wizard opens, as shown in Figure 14–21.

3. **Answer the wizard's prompts.**

   If you have difficulty, click Help for detailed information about the prompts.

4. **Modify the imported data, if necessary.**

   See "How to Modify Imported DHCP Data (DHCP Manager)" on page 337

How to Import Data on a DHCP Server (`dhcpconfig -I`)

1. **Log in to the server to which you want to import the data.**

2. **Import the data by typing a command of the following format:**

   `# dhcpconfig -I` *filename*

   filename is the name of the file that contains the exported data.

3. **Modify the imported data, if necessary.**

   See "How to Modify Imported DHCP Data (`pntadm`, `dhtadm`)" on page 338.

How to Modify Imported DHCP Data (DHCP Manager)

1. **Start DHCP Manager on the server to which you imported data.**

   See "How to Start and Stop DHCP Manager" on page 270 for information about DHCP Manager.

2. **Examine imported data for network-specific information that needs modification.**

   For example, if you moved networks, you must open the Addresses tab and change the owning server of addresses in the imported networks. You might also need to open the Macros tab to specify the correct domain names for NIS or DNS in some macros.

3. **Open the Addresses, tab and select a network that you imported.**

4. **To select all the addresses, click the first address, press and hold the Shift key, and click the last address.**

5. **From the Edit menu, choose Properties.**
   The Modify Multiple Addresses dialog box opens.

6. **At the Managing Server prompt, select the new server's name.**

7. **At the Configuration Macro prompt, select the macro that should be used for all clients on this network, and then click OK.**

8. **Open the Macros tab.**

9. **Use the Find button to locate the options that are likely to need modified values.**
   The Find button is located at the bottom of the window.

   `DNSdmain`, `DNSserv`, `NISservs`, and `NISdmain` are examples of options that might need modification on the new server.

10. **Change the options in the appropriate macros.**
    See "How to Modify DHCP Option Properties (DHCP Manager)" on page 327 for the procedure for changing options.

▼ How to Modify Imported DHCP Data (`pntadm`, `dhtadm`)

1. **Log in to the server to which you imported data.**

2. **Examine the network tables for data that needs to be modified.**
   If you moved networks, use the `pntadm -P` *network-address* command to print out the network tables for the networks you moved.

3. **Modify IP address information by using the `pntadm` command.**
   You might need to change the owning server and the configuration macro for imported addresses. For example, to change the owning server (`10.60.3.4`) and macro (`dhcpsrv-1060`) for address `10.63.0.2`, you would use the following command:

   ```
   pntadm -M 10.63.0.2 -s 10.60.3.4 -m dhcpsrv-1060 10.60.0.0
   ```

   If you have a large number of addresses, you should create a script file that contains commands to modify each address. Execute the script with the `pntadm -B` command, which runs `pntadm` in batch mode. See the `pntadm`(1M) man page.

4 Examine the `dhcptab` macros for options with values that need modification.

Use the `dhtadm -P` command to print the entire `dhcptab` table to your screen. Use `grep` or some other tool to search for options or values that you might want to change.

5 Modify options in macros, if necessary, by using the `dhtadm -M` command.

For example, you might need to modify some macros to specify the correct domain names and servers for NIS or DNS. For example, the following command changes the values of `DNSdmain` and `DNSserv` in the macro `mymacro`:

```
dhtadm -M -m mymacro -e 'DNSserv=dnssrv2:DNSdmain=example.net' -g
```

◆ ◆ ◆ CHAPTER 15

Configuring and Administering the DHCP Client

This chapter discusses the Dynamic Host Configuration Protocol (DHCP) client that is part of Oracle Solaris. The chapter explains how the client's DHCPv4 and DHCPv6 protocols work, and how you can affect the behavior of the client.

One protocol, DHCPv4, has long been part of Oracle Solaris, and enables DHCP servers to pass configuration parameters such as IPv4 network addresses to IPv4 nodes.

The other protocol, DHCPv6, enables DHCP servers to pass configuration parameters such as IPv6 network addresses to IPv6 nodes. DHCPv6 is a stateful counterpart to "IPv6 Stateless Address Autoconfiguration" (RFC 2462), and can be used separately or concurrently with the stateless to obtain configuration parameters.

This chapter contains the following information:

About the DHCP Client

The DHCP client is the `dhcpagent` daemon, part of Oracle Solaris. When you install Oracle Solaris, you are prompted to use DHCP to configure network interfaces. If you specify Yes for DHCPv4, then that protocol is enabled on your system during Oracle Solaris installation. There are no install time options specifically for DHCPv6. A related question, though, is about IPv6. If you enable IPv6, then DHCPv6 is also enabled on a local network that supports DHCPv6.

You do not need to do anything else with the Oracle Solaris client to use DHCP. The DHCP server's configuration determines what information is given to DHCP client systems that use the DHCP service.

If a client system is already running Oracle Solaris, but not using DHCP, you can reconfigure the client system to use DHCP. You can also reconfigure a DHCP client system so that it stops using DHCP and uses static network information that you provide. See "Enabling and Disabling a DHCP Client" on page 349 for more information.

DHCPv6 Server

There is no DHCPv6 server available through Sun Microsystems for Oracle Solaris. Servers available from third parties are compatible with Sun's DHCPv6, and if there is a DHCPv6 server on the network, Sun's DHCPv6 client will use it.

See "The DHCP Server" on page 232for information on the Sun DHCPv4 server.

Differences Between DHCPv4 and DHCPv6

The two major differences between DHCPv4 and DHCPv6 are the following:

- **The administrative model**
  - DHCPv4–The administrator enables DHCP for each interface. Administration is on a per-logical interface basis.
  - DHCPv6–Explicit configuration is not necessary. This protocol is enabled on a given physical interface.
- **Protocol details**
  - DHCPv4–The DHCP server supplies the subnet mask for each address. A hostname option sets the system-wide node name.
  - DHCPv6–The subnet mask is supplied by Router Advertisements, not the DHCPv6 server. There is no DHCPv6 hostname option.

The Administrative Model

DHCPv4 requires explicit client configuration. You must set up the DHCPv4 system for addressing when desired, and this is typically done during initial system installation or dynamically through the use of the `ipadm` command. See the `ipadm(1M)` man page.

DHCPv6 does not require explicit client configuration. Instead, using DHCP is a property of the network, and the signal to use it is carried in Router Advertisement messages from local routers. The DHCP client automatically creates and destroys logical interfaces as needed.

The DHCPv6 mechanism is very similar administratively to the existing IPv6 stateless (automatic) address configuration. For stateless address configuration, you would set a flag on the local router to indicate that, for a given set of prefixes, each client should automatically configure an address on its own by using the advertised prefix plus a local interface token or random number. For DHCPv6, the same prefixes are required, but the addresses are acquired and managed through a DHCPv6 server instead of being assigned "randomly."

MAC Address and Client ID

DHCPv4 uses the MAC address and an optional Client ID to identify the client for purposes of assigning an address. Each time the same client arrives on the network, it gets the same address, if possible.

DHCPv6 uses basically the same scheme, but makes the Client ID mandatory and imposes structure on it. The Client ID in DHCPv6 consists of two parts: a DHCP Unique Identifier (DUID) and an Identity Association Identifier (IAID). The DUID identifies the client **system** (rather than just an interface, as in DHCPv4), and the IAID identifies the interface on that system.

As described in RFC 3315, an identity association is the means used for a server and a client to identify, group, and manage a set of related IPv6 addresses. A client must associate at least one distinct IA with each of its network interfaces, and then uses the assigned IAs to obtain configuration information from a server for that interface. For additional information about IAs, see the next section, "Protocol Details."

DUID+IAID can also be used with DHCPv4. These can be concatenated together unambiguously so that they can serve as the Client ID. For compatibility reasons, this is not done for regular IPv4 interfaces. However, for logical interfaces (`bge0:1`), DUID+IAID is used if no Client ID is configured.

Unlike IPv4 DHCP, DHCPv6 does not provide a "client name" option, so there is no way to name your systems based on DHCPv6 alone. Instead, if you need to know the DNS name that goes with an address provided by DHCPv6, use DNS reverse-resolution (address-to-name query via the `getaddrinfo(3SOCKET)` function) to find the corresponding name information. One implication of this is that if you are using only DHCPv6 and want a node to have a specific name, you must specify the node name by using the `svccfg` command as follows:.

```
# svccfg -s svc:/system/identity:node setprop config/nodename = astring: hostname
```

Protocol Details

With DHCPv4, the DHCP server supplies the subnet mask to be used with the assigned address. With DHCPv6, the subnet mask (also known as "prefix length") is assigned by the Router Advertisements, and is not controlled by the DHCP server.

DHCPv4 carries a Hostname option that is used to set the system-wide node name. DHCPv6 has no such option.

To configure a Client ID for DHCPv6 you must specify a DUID, rather than allowing the system to choose one automatically. You can do this globally for the daemon, or on a per-interface basis. Use the following format to set the global DUID (note the initial dot):

.v6.CLIENT_ID=*DUID*

To set a particular interface to use a given DUID (and make the system appear to be multiple independent clients to a DHCPv6 server):

bge0.v6 CLIENT ID=*DUID*

Each Identity Association (IA) holds one type of address. For example, an identity association for temporary addresses (IA_TA) holds temporary addresses, while an identity association for non-temporary addresses (IA_NA), carries assigned addresses that are permanent. The version of DHCPv6 described in this guide provides only IA_NA associations.

Oracle Solaris assigns exactly one IAID to each interface, on demand, and the IAID is stored in a file in the root file system so that it remains constant for the life of the machine.

Logical Interfaces

In the DHCPv4 client, each logical interface is independent and is an administrative unit. In addition to the zeroth logical interface (which defaults to the interface MAC address as an identifier), the user may configure specific logical interfaces to run DHCP by specifying a CLIENT_ID in the `dhcpagent` configuration file. For example:

```
hme0:1.CLIENT_ID=orangutan
```

DHCPv6 works differently. The zeroth logical interface on an IPv6 interface, unlike IPv4, is always a link-local. A link-local is used to automatically assign an IP address to a device in an IP network when there is no other assignment method available, such as a DHCP server. The zeroth logical interface cannot be under DHCP control, so although DHCPv6 is run on the zeroth logical interface (known, also, as the "physical" interface), it assigns addresses only on non-zero logical interfaces.

In response to a DHCPv6 client request, the DHCPv6 server returns a list of addresses for the client to configure.

Option Negotiation

In DHCPv6 there is an Option Request Option, which provides a hint to the server of what the client prefers to see. If all possible options were sent from the server to the client, so much information could be sent that some of it would have to be dropped on the way to the client. The

server might use the hint to choose among the options to include in the reply. Alternatively, the server could ignore the hint and choose other items to include. On Oracle Solaris, for example, the preferred options might include the Oracle Solaris DNS address domain or the NIS address domain, but would probably not include the net bios server.

The same type of hint is also provided for DHCPv4, but without the special Option Request Option. Instead DHCPv4 uses the `PARAM_REQUEST_LIST` in `/etc/default/dhcpagent`.

Configuration Syntax

Configure the DHCPv6 client in much the same way as the existing DHCPv4 client, using `/etc/default/dhcpagent`.

The syntax is augmented with a ".v6" marker between the interface name (if any) and the parameter to be configured. For example, the global IPv4 option request list is set like this:

```
PARAM_REQUEST_LIST=1,3,6,12,15,28,43
```

An individual interface can be configured to omit the hostname option like this:

```
bge0.PARAM_REQUEST_LIST=1,3,6,15,28,43
```

To set a global request list for DHCPv6, note the leading dot:

```
.v6.PARAM_REQUEST_LIST=23,24
```

Or, to set an individual interface, follow this example:

```
bge0.v6.PARAM_REQUEST_LIST=21,22,23,24
```

For reference, here is an actual `/etc/default/dhcpagent` file for DHCPv6 configuration:

```
# The default DHCPv6 parameter request list has preference (7), unicast (12),
# DNS addresses (23), DNS search list (24), NIS addresses (27), and
# NIS domain (29).  This may be changed by altering the following parameter-
# value pair.  The numbers correspond to the values defined in RFC 3315 and
# the IANA dhcpv6-parameters registry.
.v6.PARAM_REQUEST_LIST=7,12,23,24,27,29
```

DHCP Client Startup

In most cases, there is nothing you need to do for DHCPv6 client startup. The `in.ndpd` daemon starts up DHCPv6 automatically when it is needed.

For DHCPv4, however, you must request the client startup, if that was not done during Oracle Solaris installation. See "How to Enable the DHCP Client" on page 349.

The dhcpagent daemon obtains configuration information that is needed by other processes involved in booting the system. For this reason, the system startup scripts start dhcpagent early in the boot process and wait until the network configuration information from the DHCP server arrives.

Although the default is to run DHCPv6, you can choose to not have DHCPv6 run. After DHCPv6 starts running, you can stop it with the ipadm delete-addr command. You can also disable DHCPv6 so that it does not start on reboot, by modifying the /etc/inet/ndpd.conf file.

The following example show how to immediately shut down DHCPv6 on the interface named bge0.

```
ex# echo ifdefault StatefulAddrConf false >> /etc/inet/ndpd.conf
ex# pkill -HUP -x in.ndpd
ex# ipadm delete-addr -r dhcp-addrobj
```

At startup, if persistent DHCP configurations exist in the system, then the dhcpagent is started as part of the startup script processes. The dhcpagent then configures the network interfaces as described in "How DHCP Works" on page 229.

DHCPv6 Communication

Unlike DHCPv4, which is invoked by manual configuration, DHCPv6 is invoked by Router Advertisements (RAs). Depending on how the router is configured, the system automatically invokes DHCPv6 on the interface on which the Router Advertisement message was received and uses DHCP to get an address and other parameters, or the system requests only data other than an address (for example, DNS servers) with DHCPv6.

The in.ndpd daemon receives the Router Advertisement message. It does this automatically on all interfaces plumbed for IPv6 on the system. When in.ndpd sees an RA that specifies that DHCPv6 should run, it invokes it.

To prevent in.ndpd from starting up DHCPv6, you can change the /etc/inet/ndpd.conf file.

You can also stop DHCPv6 after it starts by using one of the following versions of ipadm:

ipadm delete-addr *dhcp-addrobj*

or

ipadm delete-addr -r *dhcp-addrobj*

How DHCP Client Protocols Manage Network Configuration Information

DHCPv4 and DHCPv6 client protocols manage network configuration information in different ways. The key difference is that with DHCPv4 the negotiation is for the lease of a single address and some options to go with it. With DHCPv6, the negotiation is over a batch of addresses and a batch of options.

For background information on the interaction between DHCPv4 client and server, see Chapter 11, "About DHCP (Overview)."

How the DHCPv4 Client Manages Network Configuration Information

After the information packet is obtained from a DHCP server, `dhcpagent` configures the network interface and brings up the interface. The daemon controls the interface for the duration of the lease time for the IP address, and maintains the configuration data in an internal table. The system startup scripts use the `dhcpinfo` command to extract configuration option values from the internal table. The values are used to configure the system and enable it to communicate on the network.

The `dhcpagent` daemon waits passively until a period of time elapses, usually half the lease time. The daemon then requests an extension of the lease from a DHCP server. If the system notifies `dhcpagent` that the interface is down or that the IP address has changed, the daemon does not control the interface until instructed by the `ipadm` command to do so. If `dhcpagent` finds that the interface is up and the IP address has not changed, the daemon sends a request to the server for a lease renewal. If the lease cannot be renewed, `dhcpagent` takes down the interface at the end of the lease time.

Each time `dhcpagent` performs an action related to the lease, the daemon looks for an executable file called `/etc/dhcp/eventhook`. If an executable file with this name is found, `dhcpagent` invokes the executable. See "DHCP Client Event Scripts" on page 356 for more information about using the event executable.

How the DHCPv6 Client Manages Network Configuration Information

DHCPv6 communication between client and server begins with the client sending out a Solicit message, to locate servers. In response, all servers available for DHCP service send an Advertise message. The server message contains multiple IA_NA (Identity Association Non-Temporary Address) records plus other options (such as DNS server addresses) that the server can supply.

A client can request particular addresses (and multiples of them) by setting up its own IA_NA/IAADDR records in its Request message. A client typically requests specific addresses if it has old addresses recorded and it would like the server to provide the same ones, if possible. Regardless of what the client does (even if it requests no addresses at all), the server can supply any number of addresses to the client for a single DHCPv6 transaction.

This is a the message dialog that takes place between the clients and servers.

- A client sends a Solicit message to locate servers.
- Servers send an Advertise message to indicate they are available for DHCP service.
- A client sends a Request message to request configuration parameters, including IP addresses, from servers with the greatest preference values. Server preference values are set by the administrator and extend from 0, at the lowest end, to 255 at the highest.
- The server sends a Reply message that contains the address leases and configuration data.

If the preference value in the Advertise message is 255, the DHCPv6 client immediately selects that server. If the most preferred server does not respond, or fails to give a successful Reply to the Request message, then the client continues looking for less-preferred servers (in order) until there are no more Advertise messages on hand. At that point, the client starts over by again sending Solicit messages.

The chosen server sends a Reply message containing assigned addresses and configuration parameters in response to a Solicit or Request message.

DHCP Client Shutdown

At shutdown, the client sends a Release message to the server that assigned addresses to the client, to indicate that the client will no longer use one or more of the assigned addresses. When the DHCPv4 client system shuts down normally, `dhcpagent` writes the current configuration information to the file `/etc/dhcp/`*interface*`.dhc`, or for DHCPv6, to `/etc/dhcp/`*interface*`.dh6`. By default, the lease is saved rather than released, so the DHCP server does not know that the IP address is not in active use, which enables the client to easily regain the address on next boot. This default action is the same as the `ipadm delete-addr` *dhcp-addrobj* command.

If the lease in that file is still valid when the system reboots, `dhcpagent` sends an abbreviated request to use the same IP address and network configuration information. For DHCPv4, this is the Request message. For DHCPv6, the message is Confirm.

If the DHCP server permits this request, `dhcpagent` can use the information that it wrote to disk when the system shut down. If the server does not permit the client to use the information, `dhcpagent` initiates the DHCP protocol sequence described in "How DHCP Works" on page 229. As a result, the client obtains new network configuration information.

Enabling and Disabling a DHCP Client

To enable the DHCP client on a system that is already running Oracle Solaris and is not using DHCP, you must first unconfigure the system. When the system boots, you must issue some commands to set up the system and enable the DHCP client.

Note – In many deployments it is common practice to have crucial parts of the infrastructure set up with static IP addresses, rather than using DHCP. Determining which devices on your network, for example routers and certain servers, should be client and which should not, is beyond the scope of this guide.

▼ How to Enable the DHCP Client

This procedure is necessary only if DHCPv4 was not enabled during Oracle Solaris installation. It is never necessary for DHCPv6.

1 **If this system uses preconfiguration instead of interactive configuration, edit the `sysidcfg` file. Add the `dhcp` subkey to the `network_interface` keyword in the `sysidcfg` file.**

For example, `network_interface=bge0 {dhcp}`. See the `sysidcfg(4)` man page for more information.

2 **Unconfigure and shut down the system.**

```
# sys-unconfig
```

See the `sys-unconfig(1M)` man page for more information about the configuration information that is removed by this command.

3 **Reboot the system after shutdown is complete.**

If the system uses preconfiguration, the `dhcp` subkey in the `sysidcfg` file configures the system to use the DHCP client as the system boots.

If the system does not use preconfiguration, you are prompted for system configuration information by `sysidtool` programs when the system reboots. See the `sysidtool(1M)` man page for more information.

4 **When prompted to use DHCP to configure network interfaces, specify Yes.**

▼ How to Disable an DHCP Client

1 **If you used a `sysidcfg` file to preconfigure the system, remove the `dhcp` subkey from the `network_interface` keyword.**

2 **Unconfigure and shut down the system.**

```
# sys-unconfig
```

See the sys-unconfig(1M) man page for more information about the configuration information that is removed by this command.

3 **Reboot the system after shutdown is complete.**

If the system uses preconfiguration, you are not prompted for configuration information, and the DHCP client is not configured.

If the system does not use preconfiguration, you are prompted for system configuration information by `sysidtool` programs when the system reboots. See the sysidtool(1M) man page for more information.

4 **When prompted to use DHCP to configure network interfaces, specify No.**

DHCP Client Administration

The DHCP client software does not require administration under normal system operation. The `dhcpagent` daemon automatically starts when the system boots, renegotiates leases, and stops when the system shuts down. You should not manually start and stop the `dhcpagent` daemon directly. Instead, as superuser on the client system, you can use the `ipadm` command to affect `dhcpagent`'s management of the network interface, if necessary.

`ipadm` Command Options Used With the DHCP Client

This section summarizes the command options, which are documented in the ipadm(1M) man page.

The `ipadm` command enables you to do the following:

- **Start the DHCP client** – The command `ipadm create-addr -T dhcp` *dhcp-addrobj* initiates the interaction between `dhcpagent` and the DHCP server to obtain an IP address and a new set of configuration options. This command is useful when you change information that you want a client to use immediately, such as when you add IP addresses or change the subnet mask.
- **Request network configuration information only** – The command `ipadm refresh-addr -i` *dhcp-addrobj* causes `dhcpagent` to issue a request for network configuration parameters, with the exception of the IP address. This command is useful when the network interface has a static IP address, but the client system needs updated network options. For example, this command is useful if you do not use DHCP to manage IP addresses, but you do use it to configure hosts on the network.

- **Request a lease extension** – The command `ipadm refresh-addr` *dhcp-addrobj* causes `dhcpagent` to issue a request to renew the lease. The client does automatically request to renew leases. However, you might want to use this command if you change the lease time and want clients to use the new lease time immediately, rather than waiting for the next attempt at lease renewal.
- **Release the IP address** – The command `ipadm delete-addr -r` *dhcp-addrobj* causes `dhcpagent` to relinquish the IP address used by the network interface. Release of the IP address happens automatically when the lease expires. You might want to issue this command with a laptop, for example, when leaving a network and planning to start the system on a new network. See also the `/etc/default/dhcpagent` configuration file `RELEASE_ON_SIGTERM` property.
- **Drop the IP address** – The command `ipadm delete-addr` *dhcp-addrobj* causes `dhcpagent` to take down the network interface without informing the DHCP server and cache the lease in the file system. This command enables the client to use the same IP address when it reboots.

Note – Currently, the `ipadm` command has no equivalent functionality for the `ifocnfig [inet6] interface status` command.

Setting DHCP Client Configuration Parameters

The `/etc/default/dhcpagent` file on the client system contains tunable parameters for the `dhcpagent`. You can use a text editor to change several parameters that affect client operation. The `/etc/default/dhcpagent` file is well documented, so for more information, you should refer to the file as well as to the `dhcpagent(1M)` man page.

By default, the DHCP client is configured as follows:

For DHCPv4

- The client system does not require a particular host name.

  If you want a client to request a specific host name, see "DHCPv4 Client Host Names" on page 353.
- Default requests for the client are given in `/etc/default/dhcpagent`, and includes DNS Server, DNS domain, and broadcast address.

  The DHCP client's parameter file can be set up to request more options in the `PARAM_REQUEST_LIST` keyword in the `/etc/default/dhcpagent` file. The DHCP server can be configured to provide options that were not specifically requested. See "About DHCP Macros" on page 239 and "Working With DHCP Macros (Task Map)" on page 312 for information about using DHCP server macros to send information to clients.

For DHCPv4 and DHCPv6

- The client system uses DHCP on one physical network interface.

  If you want to use DHCP on more than one physical network interface, see "DHCP Client Systems With Multiple Network Interfaces" on page 352.
- The client is not automatically configured as a name service client if the DHCP client was configured after the Oracle Solaris installation.

  See "DHCP Client Systems and Name Services" on page 354 for information about using name services with DHCP clients.

DHCP Client Systems With Multiple Network Interfaces

The DHCP client can simultaneously manage several different interfaces on one system. The interfaces can be physical interfaces or logical interfaces. Each interface has its own IP address and lease time. If more than one network interface is configured for DHCP, the client issues separate requests to configure them. The client maintains a separate set of network configuration parameters for each interface. Although the parameters are stored separately, some of the parameters are global in nature. The global parameters apply to the system as a whole, rather than to a particular network interface.

The host name, NIS domain name, and time zone are examples of global parameters. Global parameters usually have different values for each interface. However, only one value can be used for each global parameter associated with each system. To be sure that there is only one answer to a query for a global parameter, only the parameters for the primary network interface are used. You can insert the word `primary` in the `/etc/dhcp.`*interface* file for the interface that you want to be treated as the primary interface. If the `primary` keyword is not used, the first interface in alphabetical order is considered to be the primary interface.

The DHCP client manages leases for logical interfaces and physical interfaces identically, except for the following limitation on logical interfaces:

- The DHCP client does not manage the default routes that are associated with logical interfaces.

  The Oracle Solaris kernel associates routes with physical interfaces, not logical interfaces. When a physical interface's IP address is established, the necessary default routes should be placed in the routing table. If DHCP is used subsequently to configure a logical interface associated with that physical interface, the necessary routes should already be in place. The logical interface uses the same routes.

  When a lease expires on a physical interface, the DHCP client removes the default routes that are associated with the interface. When a lease expires on a logical interface, the DHCP client does not remove the default routes associated with the logical interface. The associated physical interface and possibly other logical interfaces might need to use the same routes.

If you need to add or remove default routes that are associated with a DHCP-controlled interface, you can use the DHCP client event script mechanism. See "DHCP Client Event Scripts" on page 356.

DHCPv4 Client Host Names

By default, the DHCPv4 client does not supply its own host name, because the client expects the DHCP server to supply the host name. The DHCPv4 server is configured to supply host names to DHCPv4 clients by default. When you use the DHCPv4 client and server together, these defaults work well. However, when you use the DHCPv4 client with some third-party DHCP servers, the client might not receive a host name from the server. If the DHCP client does not receive a host name through DHCP, the client system looks at the `/etc/nodename` file for a name to use as the host name. If the file is empty, the host name is set to `unknown`.

If the DHCP server supplies a name in the DHCP `Hostname` option, the client uses that host name, even if a different value is placed in the `/etc/nodename` file. If you want the client to use a specific host name, you can enable the client to request that name. See the following procedure.

Note – The following procedure does not work with all DHCP servers. Through this procedure you are requiring the client to send a specific host name to the DHCP server, and to expect the same name in return.

However, the DHCP server does not have to respect this request and many do not. They simply return a different name.

▼ How to Enable a DHCPv4 Client to Request a Specific Host Name

1. **On the client system, edit the `/etc/default/dhcpagent` file.**

2. **Find the `REQUEST_HOSTNAME` keyword in the `/etc/default/dhcpagent` file and modify the keyword as follows:**

   ```
   REQUEST_HOSTNAME=yes
   ```

   If a comment sign (#) is in front of `REQUEST_HOSTNAME`, remove the #. If the `REQUEST_HOSTNAME` keyword is not present, insert the keyword.

3. **Edit the `/etc/hostname.`*interface* file on the client system to add the following line:**

   `inet` *hostname*

   hostname is the name that you want the client to use.

4 **Type the following commands to have the client perform a full DHCP negotiation upon rebooting:**

```
# ipadm delete-addr -r dhcp-addrobj
# reboot
```

The DHCP data that is cached on the client is removed. The client restarts the protocol to request new configuration information, including a new host name. The DHCP server first makes sure that the host name is not in use by another system on the network. The server then assigns the host name to the client. If configured to do so, the DHCP server can update name services with the client's host name.

If you want to change the host name later, repeat Step 3 and Step 4.

DHCP Client Systems and Name Services

Oracle Solaris systems support the following name services: DNS, NIS, and a local file store (`/etc/inet/hosts`). Each name service requires some configuration before it is usable. The name service switch configuration file (see `nsswitch.conf(4)`) must also be set up appropriately to indicate the name services to be used.

Before a DHCP client system can use a name service, you must configure the system as a client of the name service. By default, and unless configured otherwise during system installation, only local files are used.

The following table summarizes issues that are related to each name service and DHCP. The table includes links to documentation that can help you set up clients for each name service.

TABLE 15–1 Name Service Client Setup Information for DHCP Client Systems

Name Service	Client Setup Information

NIS	If you are using DHCP to send Oracle Solaris network install information to a client system, you can use a configuration macro that contains the `NISservs` and `NISdmain` options. These options pass the IP addresses of NIS servers and the NIS domain name to the client. The client then automatically becomes an NIS client.
	
	If a DHCP client system is already running Oracle Solaris, the NIS client is not automatically configured on that system when the DHCP server sends NIS information to the client.
	
	If the DHCP server is configured to send NIS information to the DHCP client system, you can see the values given to the client if you use the `dhcpinfo` command on the client as follows:
	
	`# /sbin/dhcpinfo NISdmain`
	
	`# /sbin/dhcpinfo NISservs`
	
	Note – For DHCPv6, include `-v6`, and different protocol keywords in the command.
	
	`# /sbin/dhcpinfo -v6 NISDomain`
	
	`# /sbin/dhcpinfo -v6 NISServers`
	
	Use the values returned for the NIS domain name and NIS servers when you set up the system as an NIS client.
	
	You set up an NIS client for an DHCP client system in the standard way, as documented in Chapter 5, "Setting Up and Configuring NIS Service," in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.
	
	Tip – You can write a script that uses `dhcpinfo` and `ypinit` to automate NIS client configuration on DHCP client systems.

`/etc/inet/hosts`	You must set up the `/etc/inet/hosts` file for a DHCP client system that is to use `/etc/inet/hosts` for its name service. The DHCP client system's host name is added to its own `/etc/inet/hosts` file by the DHCP tools. However, you must manually add the host name to the `/etc/inet/hosts` files of other systems in the network. If the DHCP server system uses `/etc/inet/hosts` for name resolution, you must also manually add the client's host name on the system.
DNS	If the DHCP client system receives the DNS domain name through DHCP, the client system's `/etc/resolv.conf` file is configured automatically. The `/etc/nsswitch.conf` file is also automatically updated to append `dns` to the `hosts` line after any other name services in the search order. See *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for more information about DNS.

DHCP Client Event Scripts

You can set up the DHCP client to run an executable program or script that can perform any action that is appropriate for the client system. The program or script, which is called an *event script*, is automatically executed after certain DHCP lease events occur. The event script can be used to run other commands, programs, or scripts in response to specific lease events. You must provide your own event script to use this feature.

The following event keywords are used by `dhcpagent` to signify DHCP lease events:

Event Keyword	Description
`BOUND` and `BOUND6`	The interface is configured for DHCP. The client receives the acknowledgement message (DHCPv4 `ACK`) or (DHCPv6 `Reply`) from the DHCP server, which grants the lease request for an IP address. The event script is invoked immediately after the interface is configured successfully.
`EXTEND` and `EXTEND6`	The client successfully extends a lease. The event script is invoked immediately after the client receives the acknowledgement message from the DHCP server for the renew request.
`EXPIRE` and `EXPIRE6`	The lease expires when the lease time is up. For DHCPv4, the event script is invoked immediately before the leased address is removed from the interface and the interface is marked as down. For DHCPv6, the event script is invoked just before the last remaining leased addresses are removed from the interface.
`DROP` and `DROP6`	The client drops the lease to remove the interface from DHCP control. The event script is invoked immediately before the interface is removed from DHCP control.
`RELEASE` and `RELEASE6`	The client relinquishes the IP address. The event script is invoked immediately before the client releases the address on the interface and sends the DHCPv4 `RELEASE` or DHCPv6 `Release` packet to the DHCP server.
`INFORM` and `INFORM6`	An interface acquires new or updated configuration information from a DHCP server through the DHCPv4 `INFORM` or the DHCPv6 `Information-Request` message. These events occur when the DHCP client obtains only configuration parameters from the server and does not obtain an IP address lease.
`LOSS6`	During lease expiration, when one or more valid leases still remain, the event script is invoked just before expired addresses are removed. Those being removed are marked with the `IFF_DEPRECATED` flag.

With each of these events, `dhcpagent` invokes the following command:

```
/etc/dhcp/eventhook interface event
```

where *interface* is the interface that is using DHCP and *event* is one of the event keywords described previously. For example, when the `ce0` interface is first configured for DHCP, the `dhcpagent` invokes the event script as follows:

```
/etc/dhcp/eventhook ce0 BOUND
```

To use the event script feature, you must do the following:

- Name the executable file `/etc/dhcp/eventhook`.
- Set the owner of the file to be `root`.
- Set permissions to `755` (`rwxr-xr-x`).
- Write the script or program to perform a sequence of actions in response to any of the documented events. Because Sun might add new events, the program must silently ignore any events that are not recognized or do not require action. For example, the program or script might write to a log file when the event is `RELEASE`, and ignore all other events.
- Make the script or program noninteractive. Before the event script is invoked, `stdin`, `stdout`, and `stderr` are connected to `/dev/null`. To see the output or errors, you must redirect to a file.

The event script inherits its program environment from `dhcpagent`, and runs with `root` privileges. The script can use the `dhcpinfo` utility to obtain more information about the interface, if necessary. See the `dhcpinfo(1)` man page for more information.

The `dhcpagent` daemon waits for the event script to exit on all events. If the event script does not exit after 55 seconds, `dhcpagent` sends a `SIGTERM` signal to the script process. If the process still does not exit after three additional seconds, the daemon sends a `SIGKILL` signal to kill the process.

The `dhcpagent(1M)` man page includes one example of an event script.

Example 15–1 shows how to use a DHCP event script to keep the content of the `/etc/resolv.conf` file up to date. When the `BOUND` and `EXTEND` events occur, the script replaces the names of the domain server and name server. When the `EXPIRE`, `DROP` and `RELEASE` events occur, the script removes the names of the domain server and name server from the file.

Note – The example script assumes that DHCP is the authoritative source for the names of the domain server and the name server. The script also assumes that all interfaces under DHCP control return consistent and current information. These assumptions might not reflect conditions on your system.

EXAMPLE 15–1 Event Script for Updating the `/etc/resolv.conf` File

```
#!/bin/ksh -p

PATH=/bin:/sbin export PATH
umask 0222

# Refresh the domain and name servers on /etc/resolv.conf

insert ()
{
    dnsservers=‘dhcpinfo -i $1 DNSserv‘
    if [ -n "$dnsservers" ]; then
        # remove the old domain and name servers
        if [ -f /etc/resolv.conf ]; then
            rm -f /tmp/resolv.conf.$$
            sed -e ’/^domain/d’ -e ’/^nameserver/d’ \
                /etc/resolv.conf > /tmp/resolv.conf.$$
        fi

        # add the new domain
        dnsdomain=‘dhcpinfo -i $1 DNSdmain‘
        if [ -n "$dnsdomain" ]; then
            echo "domain $dnsdomain" >> /tmp/resolv.conf.$$
        fi

        # add new name servers
        for name in $dnsservers; do
            echo nameserver $name >> /tmp/resolv.conf.$$
        done
        mv -f /tmp/resolv.conf.$$ /etc/resolv.conf
    fi
}

# Remove the domain and name servers from /etc/resolv.conf

remove ()
{
    if [ -f /etc/resolv.conf ]; then
        rm -f /tmp/resolv.conf.$$
        sed -e ’/^domain/d’ -e ’/^nameserver/d’ \
            /etc/resolv.conf > /tmp/resolv.conf.$$
        mv -f /tmp/resolv.conf.$$ /etc/resolv.conf
    fi
}

case $2 in
BOUND | EXTEND)
    insert $1
    exit 0
    ;;
EXPIRE | DROP | RELEASE)
    remove
    exit 0
    ;;
*)
    exit 0
    ;;
```

EXAMPLE 15–1 Event Script for Updating the `/etc/resolv.conf` File *(Continued)*

```
esac
```

◆ ◆ ◆ CHAPTER 16

Troubleshooting DHCP (Reference)

This chapter provides information to help you solve problems that you might encounter when you configure a DHCP server or client. The chapter also helps you with problems you might have in using DHCP after configuration is complete.

The chapter includes the following information:

See Chapter 13, "Configuring the DHCP Service (Tasks)," for information about configuring your DHCP server. See"Enabling and Disabling a DHCP Client" on page 349 for information about configuring your DHCP client.

Troubleshooting DHCP Server Problems

The problems that you might encounter when you configure the server fall into the following categories:

IP Address Allocation Errors in DHCP

When a client attempts to obtain or verify an IP address, you might see problems logged to `syslog` or in server debugging mode output. The following list of common error messages indicates the possible causes and solutions.

`There is no` *n.n.n.n* `dhcp-network table for DHCP client's network`

Cause: A client is requesting a specific IP address or seeking to extend a lease on its current IP address. The DHCP server cannot find the DHCP network table for that address.

Solution: The DHCP network table might have been deleted mistakenly. You can recreate the network table by adding the network again using DHCP Manager or the `dhcpconfig` command.

`ICMP ECHO reply to OFFER candidate:` *n.n.n.n*`, disabling`

Cause: The IP address considered for offering to a DHCP client is already in use. This problem might occur if more than one DHCP server owns the address. The problem might also occur if an address was manually configured for a non-DHCP network client.

Solution: Determine the proper ownership of the address. Correct either the DHCP server database or the host's network configuration.

`ICMP ECHO reply to OFFER candidate:` *n.n.n.n*`. No corresponding dhcp network record.`

Cause: The IP address considered for offering to a DHCP client does not have a record in a network table. This error indicates that the IP address record was deleted from the DHCP network table after the address was selected. This error can only happen in the brief period before the duplicate address check is completed.

Solution: Use DHCP Manager or the `pntadm` command to view the DHCP network table. If the IP address is missing, create the address with DHCP Manager by choosing Create from the Edit menu on the Address tab. You can also use `pntadm` to create the IP address.

`DHCP network record for` *n.n.n.n*`is unavailable, ignoring request.`

Cause: The record for the requested IP address is not in the DHCP network table, so the server is dropping the request.

Solution: Use DHCP Manager or the `pntadm` command to view the DHCP network table. If the IP address is missing, create the address with DHCP Manager by choosing Create from the Edit menu on the Address tab. You can also use `pntadm` to create the address.

n.n.n.n `currently marked as unusable.`
Cause: The requested IP address cannot be offered because the address has been marked in the network table as unusable.

Solution: You can use DHCP Manager or the `pntadm` command to make the address usable.

n.n.n.n `was manually allocated. No dynamic address will be allocated.`
Cause: The client ID has been assigned a manually allocated address, and that address is marked as unusable. The server cannot allocate a different address to this client.

Solution: You can use DHCP Manager or the `pntadm` command to make the address usable, or manually allocate a different address to the client.

`Manual allocation (`*n.n.n.n, client ID*`) has` *n* `other records. Should have 0.`
Cause: The client that has the specified client ID has been manually assigned more than one IP address. A client should be assigned only one address. The server selects the last manually assigned address that is found in the network table.

Solution: Use DHCP Manager or the `pntadm` command to modify IP addresses to remove the additional manual allocations.

`No more IP addresses on` *n.n.n.n*`network.`
Cause: All IP addresses currently managed by DHCP on the specified network have been allocated.

Solution: Use DHCP Manager or the `pntadm` command to create new IP addresses for this network.

`Client:` *clientid* `lease on` *n.n.n.n* `expired.`
Cause: The lease was not negotiable and timed out.

Solution: The client should automatically restart the protocol to obtain a new lease.

`Offer expired for client:` *n.n.n.n*
Cause: The server made an IP address offer to the client, but the client took too long to respond and the offer expired.

Solution: The client should automatically issue another discover message. If this message also times out, increase the cache offer time out for the DHCP server. In DHCP Manager, choose Modify from the Service menu.

`Client:` *clientid* `REQUEST is missing requested IP option.`

Cause: The client's request did not specify the offered IP address, so the DHCP server ignored the request. This problem might occur if you use a third-party DHCP client that is not compliant with the updated DHCP protocol, RFC 2131.

Solution: Update the client software.

`Client:` *clientid* `is trying to renew` *n.n.n.n*`, an IP address it has not leased.`

Cause: The IP address for this client in the DHCP network table does not match the IP address that the client specified in its renewal request. The DHCP server does not renew the lease. This problem might occur if you delete a client's record while the client is still using the IP address.

Solution: Use DHCP Manager or the `pntadm` command to examine the network table, and correct the client's record, if necessary. The client ID should be bound to the specified IP address. If the client ID is not bound, edit the address properties to add the client ID.

`Client:` *clientid* `is trying to verify unrecorded address:` *n.n.n.n*`, ignored.`

Cause: The specified client has not been registered in the DHCP network table with this address, so the request is ignored by this DHCP server.

Another DHCP server on the network might have assigned this client the address. However, you might also have deleted the client's record while the client was still using the IP address.

Solution: Use DHCP Manager or the `pntadm` command to examine the network table on this server and any other DHCP servers on the network. Make corrections, if necessary.

You can also do nothing and allow the lease to expire. The client automatically requests a new address lease.

If you want the client to get a new lease immediately, restart the DHCP protocol on the client by typing the following commands:

```
ipadm delete-addr -r dhcp-addrobj
ipadm create-addr -T dhcp dhcp-addrobj
```

Troubleshooting DHCP Client Configuration Problems

The problems that you might encounter with a DHCP client fall into the following categories:

- "Problems Communicating With the DHCP Server" on page 365
- "Problems With Inaccurate DHCP Configuration Information" on page 373

Problems Communicating With the DHCP Server

This section describes problems that you might encounter as you add DHCP clients to the network.

After you enable the client software and reboot the system, the client tries to reach the DHCP server to obtain its network configuration. If the client fails to reach the server, you might see error messages such as the following:

```
DHCP or BOOTP server not responding
```

Before you can determine the problem, you must gather diagnostic information from both the client and the server. To gather information, you can perform the following tasks:

1. "How to Run the DHCP Client in Debugging Mode" on page 365
2. "How to Run the DHCP Server in Debugging Mode" on page 366
3. "How to Use `snoop` to Monitor DHCP Network Traffic" on page 366

You can do these things separately or concurrently.

The information that you gather can help you determine if the problem is with the client, server, or a relay agent. Then, you can find a solution.

How to Run the DHCP Client in Debugging Mode

If the client is not a DHCP client, refer to the client's documentation for information about how to run the client in debugging mode.

If you have a DHCP client, use the following steps.

1 Kill the DHCP client daemon.

```
# pkill -x dhcpagent
```

2 Restart the daemon in debugging mode.

```
# /sbin/dhcpagent -d1 -f &
```

The `-d` switch puts the DHCP client in debugging mode with level 1 verbosity. The `-f` switch causes output to be sent to the console instead of to `syslog`.

3 Configure the interface to start DHCP negotiation.

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

When run in debugging mode, the client daemon displays messages to your screen while performing DHCP requests. See "Output from DHCP Client in Debugging Mode" on page 366 for information about client debugging mode output.

How to Run the DHCP Server in Debugging Mode

1. **Stop the DHCP server temporarily.**

   ```
   # svcadm disable -t svc:/network/dhcp-server
   ```

   You can also use DHCP Manager or `dhcpconfig` to stop the server.

2. **Restart the daemon in debugging mode.**

   ```
   # /usr/lib/inet/in.dhcpd -d -v
   ```

   You should also use any `in.dhcpd` command-line options that you normally use when you run the daemon. For example, if you run the daemon as a BOOTP relay agent, include the `-r` option with the `in.dhcpd -d -v` command.

   When run in debugging mode, the daemon displays messages to your screen while processing DHCP or BOOTP requests. See "Output from the DHCP Server in Debugging Mode" on page 367 for information about server debugging mode output.

How to Use `snoop` to Monitor DHCP Network Traffic

1. **Start `snoop` to begin tracing network traffic across the server's network interface.**

   ```
   # /usr/sbin/snoop -d interface -o snoop-output-filename udp port 67 or udp port 68
   ```

   For example, you might type the following command:

   ```
   #  /usr/sbin/snoop -d bge0 -o /tmp/snoop.output udp port 67 or udp port 68
   ```

   `snoop` continues to monitor the interface until you stop `snoop` by pressing Control-C after you have the information that you need.

2. **Boot the client system, or restart the `dhcpagent` on the client system.**

   "How to Run the DHCP Client in Debugging Mode" on page 365 describes how to restart `dhcpagent`.

3. **On the server system, use `snoop` to display the output file with the contents of network packets:**

   ```
   # /usr/sbin/snoop -i snoop-output-filename -x0 -v
   ```

   For example, you might type the following command:

   ```
   # /usr/sbin/snoop -i /tmp/snoop.output -x0 -v
   ```

See Also See "DHCP `snoop` Output" on page 370 for information about interpreting the output.

Output from DHCP Client in Debugging Mode

The following example shows normal output when a DHCP client in debugging mode sends its DHCP request and receives its configuration information from a DHCP server.

EXAMPLE 16–1 Normal Output from the DHCP Client in Debugging Mode

```
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x27fc8 (DHCP filter)
/sbin/dhcpagent: debug: init_ifs: initted interface bge0
/sbin/dhcpagent: debug: insert_ifs: bge0: sdumax 1500, optmax 1260, hwtype 1, hwlen 6
/sbin/dhcpagent: debug: insert_ifs: inserted interface bge0
/sbin/dhcpagent: debug: register_acknak: registered acknak id 5
/sbin/dhcpagent: debug: unregister_acknak: unregistered acknak id 5
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x26018 (ARP reply filter)
/sbin/dhcpagent: info: setting IP netmask on bge0 to 255.255.192.0
/sbin/dhcpagent: info: setting IP address on bge0 to 10.23.3.233
/sbin/dhcpagent: info: setting broadcast address on bge0 to 10.23.63.255
/sbin/dhcpagent: info: added default router 10.23.0.1 on bge0
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x28054 (blackhole filter)
/sbin/dhcpagent: debug: configure_if: bound ifsp->if_sock_ip_fd
/sbin/dhcpagent: info: bge0 acquired lease, expires Tue Aug 10 16:18:33 2006
/sbin/dhcpagent: info: bge0 begins renewal at Tue Aug 10 15:49:44 2006
/sbin/dhcpagent: info: bge0 begins rebinding at Tue Aug 10 16:11:03 2006
```

If the client cannot reach the DHCP server, you might see debugging mode output that is similar to the output shown in the following example.

EXAMPLE 16–2 Output Indicating a Problem from the DHCP Client in Debugging Mode

```
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x27fc8 (DHCP filter)
/sbin/dhcpagent: debug: init_ifs: initted interface bge0
/sbin/dhcpagent: debug: select_best: no valid OFFER/BOOTP reply
/sbin/dhcpagent: debug: select_best: no valid OFFER/BOOTP reply
/sbin/dhcpagent: debug: select_best: no valid OFFER/BOOTP reply
```

If you see this message, the client request never reached the server, or the server cannot send a response to the client. Run `snoop` on the server as described in "How to Use `snoop` to Monitor DHCP Network Traffic" on page 366 to determine if packets from the client have reached the server.

Output from the DHCP Server in Debugging Mode

Normal server debugging mode output shows server configuration information followed by information about each network interface as the daemon starts. After daemon startup, the debugging mode output shows information about requests the daemon processes. Example 16–3 shows debugging mode output for a DHCP server that has just started. The server extends the lease for a client that is using an address owned by another DHCP server that is not responding.

EXAMPLE 16–3 Normal Output for DHCP Server in Debugging Mode

```
Daemon Version: 3.1
Maximum relay hops: 4
Transaction logging to console enabled.
Run mode is: DHCP Server Mode.
Datastore: nisplus
```

EXAMPLE 16–3 Normal Output for DHCP Server in Debugging Mode *(Continued)*

```
Path: org_dir.dhcp.test..:dhcp.test..:$
DHCP offer TTL: 10
Ethers compatibility enabled.
BOOTP compatibility enabled.
ICMP validation timeout: 1000 milliseconds, Attempts: 2.
Monitor (0005/bge0) started...
Thread Id: 0005 - Monitoring Interface: bge0 *****
MTU: 1500      Type: DLPI
Broadcast: 10.21.255.255
Netmask: 255.255.0.0
Address: 10.21.0.2
Monitor (0006/nf0) started...
Thread Id: 0006 - Monitoring Interface: nf0 *****
MTU: 4352      Type: DLPI
Broadcast: 10.22.255.255
Netmask: 255.255.0.0
Address: 10.22.0.1
Monitor (0007/qfe0) started...
Thread Id: 0007 - Monitoring Interface: qfe0 *****
MTU: 1500      Type: DLPI
Broadcast: 10.23.63.255
Netmask: 255.255.192.0
Address: 10.23.0.1
Read 33 entries from DHCP macro database on Tue Aug 10 15:10:27 2006
Datagram received on network device: qfe0
Client: 0800201DBA3A is requesting verification of address owned by 10.21.0.4
Datagram received on network device: qfe0
Client: 0800201DBA3A is requesting verification of address owned by 10.21.0.4
Datagram received on network device: qfe0
Client: 0800201DBA3A is requesting verification of address owned by 10.21.0.4
Datagram received on network device: qfe0
Client: 0800201DBA3A maps to IP: 10.23.3.233
Unicasting datagram to 10.23.3.233 address.
Adding ARP entry: 10.23.3.233 == 0800201DBA3A
DHCP EXTEND 0934312543 0934316143 10.23.3.233 10.21.0.2
          0800201DBA3A SUNW.Ultra-5_10 0800201DBA3A
```

Example 16–4 shows debugging mode output from a DHCP daemon that starts as a BOOTP relay agent. The agent relays requests from a client to a DHCP server, and relays the server's responses to the client.

EXAMPLE 16–4 Normal Output from BOOTP Relay in Debugging Mode

```
Relay destination: 10.21.0.4 (blue-servr2)     network: 10.21.0.0
Daemon Version: 3.1
Maximum relay hops: 4
Transaction logging to console enabled.
Run mode is: Relay Agent Mode.
Monitor (0005/bge0) started...
Thread Id: 0005 - Monitoring Interface: bge0 *****
MTU: 1500      Type: DLPI
Broadcast: 10.21.255.255
Netmask: 255.255.0.0
Address: 10.21.0.2
```

EXAMPLE 16–4 Normal Output from BOOTP Relay in Debugging Mode *(Continued)*

```
Monitor (0006/nf0) started...
Thread Id: 0006 - Monitoring Interface: nf0 *****
MTU: 4352      Type: DLPI
Broadcast: 10.22.255.255
Netmask: 255.255.0.0
Address: 10.22.0.1
Monitor (0007/qfe0) started...
Thread Id: 0007 - Monitoring Interface: qfe0 *****
MTU: 1500      Type: DLPI
Broadcast: 10.23.63.255
Netmask: 255.255.192.0
Address: 10.23.0.1
Relaying request 0800201DBA3A to 10.21.0.4, server port.
BOOTP RELAY-SRVR 0934297685 0000000000 0.0.0.0 10.21.0.4 0800201DBA3A
N/A 0800201DBA3A
Packet received from relay agent: 10.23.0.1
Relaying reply to client 0800201DBA3A
Unicasting datagram to 10.23.3.233 address.
Adding ARP entry: 10.23.3.233 == 0800201DBA3A
BOOTP RELAY-CLNT 0934297688 0000000000 10.23.0.1 10.23.3.233 0800201DBA3A
N/A 0800201DBA3A
Relaying request 0800201DBA3A to 10.21.0.4, server port.
BOOTP RELAY-SRVR 0934297689 0000000000 0.0.0.0 10.21.0.4 0800201DBA3A
N/A 0800201DBA3A
Packet received from relay agent: 10.23.0.1
Relaying reply to client 0800201DBA3A
Unicasting datagram to 10.23.3.233 address.
Adding ARP entry: 10.23.3.233 == 0800201DBA3A
```

If there is a problem with DHCP, the debugging mode output might display warnings or error messages. Use the following list of DHCP server error messages to find solutions.

`ICMP ECHO reply to OFFER candidate:` *ip_address* `disabling`

Cause: Before the DHCP server offers an IP address to a client, the server pings the address to verify that the address is not in use. If a client replies, the address is in use.

Solution: Make sure the addresses that you configured are not already in use. You can use the `ping` command. See the `ping(1M)` man page for more information.

`No more IP addresses on` *network-address* `network.`

Cause: No IP addresses are available in the DHCP network table associated with the client's network.

Solution: Create more IP addresses with DHCP Manager or the `pntadm` command. If the DHCP daemon is monitoring multiple subnets, be sure the additional addresses are for the subnet where the client is located. See "Adding IP Addresses to the DHCP Service" on page 301 for more information.

`No more IP addresses for` *network-address* `network when you are running the DHCP daemon in BOOTP compatibility mode.`

Cause: BOOTP does not use a lease time, so the DHCP server looks for free addresses with the BOOTP flag set to allocate to BOOTP clients.

Solution: Use DHCP Manager to allocate BOOTP addresses. See "Supporting BOOTP Clients With the DHCP Service (Task Map)" on page 295.

`Request to access nonexistent per network database:` *database-name* `in datastore:` *datastore*.

Cause: During configuration of the DHCP server, a DHCP network table for a subnet was not created.

Solution: Use DHCP Manager or the `pntadm` command to create the DHCP network table and new IP addresses. See "Adding DHCP Networks" on page 288.

`There is no` *table-name* `dhcp-network table for DHCP client's network.`

Cause: During configuration of the DHCP server, a DHCP network table for a subnet was not created.

Solution: Use DHCP Manager or the `pntadm` command to create the DHCP network table and new IP addresses. See "Adding DHCP Networks" on page 288.

`Client using non_RFC1048 BOOTP cookie.`

Cause: A device on the network is trying to access an unsupported implementation of BOOTP.

Solution: Ignore this message, unless you need to configure this device. If you want to support the device, see "Supporting BOOTP Clients With the DHCP Service (Task Map)" on page 295 for more information.

DHCP `snoop` Output

In the `snoop` output, you should see that packets are exchanged between the DHCP client system and the DHCP server system. The IP address for each system is indicated in each packet. IP addresses for any routers or relay agents in the packet's path are also included. If the systems do not exchange packets, the client system might not be able to contact the server system at all. The problem is then at a lower level.

To evaluate `snoop` output, you must know what the expected behavior is. For example, you must know if the request should be going through a BOOTP relay agent. You must also know the MAC addresses and the IP address of the systems involved so that you can determine if those values are as expected. If there is more than one network interface, you must know the addresses of the network interfaces as well.

The following example shows normal `snoop` output for a DHCP acknowledgement message sent from the DHCP server on `blue-servr2` to a client whose MAC address is `8:0:20:8e:f3:7e`. In the message, the server assigns the client the IP address `192.168.252.6` and the host name `white-6`. The message also includes a number of standard network options and several vendor-specific options for the client.

EXAMPLE 16–5 Sample `snoop` Output for One Packet

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 26 arrived at 14:43:19.14
ETHER:  Packet size = 540 bytes
ETHER:  Destination = 8:0:20:8e:f3:7e, Sun
ETHER:  Source      = 8:0:20:1e:31:c1, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 526 bytes
IP:   Identification = 64667
IP:   Flags = 0x4 IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 254 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 157a
IP:   Source address = 10.21.0.4, blue-servr2
IP:   Destination address = 192.168.252.6, white-6
IP:   No options
IP:   UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 67
UDP:  Destination port = 68 (BOOTPC)
UDP:  Length = 506
UDP:  Checksum = 5D4C
UDP:
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP:
DHCP: Hardware address type (htype) =  1 (Ethernet (10Mb))
DHCP: Hardware address length (hlen) = 6 octets
DHCP: Relay agent hops = 0
DHCP: Transaction ID = 0x2e210f17
DHCP: Time since boot = 0 seconds
DHCP: Flags = 0x0000
DHCP: Client address (ciaddr) = 0.0.0.0
DHCP: Your client address (yiaddr) = 192.168.252.6
DHCP: Next server address (siaddr) = 10.21.0.2
DHCP: Relay agent address (giaddr) = 0.0.0.0
DHCP: Client hardware address (chaddr) = 08:00:20:11:E0:1B
DHCP:
```

EXAMPLE 16–5 Sample snoop Output for One Packet *(Continued)*

```
DHCP: ----- (Options) field options -----
DHCP:
DHCP: Message type = DHCPACK
DHCP: DHCP Server Identifier = 10.21.0.4
DHCP: Subnet Mask = 255.255.255.0
DHCP: Router at = 192.168.252.1
DHCP: Broadcast Address = 192.168.252.255
DHCP: NISPLUS Domainname = dhcp.test
DHCP: IP Address Lease Time = 3600 seconds
DHCP: UTC Time Offset = -14400 seconds
DHCP: RFC868 Time Servers at = 10.21.0.4
DHCP: DNS Domain Name = sem.example.com
DHCP: DNS Servers at = 10.21.0.1
DHCP: Client Hostname = white-6
DHCP: Vendor-specific Options (166 total octets):
DHCP:   (02) 04 octets  0x8194AE1B (unprintable)
DHCP:   (03) 08 octets  "pacific"
DHCP:   (10) 04 octets  0x8194AE1B (unprintable)
DHCP:   (11) 08 octets  "pacific"
DHCP:   (15) 05 octets  "xterm"
DHCP:   (04) 53 octets  "/export/s2/base.s2s/latest/Solaris_8/Tools/Boot"
DHCP:   (12) 32 octets  "/export/s2/base.s2s/latest"
DHCP:   (07) 27 octets  "/platform/sun4u/kernel/unix"
DHCP:   (08) 07 octets  "EST5EDT"
  0: 0800 208e f37e 0800 201e 31c1 0800 4500    .. .ó~.. .1...E.
 16: 020e fc9b 4000 fe11 157a ac15 0004 c0a8    ....@....z......
 32: fc06 0043 0044 01fa 5d4c 0201 0600 2e21    ...C.D..]L.....!
 48: 0f17 0000 0000 0000 0000 c0a8 fc06 ac15    ................
 64: 0002 0000 0000 0800 2011 e01b 0000 0000    ........ .......
 80: 0000 0000 0000 0000 0000 0000 0000 0000    ................
 96: 0000 0000 0000 0000 0000 0000 0000 0000    ................
112: 0000 0000 0000 0000 0000 0000 0000 0000    ................
128: 0000 0000 0000 0000 0000 0000 0000 0000    ................
144: 0000 0000 0000 0000 0000 0000 0000 0000    ................
160: 0000 0000 0000 0000 0000 0000 0000 0000    ................
176: 0000 0000 0000 0000 0000 0000 0000 0000    ................
192: 0000 0000 0000 0000 0000 0000 0000 0000    ................
208: 0000 0000 0000 0000 0000 0000 0000 0000    ................
224: 0000 0000 0000 0000 0000 0000 0000 0000    ................
240: 0000 0000 0000 0000 0000 0000 0000 0000    ................
256: 0000 0000 0000 0000 0000 0000 0000 0000    ................
272: 0000 0000 0000 6382 5363 3501 0536 04ac    ......c.Sc5..6..
288: 1500 0401 04ff ffff 0003 04c0 a8fc 011c    ................
304: 04c0 a8fc ff40 0964 6863 702e 7465 7374    .....@.dhcp.test
320: 3304 0000 0e10 0204 ffff c7c0 0404 ac15    3...............
336: 0004 0f10 736e 742e 6561 7374 2e73 756e    ....sem.example.
352: 2e63 6f6d 0604 ac15 0001 0c07 7768 6974    com.........whit
368: 652d 362b a602 0481 94ae 1b03 0861 746c    e-6+.........pac
384: 616e 7469 630a 0481 94ae 1b0b 0861 746c    ific.........pac
400: 616e 7469 630f 0578 7465 726d 0435 2f65    ific...xterm.5/e
416: 7870 6f72 742f 7332 382f 6261 7365 2e73    xport/sx2/bcvf.s
432: 3238 735f 776f 732f 6c61 7465 7374 2f53    2xs_btf/latest/S
448: 6f6c 6172 6973 5f38 2f54 6f6f 6c73 2f42    olaris_x/Tools/B
464: 6f6f 740c 202f 6578 706f 7274 2f73 3238    oot. /export/s2x
480: 2f62 6173 652e 7332 3873 5f77 6f73 2f6c    /bcvf.s2xs_btf/l
496: 6174 6573 7407 1b2f 706c 6174 666f 726d    atest../platform
```

EXAMPLE 16–5 Sample snoop Output for One Packet *(Continued)*

```
512: 2f73 756e 346d 2f6b 6572 6e65 6c2f 756e    /sun4u/kernel/un
528: 6978 0807 4553 5435 4544 54ff              ix..EST5EDT.
```

Problems With Inaccurate DHCP Configuration Information

If a DHCP client receives inaccurate information in its network configuration information, look at the DHCP server data. You must examine the option values in the macros that the DHCP server processes for this client. Examples of inaccurate information might be the wrong NIS domain name or router IP address.

Use the following general guidelines to help you determine the source of the inaccurate information:

- Look at the macros defined on the server as described in "How to View Macros Defined on a DHCP Server (DHCP Manager)" on page 314. Review the information in "Order of Macro Processing" on page 240, and determine which macros are processed automatically for this client.
- Look at the network table to determine what macro (if any) is assigned to the client's IP address as the configuration macro. See "Working With IP Addresses in the DHCP Service (Task Map)" on page 297 for more information.
- Take note of any options that occur in more than one macro. Make sure the value that you want for an option is set in the last processed macro.
- Edit the appropriate macro or macros to assure that the correct value is passed to the client. See "Modifying DHCP Macros" on page 314.

Problems With the DHCP Client-Supplied Host Name

This section describes problems that you might experience with DHCP clients that supply their own host names to be registered with DNS.

DHCP Client Does Not Request a Host Name

If your client is not a DHCP client, consult the client's documentation to determine how to configure the client to request a host name. For DHCP clients, see "How to Enable a DHCPv4 Client to Request a Specific Host Name" on page 353.

DHCP Client Does Not Get Requested Host Name

The following list includes describes possible problems a client might have in getting its requested hostname, and suggested solutions.

Problem: Client accepted an offer from a DHCP server that does not issue DNS updates.

Solution: If two DHCP servers are available to the client, the servers should both be configured to provide DNS updates. See "Enabling Dynamic DNS Updates by a DHCP Server" on page 280 for information about configuring the DHCP server and the DNS server.

To determine whether the DHCP server is configured to provide DNS updates:

1. Determine the IP address of the client's DHCP server. On the client system, use `snoop` or another application for capturing network packets. See "How to Use snoop to Monitor DHCP Network Traffic" on page 366, and perform the procedure on the client instead of the server. In the `snoop` output, look for the DHCP Server Identifier to get the IP address of the server.
2. Log in to the DHCP server system to verify that the system is configured to make DNS updates. Type the following command as superuser:

   ```
   dhcpconfig -P
   ```

   If `UPDATE_TIMEOUT` is listed as a server parameter, the DHCP server is configured to make DNS updates.
3. On the DNS server, look at the `/etc/named.conf` file. Find the `allow-update` keyword in the `zone` section of the appropriate domain. If the server allows DNS updates by the DHCP server, the DHCP server's IP address is listed in the `allow-update` keyword.

Problem: Client is using `FQDN` option to specify host name. DHCP does not currently support the `FQDN` option because the option is not officially in the DHCP protocol.

Solution: On the server, use `snoop` or another application for capturing network packets. See "How to Use snoop to Monitor DHCP Network Traffic" on page 366. In the `snoop` output, look for the `FQDN` option in a packet from the client.

Configure the client to specify host name using `Hostname` option. `Hostname` is option code 12. Refer to client documentation for instructions.

For an Oracle Solaris client, see "How to Enable a DHCPv4 Client to Request a Specific Host Name" on page 353

Problem: DHCP server that makes an address offer to the client does not know the client's DNS domain.

Solution: On the DHCP server look for the `DNSdmain` option with a valid value. Set the `DNSdmain` option to the correct DNS domain name in a macro that is processed for this client. `DNSdmain` is usually contained in the network macro. See "Modifying DHCP Macros" on page 314 for information about changing values of options in a macro.

Problem: The host name requested by client corresponds to an IP address that is not managed by the DHCP server. The DHCP server does not perform DNS updates for IP addresses that the server does not manage.

Solution: Check `syslog` for one of the following messages from the DHCP server:

- `There is no` *n.n.n.n* `dhcp-network table for DHCP client's network.`
- `DHCP network record for` *n.n.n.n* `is unavailable, ignoring request.`

Configure the client to request a different name. See "How to Enable a DHCPv4 Client to Request a Specific Host Name" on page 353. Choose a name that is mapped to an address managed by the DHCP server. You can see address mappings in DHCP Manager's Addresses tab. Alternatively, choose an address that is not mapped to any IP address.

Problem: The host name requested by client corresponds to an IP address that is currently not available for use. The address might be in use, leased to another client, or under offer to another client.

Solution: Check `syslog` for the following message from the DHCP server: `ICMP ECHO reply to OFFER candidate:` *n.n.n.n*.

Configure the client to choose a name corresponding to a different IP address. Alternatively, reclaim the address from the client that uses the address.

Problem: DNS server is not configured to accept updates from the DHCP server.

Solution: Examine the `/etc/named.conf` file on the DNS server. Look for the DHCP server's IP address with the `allow-update` keyword in the appropriate `zone` section for the DHCP server's domain. If the IP address is not present, the DNS server is not configured to accept updates from the DHCP server.

See "How to Enable Dynamic DNS Updating for DHCP Clients" on page 281 for information about configuring the DNS server.

If the DHCP server has multiple interfaces, you might need to configure the DNS server to accept updates from all of the DHCP server's addresses. Enable debugging on the DNS server to see whether the updates are reaching the DNS server. If the DNS server received update requests, examine the debugging mode output to determine why the updates did not occur. See the `in.named`.1M man page for information about DNS debugging mode.

Problem: DNS updates might not have completed in the allotted time. DHCP servers do not return host names to clients if the DNS updates have not completed by the configured time limit. However, attempts to complete the DNS updates continue.

Solution: Use the `nslookup` command to determine whether the updates completed successfully. See the `nslookup(1M)` man page.

For example, suppose the DNS domain is `hills.example.org`, and the DNS server's IP address is `10.76.178.11`. The host name that the client wants to register is `cathedral`. You could use the following command to determine if `cathedral` has been registered with that DNS server:

```
nslookup cathedral.hills.example.org 10.76.178.11
```

If the updates completed successfully, but not in the allotted time, you need to increase the time out value. See "How to Enable Dynamic DNS Updating for DHCP Clients" on page 281. In this procedure, you should increase the number of seconds to wait for a response from the DNS server before timing out.

◆ ◆ ◆ CHAPTER 17

DHCP Commands and Files (Reference)

This chapter explains the relationships between the DHCP commands and the DHCP files. However, the chapter does not explain how to use the commands.

The chapter contains the following information:

DHCP Commands

The following table lists the commands that you can use to manage DHCP on your network.

TABLE 17–1 Commands Used in DHCP

Command	Description	Man Page
`dhtadm`	Used to make changes to the options and macros in the `dhcptab`. This command is most useful in scripts that you create to automate changes to your DHCP information. Use `dhtadm` with the `-P` option, and pipe the output through the `grep` command for a quick way to search for particular option values in the `dhcptab` table.	`dhtadm`(1M)
`pntadm`	Used to make changes to the DHCP network tables that map client IDs to IP addresses and optionally associate configuration information with IP addresses.	`pntadm`(1M)
`dhcpconfig`	Used to configure and unconfigure DHCP servers and BOOTP relay agents. Also used to convert to a different data store format, and to import and export DHCP configuration data.	`dhcpconfig`(1M)

TABLE 17–1 Commands Used in DHCP *(Continued)*

Command	Description	Man Page
`in.dhcpd`	The DHCP server daemon. The daemon is started when the system is started. You should not start the server daemon directly. Use DHCP Manager, the `svcadm` command, or `dhcpconfig` to start and stop the daemon. The daemon should be invoked directly only to run the server in debug mode to troubleshoot problems.	`in.dhcpd`(1M)
`dhcpmgr`	The DHCP Manager, a graphical user interface (GUI) tool used to configure and manage the DHCP service. DHCP Manager is the recommended DHCP management tool.	`dhcpmgr`(1M)
`ipadm`	Used at system boot to assign IP addresses to network interfaces, configure network interface parameters, or both. On an DHCP client, `ipadm` starts DHCP to get the parameters (including the IP address) needed to configure a network interface.	`ipadm`(1M)
`dhcpinfo`	Used by system startup scripts on Oracle Solaris client systems to obtain information (such as the host name) from the DHCP client daemon, `dhcpagent`. You can also use `dhcpinfo` in scripts or at the command line to obtain specified parameter values.	`dhcpinfo`(1)
`snoop`	Used to capture and display the contents of packets being passed across the network. `snoop` is useful for troubleshooting problems with the DHCP service.	`snoop`(1M)
`dhcpagent`	The DHCP client daemon, which implements the client side of the DHCP protocol.	`dhcpagent`(1M)

Running DHCP Commands in Scripts

The `dhcpconfig`, `dhtadm`, and `pntadm` commands are optimized for use in scripts. In particular, the `pntadm` command is useful for creating a large number of IP address entries in a DHCP network table. The following sample script uses `pntadm` in batch mode to create IP addresses.

EXAMPLE 17–1 `addclient.ksh` Script With the `pntadm` Command

```
#! /usr/bin/ksh
#
# This script utilizes the pntadm batch facility to add client entries
# to a DHCP network table. It assumes that the user has the rights to
# run pntadm to add entries to DHCP network tables.

#
# Based on the nsswitch setting, query the netmasks table for a netmask.
# Accepts one argument, a dotted IP address.
#
get_netmask()
{
    MTMP=‘getent netmasks ${1} | awk ’{ print $2 }’‘
    if [ ! -z "${MTMP}" ]
```

EXAMPLE 17–1 addclient.ksh Script With the pntadm Command *(Continued)*

```
    then
        print - ${MTMP}
    fi
}

#
# Based on the network specification, determine whether or not network is
# subnetted or supernetted.
# Given a dotted IP network number, convert it to the default class
# network.(used to detect subnetting). Requires one argument, the
# network number. (e.g. 10.0.0.0) Echos the default network and default
# mask for success, null if error.
#
get_default_class()
{
    NN01=${1%%.*}
    tmp=${1#*.}
    NN02=${tmp%%.*}
    tmp=${tmp#*.}
    NN03=${tmp%%.*}
    tmp=${tmp#*.}
    NN04=${tmp%%.*}
    RETNET=""
    RETMASK=""

    typeset -i16 ONE=10#${1%%.*}
    typeset -i10 X=$((${ONE}&16#f0))
    if [ ${X} -eq 224 ]
    then
        # Multicast
        typeset -i10 TMP=$((${ONE}&16#f0))
        RETNET="${TMP}.0.0.0"
        RETMASK="240.0.0.0"
    fi
    typeset -i10 X=$((${ONE}&16#80))
    if [ -z "${RETNET}" -a ${X} -eq 0 ]
    then
        # Class A
        RETNET="${NN01}.0.0.0"
        RETMASK="255.0.0.0"
    fi
    typeset -i10 X=$((${ONE}&16#c0))
    if [ -z "${RETNET}" -a ${X} -eq 128 ]
    then
        # Class B
        RETNET="${NN01}.${NN02}.0.0"
        RETMASK="255.255.0.0"
    fi
    typeset -i10 X=$((${ONE}&16#e0))
    if [ -z "${RETNET}" -a ${X} -eq 192 ]
    then
        # Class C
        RETNET="${NN01}.${NN02}.${NN03}.0"
        RETMASK="255.255.255.0"
    fi
    print - ${RETNET} ${RETMASK}
```

EXAMPLE 17–1 addclient.ksh Script With the pntadm Command *(Continued)*

```
    unset NNO1 NNO2 NNO3 NNO4 RETNET RETMASK X ONE
}

#
# Given a dotted form of an IP address, convert it to its hex equivalent.
#
convert_dotted_to_hex()
{
    typeset -i10 one=${1%%.*}
    typeset -i16 one=${one}
    typeset -Z2 one=${one}
    tmp=${1#*.}

    typeset -i10 two=${tmp%%.*}
    typeset -i16 two=${two}
    typeset -Z2 two=${two}
    tmp=${tmp#*.}

    typeset -i10 three=${tmp%%.*}
    typeset -i16 three=${three}
    typeset -Z2 three=${three}
    tmp=${tmp#*.}

    typeset -i10 four=${tmp%%.*}
    typeset -i16 four=${four}
    typeset -Z2 four=${four}

     hex=‘print - ${one}${two}${three}${four} | sed -e ’s/#/0/g’‘
     print - 16#${hex}
     unset one two three four tmp
}

#
# Generate an IP address given the network address, mask, increment.
#
get_addr()
{
    typeset -i16 net=‘convert_dotted_to_hex ${1}‘
    typeset -i16 mask=‘convert_dotted_to_hex ${2}‘
    typeset -i16 incr=10#${3}

    # Maximum legal value - invert the mask, add to net.
    typeset -i16 mhosts=~${mask}
    typeset -i16 maxnet=${net}+${mhosts}

    # Add the incr value.
    let net=${net}+${incr}

    if [ $((${net} < ${maxnet})) -eq 1 ]
    then
        typeset -i16 a=${net}\&16#ff000000
        typeset -i10 a="${a}>>24"

        typeset -i16 b=${net}\&16#ff0000
        typeset -i10 b="${b}>>16"
```

EXAMPLE 17–1 addclient.ksh Script With the pntadm Command *(Continued)*

```
        typeset -i16 c=${net}\&16#ff00
        typeset -i10 c="${c}>>8"

        typeset -i10 d=${net}\&16#ff
        print - "${a}.${b}.${c}.${d}"
    fi
    unset net mask incr mhosts maxnet a b c d
}

# Given a network address and client address, return the index.
client_index()
{
    typeset -i NNO1=${1%%.*}
    tmp=${1#*.}
    typeset -i NNO2=${tmp%%.*}
    tmp=${tmp#*.}
    typeset -i NNO3=${tmp%%.*}
    tmp=${tmp#*.}
    typeset -i NNO4=${tmp%%.*}

    typeset -i16 NNF1
    let NNF1=${NNO1}
    typeset -i16 NNF2
    let NNF2=${NNO2}
    typeset -i16 NNF3
    let NNF3=${NNO3}
    typeset -i16 NNF4
    let NNF4=${NNO4}
    typeset +i16 NNF1
    typeset +i16 NNF2
    typeset +i16 NNF3
    typeset +i16 NNF4
    NNF1=${NNF1#16\#}
    NNF2=${NNF2#16\#}
    NNF3=${NNF3#16\#}
    NNF4=${NNF4#16\#}
    if [ ${#NNF1} -eq 1 ]
    then
        NNF1="0${NNF1}"
    fi
    if [ ${#NNF2} -eq 1 ]
    then
        NNF2="0${NNF2}"
    fi
    if [ ${#NNF3} -eq 1 ]
    then
        NNF3="0${NNF3}"
    fi
    if [ ${#NNF4} -eq 1 ]
    then
        NNF4="0${NNF4}"
    fi
    typeset -i16 NN
    let NN=16#${NNF1}${NNF2}${NNF3}${NNF4}
    unset NNF1 NNF2 NNF3 NNF4
```

EXAMPLE 17–1 addclient.ksh Script With the pntadm Command *(Continued)*

```
    typeset -i NNO1=${2%%.*}
    tmp=${2#*.}
    typeset -i NNO2=${tmp%%.*}
    tmp=${tmp#*.}
    typeset -i NNO3=${tmp%%.*}
    tmp=${tmp#*.}
    typeset -i NNO4=${tmp%%.*}
    typeset -i16 NNF1
    let NNF1=${NNO1}
    typeset -i16 NNF2
    let NNF2=${NNO2}
    typeset -i16 NNF3
    let NNF3=${NNO3}
    typeset -i16 NNF4
    let NNF4=${NNO4}
    typeset +i16 NNF1
    typeset +i16 NNF2
    typeset +i16 NNF3
    typeset +i16 NNF4
    NNF1=${NNF1#16\#}
    NNF2=${NNF2#16\#}
    NNF3=${NNF3#16\#}
    NNF4=${NNF4#16\#}
    if [ ${#NNF1} -eq 1 ]
    then
        NNF1="0${NNF1}"
    fi
    if [ ${#NNF2} -eq 1 ]
    then
        NNF2="0${NNF2}"
    fi
    if [ ${#NNF3} -eq 1 ]
    then
        NNF3="0${NNF3}"
    fi
    if [ ${#NNF4} -eq 1 ]
    then
        NNF4="0${NNF4}"
    fi
    typeset -i16 NC
    let NC=16#${NNF1}${NNF2}${NNF3}${NNF4}
    typeset -i10 ANS
    let ANS=${NC}-${NN}
    print - $ANS
}

#
# Check usage.
#
if [ "$#" != 3 ]
then
    print "This script is used to add client entries to a DHCP network"
    print "table by utilizing the pntadm batch facilty.\n"
    print "usage: $0 network start_ip entries\n"
    print "where: network is the IP address of the network"
        print "       start_ip is the starting IP address \n"
```

EXAMPLE 17–1 addclient.ksh Script With the pntadm Command *(Continued)*

```
        print "       entries is the number of the entries to add\n"
    print "example: $0 10.148.174.0 10.148.174.1 254\n"
    return
fi

#
# Use input arguments to set script variables.
#
NETWORK=$1
START_IP=$2
typeset -i STRTNUM=‘client_index ${NETWORK} ${START_IP}‘
let ENDNUM=${STRTNUM}+$3
let ENTRYNUM=${STRTNUM}
BATCHFILE=/tmp/batchfile.$$
MACRO=‘uname -n‘

#
# Check if mask in netmasks table. First try
# for network address as given, in case VLSM
# is in use.
#
NETMASK=‘get_netmask ${NETWORK}‘
if [ -z "${NETMASK}" ]
then
    get_default_class ${NETWORK} | read DEFNET DEFMASK
    # use the default.
    if [ "${DEFNET}" != "${NETWORK}" ]
    then
        # likely subnetted/supernetted.
        print - "\n\n###\tWarning\t###\n"
        print - "Network ${NETWORK} is netmasked, but no entry was found  \n
              in the ’netmasks’ table; please update the ’netmasks’  \n
              table in the appropriate nameservice before continuing. \n
              (See /etc/nsswitch.conf.) \n" >&2
        return 1
    else
        # use the default.
        NETMASK="${DEFMASK}"
    fi
fi

#
# Create a batch file.
#
print -n "Creating batch file "
while [ ${ENTRYNUM} -lt ${ENDNUM} ]
do
    if [ $((${ENTRYNUM}-${STRTNUM}))%50 -eq 0 ]
    then
        print -n "."
    fi

    CLIENTIP=‘get_addr ${NETWORK} ${NETMASK} ${ENTRYNUM}‘
    print "pntadm -A ${CLIENTIP} -m ${MACRO} ${NETWORK}" >> ${BATCHFILE}
    let ENTRYNUM=${ENTRYNUM}+1
done
```

EXAMPLE 17–1 `addclient.ksh` Script With the `pntadm` Command *(Continued)*

```
print " done.\n"

#
# Run pntadm in batch mode and redirect output to a temporary file.
# Progress can be monitored by using the output file.
#
print "Batch processing output redirected to ${BATCHFILE}"
print "Batch processing started."

pntadm -B ${BATCHFILE} -v > /tmp/batch.out 2 >&1

print "Batch processing completed."
```

Files Used by the DHCP Service

The following table lists files associated with DHCP.

TABLE 17–2 Files and Tables Used by DHCP Daemons and Commands

File or Table Name	Description	Man Page
`dhcptab`	A generic term for the table of DHCP configuration information that is recorded as options with assigned values, which are then grouped into macros. The name of the `dhcptab` table and its location is determined by the data store you use for DHCP information.	`dhcptab(4)`
DHCP network table	Maps IP addresses to client IDs and configuration options. DHCP network tables are named according to the IP address of the network, such as `10.21.32.0`. There is no file that is called `dhcp_network`. The name and location of DHCP network tables is determined by the data store you use for DHCP information.	`dhcp_network(4)`
`dhcpsvc.conf`	Stores startup options for the DHCP daemon and data store information. This file must not be edited manually. Use the `dhcpconfig` command to change startup options.	`dhcpsvc.conf(4)`
`nsswitch.conf`	Specifies the location of name service databases and the order in which to search name services for various kinds of information. The `nsswitch.conf` file is read to obtain accurate configuration information when you configure a DHCP server. The file is located in the `/etc` directory.	`nsswitch.conf(4)`
`resolv.conf`	Contains information used to resolve DNS queries. During DHCP server configuration, this file is consulted for information about the DNS domain and DNS server. The file is located in the `/etc` directory.	`resolv.conf(4)`

TABLE 17–2 Files and Tables Used by DHCP Daemons and Commands *(Continued)*

File or Table Name	Description	Man Page
`dhcp.`*interface*	Indicates that DHCP is to be used on the client's network interface that is specified in the `dhcp.`*interface* file name.	No specific man page, see `dhcp(5)`
interface`.dhc`	Contains the configuration parameters that are obtained from DHCP for the given network interface. The client caches the current configuration information in `/etc/dhcp/`*interface*`.dhc` when the interface's IP address lease is dropped. For example, if DHCP is used on the `qe0` interface, the `dhcpagent` caches the configuration information in `/etc/dhcp/qe0.dhc`. The next time DHCP starts on the interface, the client requests to use the cached configuration if the lease has not expired. If the DHCP server denies the request, the client begins the standard process for DHCP lease negotiation.	No specific man page, see `dhcpagent(1M)`
`dhcpagent`	Sets parameter values for the `dhcpagent` client daemon. The path to the file is `/etc/default/dhcpagent`. See the `/etc/default/dhcpagent` file or the `dhcpagent(1M)` man page for information about the parameters.	`dhcpagent(1M)`
DHCP `inittab`	Defines aspects of DHCP option codes, such as the data type, and assigns mnemonic labels. See the `dhcp_inittab(4)` man page for more information about the file syntax. On the client, the information in the `/etc/dhcp/inittab` file is used by `dhcpinfo` to provide more meaningful information to human readers of the information. On the DHCP server system, this file is used by the DHCP daemon and management tools to obtain DHCP option information. The `/etc/dhcp/inittab` file replaces the `/etc/dhcp/dhcptags` file that was used in previous releases. "DHCP Option Information" on page 385 provides more information about this replacement.	`dhcp_inittab(4)`

DHCP Option Information

Historically, DHCP option information has been stored in several places, including the server's `dhcptab` table, the client's `dhcptags` file, and internal tables of various programs. In the Solaris 8 release and later releases, the option information is consolidated in the `/etc/dhcp/inittab` file. See the `dhcp_inittab(4)` man page for detailed information about the file.

The DHCP client uses the DHCP `inittab` file as a replacement for the `dhcptags` file. The client uses the file to obtain information about option codes that were received in a DHCP packet. The `in.dhcpd`, `snoop`, and `dhcpmgr` programs on the DHCP server use the `inittab` file as well.

Determining if Your Site Is Affected

Most sites that use DHCP are *not* affected by the switch to the `/etc/dhcp/inittab` file. Your site is affected if you meet all of the following criteria:

- You previously created new DHCP options.
- You modified the `/etc/dhcp/dhcptags` file, and you want to retain the changes.

When you upgrade, the upgrade log notifies you that your `dhcptags` file had been modified and that you should make changes to the DHCP `inittab` file.

Differences Between `dhcptags` and `inittab` Files

The `inittab` file contains more information than the `dhcptags` file. The `inittab` file also uses a different syntax.

A sample `dhcptags` entry is as follows:

```
33 StaticRt - IPList Static_Routes
```

`33` is the numeric code that is passed in the DHCP packet. `StaticRt` is the option name. `IPList` indicates that the data type for `StaticRt` must be a list of IP addresses. `Static_Routes` is a more descriptive name.

The `inittab` file consists of one-line records that describe each option. The format is similar to the format that defines symbols in `dhcptab`. The following table describes the syntax of the `inittab` file.

Option	Description
option-name	Name of the option. The option name must be unique within its option category, and not overlap with other option names in the Standard, Site, and Vendor categories. For example, you cannot have two Site options with the same name, and you should not create a Site option with the same name as a Standard option.
category	Identifies the namespace in which the option belongs. Must be one of the following: Standard, Site, Vendor, Field, or Internal.
code	Identifies the option when sent over the network. In most cases, the code uniquely identifies the option, without a category. However, in the case of internal categories such as Field or Internal, a code might be used for other purposes. The code might not be globally unique. The code should be unique within the option's category, and not overlap with codes in the Standard and Site fields.

type Describes the data that is associated with this option. Valid types are IP, ASCII, Octet, Boolean, Unumber8, Unumber16, Unumber32, Unumber64, Snumber8, Snumber16, Snumber32, and Snumber64. For numbers, an initial U or S indicates that the number is unsigned or signed. The digits at the end indicate how many bits are in the number. For example, Unumber8 is an unsigned 8-bit number. The type is not case sensitive.

granularity Describes how many units of data make up a whole value for this option.

maximum Describes how many whole values are allowed for this option. 0 indicates an infinite number.

consumers Describes which programs can use this information. Consumers should be set to `sdmi`, where:

```
s    snoop
d    in.dhcpd
m    dhcpmgr
i    dhcpinfo
```

A sample `inittab` entry is as follows:

```
StaticRt - Standard, 33, IP, 2, 0, sdmi
```

This entry describes an option that is named `StaticRt`. The option is in the Standard category, and is option code 33. The expected data is a potentially infinite number of pairs of IP addresses because the type is `IP`, the granularity is `2`, and the maximum is infinite (`0`). The consumers of this option are `sdmi`: `snoop`, `in.dhcpd`, `dhcpmgr`, and `dhcpinfo`.

Converting `dhcptags` Entries to `inittab` Entries

If you previously added entries to your `dhcptags` file, you must add corresponding entries to the new `inittab` file if you want to continue using the options you added to your site. The following example shows how a sample `dhcptags` entry might be expressed in `inittab` format.

Suppose you had added the following `dhcptags` entry for fax machines that are connected to the network:

```
128 FaxMchn - IP Fax_Machine
```

The code `128` means that the option must be in the Site category. The option name is `FaxMchn`, and the data type is `IP`.

The corresponding `inittab` entry might be:

```
FaxMchn SITE, 128, IP, 1, 1, sdmi
```

The granularity of 1 and the maximum of 1 indicate that one IP address is expected for this option.

PART III

IP Security

This section focuses on network security. IP security architecture (IPsec) protects the network at the packet level. Internet key management (IKE) manages the keys for IPsec. IP filter provides a firewall.

◆ ◆ ◆ CHAPTER 18

IP Security Architecture (Overview)

The IP Security Architecture (IPsec) provides cryptographic protection for IP datagrams in IPv4 and IPv6 network packets.

This chapter contains the following information:

To implement IPsec on your network, see Chapter 19, "Configuring IPsec (Tasks)." For reference information, see Chapter 20, "IP Security Architecture (Reference)."

Introduction to IPsec

IPsec protects IP packets by authenticating the packets, by encrypting the packets, or by doing both. IPsec is performed inside the IP module, well below the application layer. Therefore, an Internet application can take advantage of IPsec while not having to configure itself to use IPsec. When used properly, IPsec is an effective tool in securing network traffic.

IPsec protection involves five main components:

- **Security protocols –** The IP datagram protection mechanisms. The authentication header (AH) signs IP packets and ensures integrity. The content of the datagram is not encrypted, but the receiver is assured that the packet contents have not been altered. The receiver is also assured that the packets were sent by the sender. The encapsulating security payload (ESP) encrypts IP data, thus obscuring the content during packet transmission. ESP also can ensure data integrity through an authentication algorithm option.
- **Security associations database (SADB) –** The database that associates a security protocol with an IP destination address and an indexing number. The indexing number is called the security parameter index (SPI). These three elements (the security protocol, the destination address, and the SPI) uniquely identify a legitimate IPsec packet. The database ensures that a protected packet that arrives to the packet destination is recognized by the receiver. The receiver also uses information from the database to decrypt the communication, verify that the packets are unchanged, reassemble the packets, and deliver the packets to their ultimate destination.
- **Key management –** The generation and distribution of keys for the cryptographic algorithms and for the SPI.
- **Security mechanisms –** The authentication and encryption algorithms that protect the data in the IP datagrams.
- **Security policy database (SPD) –** The database that specifies the level of protection to apply to a packet. The SPD filters IP traffic to determine how the packets should be processed. A packet can be discarded. A packet can be passed in the clear. Or, a packet can be protected with IPsec. For outbound packets, the SPD and the SADB determine what level of protection to apply. For inbound packets, the SPD helps to determine if the level of protection on the packet is acceptable. If the packet is protected by IPsec, the SPD is consulted after the packet has been decrypted and has been verified.

IPsec applies the security mechanisms to IP datagrams that travel to the IP destination address. The receiver uses information in its SADB to verify that the arriving packets are legitimate and to decrypt them. Applications can invoke IPsec to apply security mechanisms to IP datagrams on a per-socket level as well.

Note that sockets behave differently from ports:

- Per-socket SAs override their corresponding port entry in the SPD.
- Also, if a socket on a port is connected, and IPsec policy is later applied to that port, then traffic that uses that socket is not protected by IPsec.

  Of course, a socket that is opened on a port *after* IPsec policy is applied to the port is protected by IPsec policy.

IPsec RFCs

The Internet Engineering Task Force (IETF) has published a number of Requests for Comment (RFCs) that describe the security architecture for the IP layer. All RFCs are copyrighted by the Internet Society. For a link to the RFCs, see http://www.ietf.org/. The following list of RFCs covers the more general IP security references:

- RFC 2411, "IP Security Document Roadmap," November 1998
- RFC 2401, "Security Architecture for the Internet Protocol," November 1998
- RFC 2402, "IP Authentication Header," November 1998
- RFC 2406, "IP Encapsulating Security Payload (ESP)," November 1998
- RFC 2408, "Internet Security Association and Key Management Protocol (ISAKMP)," November 1998
- RFC 2407, "The Internet IP Security Domain of Interpretation for ISAKMP," November 1998
- RFC 2409, "The Internet Key Exchange (IKE)," November 1998
- RFC 3554, "On the Use of Stream Control Transmission Protocol (SCTP) with IPsec," July 2003

IPsec Terminology

The IPsec RFCs define a number of terms that are useful to recognize when implementing IPsec on your systems. The following table lists IPsec terms, provides their commonly used acronyms, and defines each term. For a list of terminology used in key negotiation, see Table 21–1.

TABLE 18–1 IPsec Terms, Acronyms, and Uses

IPsec Term	Acronym	Definition
Security association	SA	A unique connection between two nodes on a network. The connection is defined by a triplet: a security protocol, a security parameter index, and an IP destination. The IP destination can be an IP address or a socket.
Security associations database	SADB	Database that contains all active security associations.
Security parameter index	SPI	The indexing value for a security association. An SPI is a 32-bit value that distinguishes among SAs that have the same IP destination and security protocol.
Security policy database	SPD	Database that determines if outbound packets and inbound packets have the specified level of protection.

TABLE 18–1 IPsec Terms, Acronyms, and Uses *(Continued)*

IPsec Term	Acronym	Definition
Key exchange		The process of generating keys for asymmetric cryptographic algorithms. The two main methods are RSA protocols and the Diffie-Hellman protocol.
Diffie-Hellman protocol	DH	A key exchange protocol that involves key generation and key authentication. Often called *authenticated key exchange*.
RSA protocol	RSA	A key exchange protocol that involves key generation and key distribution. The protocol is named for its three creators, Rivest, Shamir, and Adleman.
Internet Security Association and Key Management Protocol	ISAKMP	The common framework for establishing the format of SA attributes, and for negotiating, modifying, and deleting SAs. ISAKMP is the IETF standard for handling IPsec SAs.

IPsec Packet Flow

Figure 18–1 shows how an IP addressed packet, as part of an IP datagram, proceeds when IPsec has been invoked on an outbound packet. The flow diagram illustrates where authentication header (AH) and encapsulating security payload (ESP) entities can be applied to the packet. How to apply these entities, as well as how to choose the algorithms, are described in subsequent sections.

Figure 18–2 shows the IPsec inbound process.

FIGURE 18–1 IPsec Applied to Outbound Packet Process

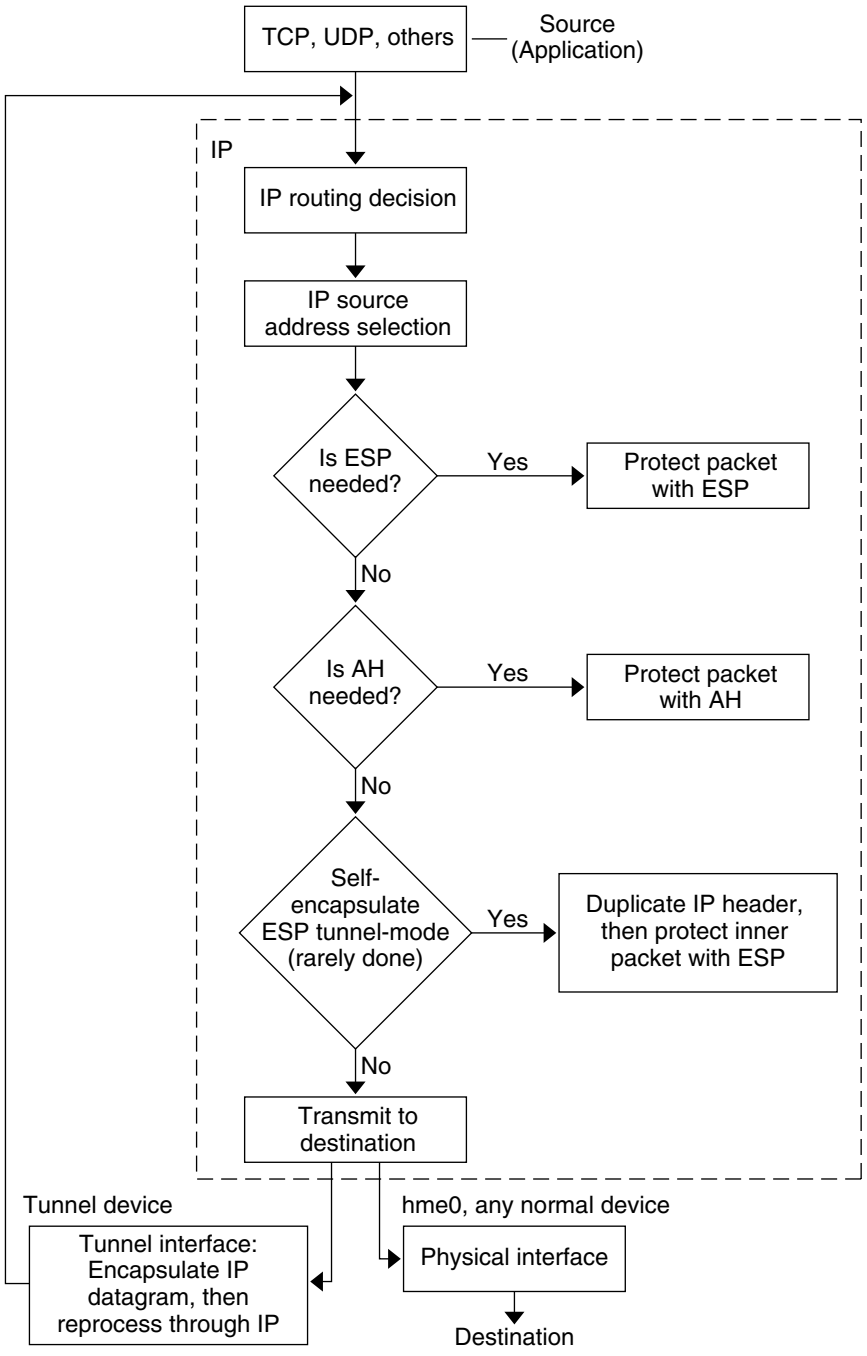
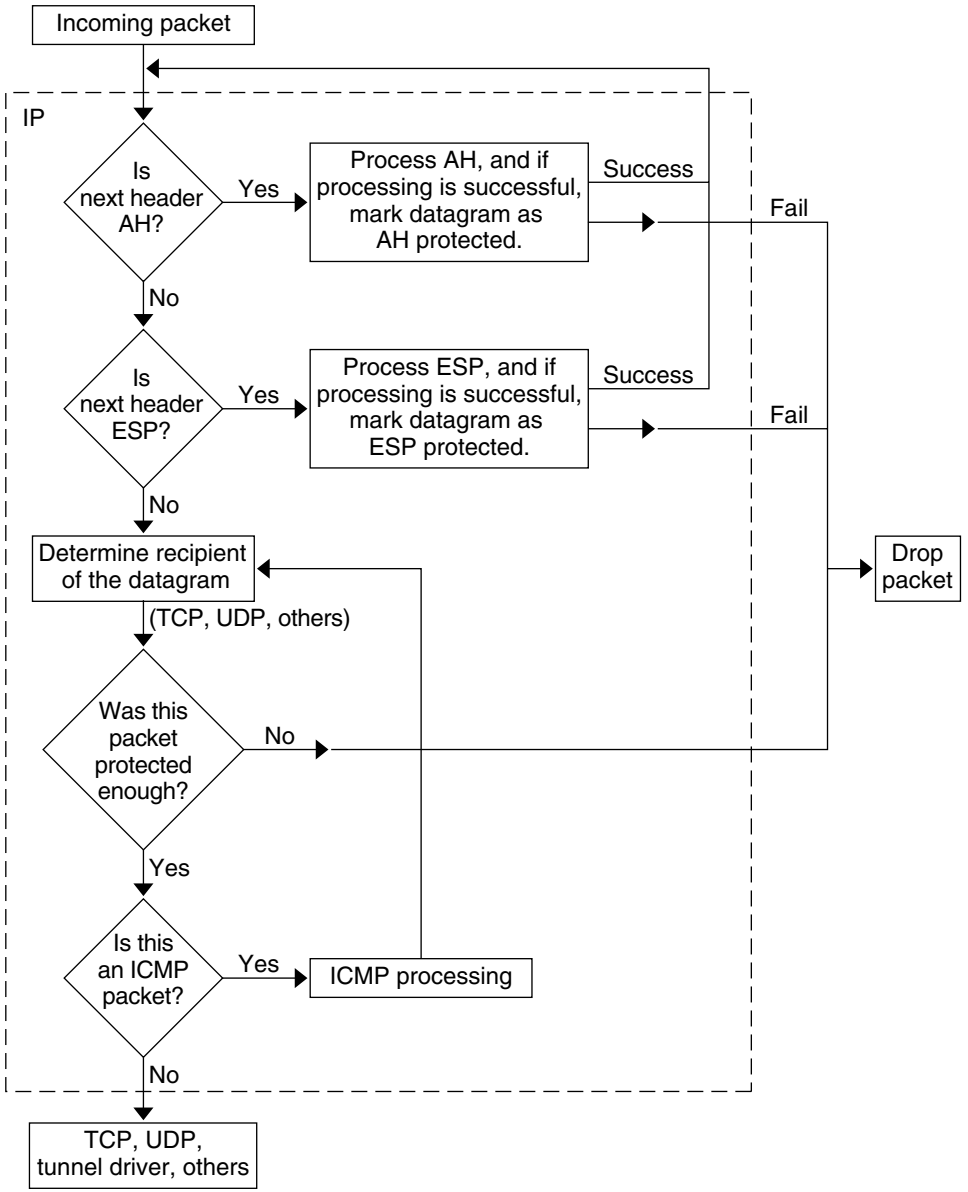


FIGURE 18–2 IPsec Applied to Inbound Packet Process



IPsec Security Associations

An IPsec *security association* (SA) specifies security properties that are recognized by communicating hosts. A single SA protects data in one direction. The protection is either to a single host or to a group (multicast) address. Because most communication is either peer-to-peer or client-server, two SAs must be present to secure traffic in both directions.

The following three elements uniquely identify an IPsec SA:

- The security protocol (AH or ESP)
- The destination IP address
- The security parameter index (SPI)

The SPI, an arbitrary 32-bit value, is transmitted with an AH or ESP packet. The `ipsecah(7P)` and `ipsecesp(7P)` man pages explain the extent of protection that is provided by AH and ESP. An integrity checksum value is used to authenticate a packet. If the authentication fails, the packet is dropped.

Security associations are stored in a *security associations database* (SADB). A socket-based administration engine, the `PF_KEY` interface enables privileged applications to manage the database. For example, the IKE application and the `ipseckeys` command use the `PF_KEY` socket interface.

- For a more complete description of the IPsec SADB, see "Security Associations Database for IPsec" on page 449.
- For more information about how to manage the SADB, see the `pf_key(7P)` man page.

Key Management in IPsec

Security associations (SAs) require keying material for authentication and for encryption. The managing of this *keying material* is called *key management*. The Internet Key Exchange (IKE) protocol handles key management automatically. You can also manage keys manually with the `ipseckey` command.

SAs on IPv4 and IPv6 packets can use either method of key management. Unless you have an overriding reason to use manual key management, automatic key management is preferred. For example, to interoperate with systems other than Solaris systems might require manual key management.

In the current release, SMF provides the following key management services for IPsec:

- `svc:/network/ipsec/ike:default` **service** – Is the SMF service for automatic key management. The `ike` service runs the `in.iked` daemon to provide automatic key management. For a description of IKE, see Chapter 21, "Internet Key Exchange (Overview)." For more information about the `in.iked` daemon, see the `in.iked`(1M) man page. For information about the `ike` service, see the "IKE Service Management Facility" on page 501.
- `svc:/network/ipsec/manual-key:default` **service** – Is the SMF service for manual key management. The `manual-key` service runs the `ipseckey` command with various options to manage keys manually. For a description of the `ipseckey` command, see "Utilities for Key Generation in IPsec" on page 449. For a detailed description of the `ipseckey` command options, see the `ipseckey`(1M) man page.

IPsec Protection Mechanisms

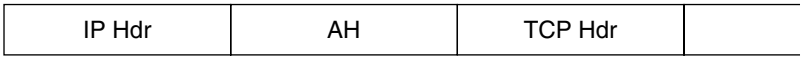
IPsec provides two security protocols for protecting data:

- Authentication Header (AH)
- Encapsulating Security Payload (ESP)

An AH protects data with an authentication algorithm. An ESP protects data with an encryption algorithm. Optionally, an ESP protects data with an authentication algorithm. Each implementation of an algorithm is called a *mechanism*.

Authentication Header

The authentication header provides data authentication, strong integrity, and replay protection to IP datagrams. AH protects the greater part of the IP datagram. As the following illustration shows, AH is inserted between the IP header and the transport header.



The transport header can be TCP, UDP, SCTP, or ICMP. If a tunnel is being used, the transport header can be another IP header.

Encapsulating Security Payload

The encapsulating security payload (ESP) module provides confidentiality over what the ESP encapsulates. ESP also provides the services that AH provides. However, ESP only provides its protections over the part of the datagram that ESP encapsulates. ESP provides optional

authentication services to ensure the integrity of the protected packet. Because ESP uses encryption-enabling technology, a system that provides ESP can be subject to import and export control laws.

ESP encapsulates its data, so ESP only protects the data that follows its beginning in the datagram, as shown in the following illustration.

Encrypted

In a TCP packet, ESP encapsulates only the TCP header and its data. If the packet is an IP-in-IP datagram, ESP protects the inner IP datagram. Per-socket policy allows *self-encapsulation*, so ESP can encapsulate IP options when ESP needs to.

If self-encapsulation is set, a copy of the IP header is made to construct an IP-in-IP datagram. For example, when self-encapsulation is not set on a TCP socket, the datagram is sent in the following format:

```
[ IP(a -> b) options + TCP + data ]
```

When self-encapsulation is set on that TCP socket, the datagram is sent in the following format:

```
[ IP(a -> b) + ESP [ IP(a -> b) options + TCP + data ] ]
```

For further discussion, see "Transport and Tunnel Modes in IPsec" on page 401.

Security Considerations When Using AH and ESP

The following table compares the protections that are provided by AH and ESP.

TABLE 18–2 Protections Provided by AH and ESP in IPsec

Protocol	Packet Coverage	Protection	Against Attacks
AH	Protects packet from the IP header to the transport header	Provides strong integrity, data authentication: ■ Ensures that the receiver receives exactly what the sender sent ■ Is susceptible to replay attacks when an AH does not enable replay protection	Replay, cut-and-paste

TABLE 18–2 Protections Provided by AH and ESP in IPsec *(Continued)*

Protocol	Packet Coverage	Protection	Against Attacks
ESP	Protects packet following the beginning of ESP in the datagram.	With encryption option, encrypts the IP datagram. Ensures confidentiality	Eavesdropping
		With authentication option, provides the same protection as AH	Replay, cut-and-paste
		With both options, provides strong integrity, data authentication, and confidentiality	Replay, cut-and-paste, eavesdropping

Authentication and Encryption Algorithms in IPsec

IPsec security protocols use two types of algorithms, authentication and encryption. The AH module uses authentication algorithms. The ESP module can use encryption as well as authentication algorithms. You can obtain a list of the algorithms on your system and their properties by using the `ipsecalgs` command. For more information, see the `ipsecalgs(1M)` man page. You can also use the functions that are described in the `getipsecalgbyname(3NSL)` man page to retrieve the properties of algorithms.

IPsec on a Solaris system uses the Solaris cryptographic framework to access the algorithms. The framework provides a central repository for algorithms, in addition to other services. The framework enables IPsec to take advantage of high performance cryptographic hardware accelerators. The framework also provides resource control features. For example, the framework enables you to limit the amount of CPU time spent in cryptographic operations in the kernel.

For more information, see the following:

- Chapter 13, "Oracle Solaris Cryptographic Framework (Overview)," in *System Administration Guide: Security Services*
- Chapter 8, "Introduction to the Oracle Solaris Cryptographic Framework," in *Developer's Guide for Oracle Solaris Security*

Authentication Algorithms in IPsec

Authentication algorithms produce an integrity checksum value or *digest* that is based on the data and a key. The AH module uses authentication algorithms. The ESP module can use authentication algorithms as well.

Encryption Algorithms in IPsec

Encryption algorithms encrypt data with a key. The ESP module in IPsec uses encryption algorithms. The algorithms operate on data in units of a *block size*.

IPsec Protection Policies

IPsec protection policies can use any of the security mechanisms. IPsec policies can be applied at the following levels:

- On a system-wide level
- On a per-socket level

IPsec applies the system-wide policy to outbound datagrams and inbound datagrams. Outbound datagrams are either sent with protection or without protection. If protection is applied, the algorithms are either specific or non-specific. You can apply some additional rules to outbound datagrams, because of the additional data that is known by the system. Inbound datagrams can be either accepted or dropped. The decision to drop or accept an inbound datagram is based on several criteria, which sometimes overlap or conflict. Conflicts are resolved by determining which rule is parsed first. The traffic is automatically accepted, except when a policy entry states that traffic should bypass all other policies.

The policy that normally protects a datagram can be bypassed. You can either specify an exception in the system-wide policy, or you can request a bypass in the per-socket policy. For traffic within a system, policies are enforced, but actual security mechanisms are not applied. Instead, the outbound policy on an intra-system packet translates into an inbound packet that has had those mechanisms applied.

You use the `ipsecinit.conf` file and the `ipsecconf` command to configure IPsec policies. For details and examples, see the `ipsecconf(1M)` man page.

Transport and Tunnel Modes in IPsec

The IPsec standards define two distinct modes of IPsec operation, *transport mode* and *tunnel mode*. The modes do not affect the encoding of packets. The packets are protected by AH, ESP, or both in each mode. The modes differ in policy application when the inner packet is an IP packet, as follows:

- In transport mode, the outer header determines the IPsec policy that protects the inner IP packet.
- In tunnel mode, the inner IP packet determines the IPsec policy that protects its contents.

In transport mode, the outer header, the next header, and any ports that the next header supports, can be used to determine IPsec policy. In effect, IPsec can enforce different transport mode policies between two IP addresses to the granularity of a single port. For example, if the next header is TCP, which supports ports, then IPsec policy can be set for a TCP port of the outer IP address. Similarly, if the next header is an IP header, the outer header and the inner IP header can be used to determine IPsec policy.

Tunnel mode works only for IP-in-IP datagrams. Tunneling in tunnel mode can be useful when computer workers at home are connecting to a central computer location. In tunnel mode,

IPsec policy is enforced on the contents of the inner IP datagram. Different IPsec policies can be enforced for different inner IP addresses. That is, the inner IP header, its next header, and the ports that the next header supports, can enforce a policy. Unlike transport mode, in tunnel mode the outer IP header does not dictate the policy of its inner IP datagram.

Therefore, in tunnel mode, IPsec policy can be specified for subnets of a LAN behind a router and for ports on those subnets. IPsec policy can also be specified for particular IP addresses, that is, hosts, on those subnets. The ports of those hosts can also have a specific IPsec policy. However, if a dynamic routing protocol is run over a tunnel, do not use subnet selection or address selection because the view of the network topology on the peer network could change. Changes would invalidate the static IPsec policy. For examples of tunneling procedures that include configuring static routes, see "Protecting a VPN With IPsec" on page 424.

In the Solaris OS, tunnel mode can be enforced only on an IP tunneling network interface. For information about tunneling interfaces, see Chapter 7, "Configuring IP Tunnels." The `ipsecconf` command provides a `tunnel` keyword to select an IP tunneling network interface. When the `tunnel` keyword is present in a rule, all selectors that are specified in that rule apply to the inner packet.

In transport mode, ESP, AH, or both, can protect the datagram.

The following figure shows an IP header with an unprotected TCP packet.

FIGURE 18–3 Unprotected IP Packet Carrying TCP Information

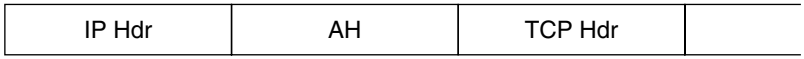
In transport mode, ESP protects the data as shown in the following figure. The shaded area shows the encrypted part of the packet.

FIGURE 18–4 Protected IP Packet Carrying TCP Information

Encrypted

In transport mode, AH protects the data as shown in the following figure.

FIGURE 18–5 Packet Protected by an Authentication Header



AH actually covers the data before the data appears in the datagram. Consequently, the protection that is provided by AH, even in transport mode, covers some of the IP header.

In tunnel mode, the entire datagram is *inside* the protection of an IPsec header. The datagram in Figure 18–3 is protected in tunnel mode by an outer IPsec header, and in this case ESP, as is shown in the following figure.

FIGURE 18–6 IPsec Packet Protected in Tunnel Mode

Encrypted

The `ipsecconf` command includes keywords to set tunnels in tunnel mode or transport mode.

- For details on per-socket policy, see the `ipsec(7P)` man page.
- For an example of per-socket policy, see "How to Use IPsec to Protect a Web Server From Nonweb Traffic" on page 414.
- For more information about tunnels, see the `ipsecconf(1M)` man page.
- For an example of tunnel configuration, see "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode" on page 429.

Virtual Private Networks and IPsec

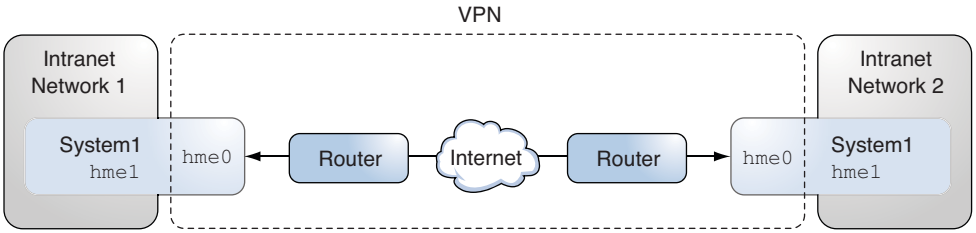
A configured tunnel is a point-to-point interface. The tunnel enables one IP packet to be encapsulated within another IP packet. A correctly configured tunnel requires both a tunnel source and a tunnel destination. For more information, see "How to Create and Configure an IP Tunnel" on page 161.

A tunnel creates an apparent physical interface to IP. The physical link's integrity depends on the underlying security protocols. If you set up the security associations (SAs) securely, then you can trust the tunnel. Packets that exit the tunnel must have originated from the peer that was specified in the tunnel destination. If this trust exists, you can use per-interface IP forwarding to create a virtual private network (VPN).

You can use IPsec to construct a VPN. IPsec secures the connection. For example, an organization that uses VPN technology to connect offices with separate networks can deploy IPsec to secure traffic between the two offices.

The following figure illustrates how two offices use the Internet to form their VPN with IPsec deployed on their network systems.

FIGURE 18–7 Virtual Private Network



For a detailed example of the setup procedure, see "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode" on page 429.

The procedures include examples of IPv6 syntax.

IPsec and NAT Traversal

IKE can negotiate IPsec SAs across a NAT box. This ability enables systems to securely connect from a remote network, even when the systems are behind a NAT device. For example, employees who work from home, or who log on from a conference site can protect their traffic with IPsec.

NAT stands for network address translation. A NAT box is used to translate a private internal address into a unique Internet address. NATs are very common at public access points to the Internet, such as hotels. For a fuller discussion, see "Using IP Filter's NAT Feature" on page 516.

The ability to use IKE when a NAT box is between communicating systems is called NAT traversal, or NAT-T. NAT-T has the following limitations:

- NAT-T cannot take advantage of the IPsec ESP acceleration provided by the Sun Crypto Accelerator 4000 board. However, IKE acceleration with the Sun Crypto Accelerator 4000 board works.
- The AH protocol depends on an unchanging IP header, therefore AH cannot work with NAT-T. The ESP protocol is used with NAT-T.
- The NAT box does not use special processing rules. A NAT box with special IPsec processing rules might interfere with the implementation of NAT-T.
- NAT-T works only when the IKE initiator is the system behind the NAT box. An IKE responder cannot be behind a NAT box unless the box has been programmed to forward IKE packets to the appropriate individual system behind the box.

The following RFCs describe NAT functionality and the limits of NAT-T. Copies of the RFCs can be retrieved from `http://www.rfc-editor.org`.

- RFC 3022, "Traditional IP Network Address Translator (Traditional NAT)," January 2001
- RFC 3715, "IPsec-Network Address Translation (NAT) Compatibility Requirements," March 2004
- RFC 3947, "Negotiation of NAT-Traversal in the IKE," January 2005
- RFC 3948, "UDP Encapsulation of IPsec Packets," January 2005

To use IPsec across a NAT, see "Configuring IKE for Mobile Systems (Task Map)" on page 487.

IPsec and SCTP

The Solaris OS supports the Streams Control Transmission Protocol (SCTP). The use of the SCTP protocol and SCTP port number to specify IPsec policy is supported, but is not robust. The IPsec extensions for SCTP as specified in RFC 3554 are not yet implemented. These limitations can create complications in creating IPsec policy for SCTP.

SCTP can make use of multiple source and destination addresses in the context of a single SCTP association. When IPsec policy is applied to a single source or a single destination address, communication can fail when SCTP switches the source or the destination address of that association. IPsec policy only recognizes the original address. For information about SCTP, read the RFCs and "SCTP Protocol" in *System Administration Guide: IP Services*.

IPsec and Solaris Zones

For shared-IP zones, IPsec is configured from the global zone. The IPsec policy configuration file, `ipsecinit.conf`, exists in the global zone only. The file can have entries that apply to non-global zones, as well as entries that apply to the global zone.

For exclusive-IP zones, IPsec is configured in the non-global zone.

For information about how to use IPsec with zones, see "Protecting Traffic With IPsec" on page 410. For information about zones, see Chapter 15, "Introduction to Oracle Solaris Zones," in *System Administration Guide: Oracle Solaris Zones, Oracle Solaris 10 Containers, and Resource Management*.

IPsec and Logical Domains

IPsec works with logical domains. The logical domain must be running a version of the Solaris OS that includes IPsec, such as the Solaris 10 release.

To create logical domains, you must use the Oracle VM Server for SPARC, which was previously called Logical Domains. For information about how to configure logical domains, see *Logical Domains 1.2 Administration Guide* or *Oracle VM Server for SPARC 2.0 Administration Guide*.

IPsec Utilities and Files

Table 18–3 describes the files, commands, and service identifiers that are used to configure and manage IPsec. For completeness, the table includes key management files, socket interfaces, and commands.

For more information about service identifiers, see Chapter 11, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*.

- For instructions on implementing IPsec on your network, see "Protecting Traffic With IPsec (Task Map)" on page 409.
- For more details about IPsec utilities and files, see Chapter 20, "IP Security Architecture (Reference)."

TABLE 18–3 List of Selected IPsec Utilities and Files

IPsec Utility, File, or Service	Description	Man Page
`svc:/network/ipsec/ipsecalgs`	The SMF service that manages IPsec algorithms.	`smf(5)`, `ipsecalgs(1M)`
`svc:/network/ipsec/manual-key`	The SMF service that manages manual security associations (SAs).	`smf(5)`, `ipseckey(1M)`
`svc:/network/ipsec/policy`	The SMF service that manages IPsec policy.	`smf(5)`, `ipsecconf(1M)`
`svc:/network/ipsec/ike`	The SMF service for the automatic management of IPsec SAs.	`smf(5)`, `in.iked(1M)`
`/etc/inet/ipsecinit.conf` file	IPsec policy file. The SMF `policy` service uses this file to configure IPsec policy at system boot.	`ipsecconf(1M)`

TABLE 18–3 List of Selected IPsec Utilities and Files *(Continued)*

IPsec Utility, File, or Service	Description	Man Page
`ipsecconf` command	IPsec policy command. Useful for viewing and modifying the current IPsec policy, and for testing. Is used by the SMF `policy` service to configure IPsec policy at system boot.	`ipsecconf(1M)`
`PF_KEY` socket interface	Interface for the security associations database (SADB). Handles manual key management and automatic key management.	`pf_key(7P)`
`ipseckey` command	IPsec SAs keying command. `ipseckey` is a command-line front end to the `PF_KEY` interface. `ipseckey` can create, destroy, or modify SAs.	`ipseckey(1M)`
`/etc/inet/secret/ipseckeys` file	Keys for IPsec SAs. Is used by the SMF `manual-key` service to configure SAs manually at system boot.	
`ipsecalgs` command	IPsec algorithms command. Useful for viewing and modifying the list of IPsec algorithms and their properties. Is used by the SMF `ipsecalgs` service to synchronize known IPsec algorithms with the kernel at system boot.	`ipsecalgs(1M)`
`/etc/inet/ipsecalgs` file	Contains the configured IPsec protocols and algorithm definitions. This file is managed by the `ipsecalgs` command and must never be edited manually.	
`/etc/inet/ike/config` file	IKE configuration and policy file. By default, this file does not exist. The management is based on rules and global parameters in the `/etc/inet/ike/config` file. See "IKE Utilities and Files" on page 455. If this file exists, the `svc:/network/ipsec/ike` service starts the IKE daemon, `in.iked`, to provide automatic key management.	`ike.config(4)`

◆ ◆ ◆ CHAPTER 19

Configuring IPsec (Tasks)

This chapter provides procedures for implementing IPsec on your network. The procedures are described in the following task maps:

- "Protecting Traffic With IPsec (Task Map)" on page 409
- "Protecting a VPN With IPsec (Task Map) " on page 427

For overview information about IPsec, see Chapter 18, "IP Security Architecture (Overview)." For reference information about IPsec, see Chapter 20, "IP Security Architecture (Reference)."

Protecting Traffic With IPsec (Task Map)

The following task map points to procedures that set up IPsec between one or more systems. The `ipsecconf`(1M), `ipseckey`(1M), `ipadm`(1M), and `ifconfig`(1M) man pages also describe useful procedures in their respective Examples sections.

Task	Description	For Instructions
Secure traffic between two systems.	Protects packets from one system to another system.	"How to Secure Traffic Between Two Systems With IPsec" on page 411
Secure a web server by using IPsec policy.	Requires non-web traffic to use IPsec. Web clients are identified by particular ports, which bypass IPsec checks.	"How to Use IPsec to Protect a Web Server From Nonweb Traffic" on page 414
Display IPsec policies.	Displays the IPsec policies that are currently being enforced, in the order in which the policies are enforced.	"How to Display IPsec Policies" on page 415

Task	Description	For Instructions
Generate random numbers.	Generates random numbers for keying material for manually created security associations.	"How to Generate Random Numbers on a Solaris System" on page 416 "How to Generate a Symmetric Key by Using the pktool Command" in *System Administration Guide: Security Services*
Create or replace security associations manually.	Provides the raw data for security associations: ■ IPsec algorithm name and keying material ■ Key for the security parameter index ■ IP source and destination addresses	"How to Manually Create IPsec Security Associations" on page 417
Check that IPsec is protecting the packets.	Examines `snoop` output for specific headers that indicate how the IP datagrams are protected.	"How to Verify That Packets Are Protected With IPsec" on page 420
(Optional) Create a Network Security role.	Creates a role that can set up a secure network, but has fewer powers than superuser.	"How to Configure a Role for Network Security" on page 421
Manage IPsec and keying material as a set of SMF services.	Describes when and how to use the commands that enable, disable, refresh, and restart services. Also describes the commands that change the property values of services.	"How to Manage IKE and IPsec Services" on page 423
Set up a secure virtual private network (VPN).	Sets up IPsec between two systems that are separated by the Internet.	"Protecting a VPN With IPsec (Task Map) " on page 427

Protecting Traffic With IPsec

This section provides procedures that enable you to secure traffic between two systems and to secure a web server. To protect a VPN, see "Protecting a VPN With IPsec (Task Map) " on page 427. Additional procedures provide keying material and security associations, and verify that IPsec is working as configured.

The following information applies to all IPsec configuration tasks:

- **IPsec and zones –** To manage IPsec policy and keys for a shared-IP non-global zone, create the IPsec policy file in the global zone, and run the IPsec configuration commands from the global zone. Use the source address that corresponds to the non-global zone that is being configured. You can also configure IPsec policy and keys in the global zone for the global zone. For an exclusive-IP zone, you configure IPsec policy in the non-global zone. In this Solaris release, you can use IKE to manage keys in a non-global zone.
- **IPsec and RBAC –** To use roles to administer IPsec, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*. For an example, see "How to Configure a Role for Network Security" on page 421.

- **IPsec and SCTP –** IPsec can be used to protect Streams Control Transmission Protocol (SCTP) associations, but caution must be used. For more information, see "IPsec and SCTP" on page 405.
- **IPsec and Trusted Extensions labels –** On Trusted Extensions systems, labels can be added to IPsec packets. For more information, see "Administration of Labeled IPsec" in *Oracle Solaris Trusted Extensions Configuration and Administration*.

▼ How to Secure Traffic Between Two Systems With IPsec

This procedure assumes the following setup:

- The two systems are named `enigma` and `partym`.
- Each system has two addresses, an IPv4 address and an IPv6 address.
- Each system requires ESP encryption with the AES algorithm, which requires a key of 128 bits, and ESP authentication with the SHA1 message digest, which requires a 160–bit key.
- Each system uses shared security associations.

  With shared SAs, only one pair of SAs is needed to protect the two systems.

Note – To use IPsec with labels on a Trusted Extensions system, see the extension of this procedure in "How to Apply IPsec Protections in a Multilevel Trusted Extensions Network" in *Oracle Solaris Trusted Extensions Configuration and Administration*.

Before You Begin You must be in the global zone to configure IPsec policy for the system or for a shared-IP zone. For an exclusive-IP zone, you configure IPsec policy in the non-global zone.

1 Become an administrator.

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session. Use the `ssh` command for a secure remote login. For an example, see Example 19–1.

---

2 On each system, add host entries to the `/etc/inet/hosts` file.

This step enables the service management facility (SMF) to use the system names without depending on nonexistent naming services. For more information, see the `smf(5)` man page.

a. On a system that is named `partym`, type the following in the `hosts` file:

```
# Secure communication with enigma
192.168.116.16 enigma
2001::aaaa:6666:6666 enigma
```

b. On a system that is named `enigma`, type the following in the `hosts` file:

```
# Secure communication with partym
192.168.13.213 partym
2001::eeee:3333:3333 partym
```

3 On each system, create the IPsec policy file.

The file name is `/etc/inet/ipsecinit.conf`. For an example, see the `/etc/inet/ipsecinit.sample` file.

4 Add an IPsec policy entry to the `ipsecinit.conf` file.

a. On the `enigma` system, add the following policy:

```
{laddr enigma raddr partym} ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

b. On the `partym` system, add the identical policy:

```
{laddr partym raddr enigma} ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

For the syntax of IPsec policy entries, see the `ipsecconf(1M)` man page.

5 On each system, add a pair of IPsec SAs between the two systems.

You can configure Internet Key Exchange (IKE) to create the SAs automatically. You can also add the SAs manually.

Note – You should use IKE unless you have good reason to generate and maintain your keys manually. IKE key management is more secure than manual key management.

- Configure IKE by following one of the configuration procedures in "Configuring IKE (Task Map)" on page 459. For the syntax of the IKE configuration file, see the `ike.config(4)` man page.
- To add the SAs manually, see "How to Manually Create IPsec Security Associations" on page 417.

6 Verify the syntax of the IPsec policy file.

```
# ipsecconf -c -f /etc/inet/ipsecinit.conf
```

Fix any errors, verify the syntax of the file, and continue.

7 **Refresh the IPsec policy.**

```
# svcadm refresh svc:/network/ipsec/policy:default
```

IPsec policy is enabled by default, so you *refresh* it. If you have disabled IPsec policy, enable it.

```
# svcadm enable svc:/network/ipsec/policy:default
```

8 **Activate the keys for IPsec.**

- **If you configured IKE in Step 5, do one of the following:**
  - **If the `ike` service is not enabled, enable it.**

    ```
    # svcadm enable svc:/network/ipsec/ike:default
    ```

  - **If the `ike` service is enabled, restart it.**

    ```
    # svcadm restart svc:/network/ipsec/ike:default
    ```

- **If you manually configured keys in Step 5, do one of the following:**
  - **If the `manual-key` service is not enabled, enable it.**

    ```
    # svcadm enable svc:/network/ipsec/manual-key:default
    ```

  - **If the `manual-key` service is enabled, refresh it.**

    ```
    # svcadm refresh svc:/network/ipsec/manual-key:default
    ```

9 **Verify that packets are being protected.**

For the procedure, see "How to Verify That Packets Are Protected With IPsec" on page 420.

Example 19–1 Adding IPsec Policy When Using an `ssh` Connection

In this example, the administrator as superuser configures IPsec policy and keys on two systems by using the `ssh` command to reach the second system. For more information, see the `ssh(1)` man page.

- First, the administrator configures the first system by performing Step 2 through Step 5 of the preceding procedure.
- Then, in a different terminal window, the administrator uses the `ssh` command to log in to the second system.

  ```
  local-system # ssh other-system
  other-system #
  ```

- In the terminal window of the `ssh` session, the administrator configures the IPsec policy and keys of the second system by completing Step 2 through Step 8.
- Then, the administrator ends the `ssh` session.

```
other-system # exit
local-system #
```

- Finally, the administrator enables IPsec policy on the first system by completing Step 6 and Step 8.

The next time the two systems communicate, including by using an `ssh` connection, the communication is protected by IPsec.

How to Use IPsec to Protect a Web Server From Nonweb Traffic

A secure web server allows web clients to talk to the web service. On a secure web server, traffic that is not web traffic *must* pass security checks. The following procedure includes bypasses for web traffic. In addition, this web server can make unsecured DNS client requests. All other traffic requires ESP with AES and SHA-1 algorithms.

Before You Begin You must be in the global zone to configure IPsec policy. For an exclusive-IP zone, you configure IPsec policy in the non-global zone. You have completed "How to Secure Traffic Between Two Systems With IPsec" on page 411 so that the following conditions are in effect:

- Communication between the two systems is protected by IPsec.
- Keying material is being generated, either manually or by IKE.
- You have verified that packets are being protected.

1 Become an administrator.

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session. Use the `ssh` command for a secure remote login. For an example, see Example 19–1.

---

2 Determine which services need to bypass security policy checks.

For a web server, these services include TCP ports 80 (HTTP) and 443 (Secure HTTP). If the web server provides DNS name lookups, the server might also need to include port 53 for both TCP and UDP.

3 Add the web server policy to the IPsec policy file.

Add the following lines to the `/etc/inet/ipsecinit.conf` file:

```
# Web traffic that web server should bypass.
{lport  80 ulp tcp dir both} bypass {}
{lport 443 ulp tcp dir both} bypass {}
```

```
# Outbound DNS lookups should also be bypassed.
{rport 53 dir both} bypass {}

# Require all other traffic to use ESP with AES and SHA-1.
# Use a unique SA for outbound traffic from the port
{} ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

This configuration allows only secure traffic to access the system, with the bypass exceptions that are described in Step 3.

4 **Verify the syntax of the IPsec policy file.**

```
# ipsecconf -c -f /etc/inet/ipsecinit.conf
```

5 **Refresh the IPsec policy.**

```
# svcadm refresh svc:/network/ipsec/policy:default
```

6 **Refresh the keys for IPsec.**

- **If you configured IKE in Step 5 of "How to Secure Traffic Between Two Systems With IPsec" on page 411, restart the `ike` service.**

  ```
  # svcadm restart svc:/network/ipsec/ike
  ```

- **If you manually configured keys in Step 5 of "How to Secure Traffic Between Two Systems With IPsec" on page 411, refresh the `manual-key` service.**

  ```
  # svcadm refresh svc:/network/ipsec/manual-key:default
  ```

Your setup is complete. Optionally, you can perform Step 7.

7 **(Optional) Enable a remote system to communicate with the web server for nonweb traffic.**

Type the following policy in a remote system's `ipsecinit.conf` file:

```
# Communicate with web server about nonweb stuff
#
{laddr webserver} ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

A remote system can communicate securely with the web server for nonweb traffic only when the systems' IPsec policies match.

▼ How to Display IPsec Policies

You can see the policies that are configured in the system when you issue the `ipsecconf` command without any arguments.

Before You Begin You must run the `ipsecconf` command in the global zone. For an exclusive-IP zone, you run the `ipsecconf` command in the non-global zone.

1 **Assume a role that includes the Network IPsec Management profile, or become superuser.**

To create a role that includes a network security profile and assign that role to a user, see "How to Configure a Role for Network Security" on page 421.

2 **Display IPsec policies.**

a. **Display the global IPsec policy entries in the order that the entries were added.**

```
$ ipsecconf
```

The command displays each entry with an *index* followed by a number.

b. **Display the IPsec policy entries in the order in which a match occurs.**

```
$ ipsecconf -l
```

c. **Display the IPsec policy entries, including per-tunnel entries, in the order in which a match occurs.**

```
$ ipsecconf -L
```

▼ How to Generate Random Numbers on a Solaris System

If you are specifying keys manually, the keying material must be random. The format for keying material for a Solaris system is hexadecimal. Other operating systems can require ASCII keying material. To generate keying material for a Solaris system that is communicating with an operating system that requires ASCII, see Example 22–1.

On a Solaris system, you have three options for generating random keys.

- Use a random number generator. If your site has a generator, do not use this procedure.
- Use the `od` command that is described in this procedure. The `od` command uses the `/dev/random` Solaris device as input. For more information, see the `od(1)` man page.
- Use the `pktool` command. The syntax of this command is simpler than the syntax of the `od` command. For details, see "How to Generate a Symmetric Key by Using the pktool Command" in *System Administration Guide: Security Services*.

1 **Generate random numbers in hexadecimal format.**

```
% od -x|-X -A n file | head -n
```

`-x` Displays the octal dump in hexadecimal format. Hexadecimal format is useful for keying material. The hexadecimal is printed in 4-character chunks.

`-X` Displays the octal dump in hexadecimal format. The hexadecimal is printed in 8-character chunks.

`-A n` Removes the input offset base from the display.

file Serves as a source for random numbers.

`head -`*n* Restricts the display to the first *n* lines of output.

2 Combine the output to create a key of the appropriate length.

Remove the spaces between the numbers on one line to create a 32-character key. A 32-character key is 128 bits. For a security parameter index (SPI), you should use an 8-character key. The key should use the `0x` prefix.

Example 19–2 Generating Key Material for IPsec

The following example displays two lines of keys in groups of eight hexadecimal characters each.

```
% od -X -A n /dev/random | head -2
         d54d1536 4a3e0352 0faf93bd 24fd6cad
         8ecc2670 f3447465 20db0b0c c83f5a4b
```

By combining the four numbers on the first line, you can create a 32-character key. An 8-character number that is preceded by `0x` provides a suitable SPI value, for example, `0xf3447465`.

The following example displays two lines of keys in groups of four hexadecimal characters each.

```
% od -x -A n /dev/random | head -2
         34ce 56b2 8b1b 3677 9231 42e9 80b0 c673
         2f74 2817 8026 df68 12f4 905a db3d ef27
```

By combining the eight numbers on the first line, you can create a 32-character key.

▼ How to Manually Create IPsec Security Associations

The following procedure provides the keying material for Step 5 in "How to Secure Traffic Between Two Systems With IPsec" on page 411. You are generating keys for two systems, `partym` and `enigma`. You generate the keys on one system, and then use the keys from the first system on both systems.

Before You Begin You must be in the global zone to manually manage keying material for a non-global zone.

1 Generate the keying material for the SAs.

a. Determine the keys that you require.

You need three hexadecimal random numbers for outbound traffic and three hexadecimal random numbers for inbound traffic. Therefore, one system needs to generate the following numbers:

- Two hexadecimal random numbers as the value for the `spi` keyword. One number is for outbound traffic. One number is for inbound traffic. Each number can be up to eight characters long.
- Two hexadecimal random numbers for the MD5 algorithm for AH. Each number must be 32 characters long. One number is for `dst enigma`. One number is for `dst partym`.
- Two hexadecimal random numbers for the 3DES algorithm for ESP. For a 192-bit key, each number must be 48 characters long. One number is for `dst enigma`. One number is for `dst partym`.

b. Decide on your method of key generation.

- If you have a random number generator at your site, use the generator.
- To use the `pktool` command, see "How to Generate a Symmetric Key by Using the pktool Command" in *System Administration Guide: Security Services*.
- To use the `od` command, see "How to Generate Random Numbers on a Solaris System" on page 416.

2 On each system, perform the following steps:

a. Become an administrator.

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session. Use the `ssh` command for a secure remote login. For an example, see Example 19–1.

---

b. Add the keys to the manual keys file for IPsec.

i. Edit the `/etc/inet/secret/ipseckeys` file on the `enigma` system to appear similar to the following:

```
# ipseckeys - This file takes the file format documented in
#   ipseckey(1m).
#   Note that naming services might not be available when this file
#   loads, just like ipsecinit.conf.
#
#   Backslashes indicate command continuation.
#
# for outbound packets on enigma
```

```
add esp spi 0x8bcd1407 \
   src 192.168.116.16 dst 192.168.13.213  \
   encr_alg 3des \
   auth_alg md5  \
   encrkey  d41fb74470271826a8e7a80d343cc5aae9e2a7f05f13730d \
   authkey  e896f8df7f78d6cab36c94ccf293f031
#
# for inbound packets
add esp spi 0x122a43e4 \
   src 192.168.13.213 dst 192.168.116.16 \
   encr_alg 3des \
   auth_alg md5  \
   encrkey dd325c5c137fb4739a55c9b3a1747baa06359826a5e4358e \
   authkey ad9ced7ad5f255c9a8605fba5eb4d2fd
```

ii. Protect the manual keys file for IPsec with read-only permissions.

```
# chmod 400 /etc/inet/secret/ipseckeys
```

iii. Verify the syntax of the manual keys file.

```
# ipseckey -c -f /etc/inet/secret/ipseckeys
```

Note – The keying material on the two systems *must* be identical.

Example 19–3 Manually Creating Temporary IPsec Security Associations

In this example, the administrator tests various keys. Later, the administrator will type the permanent keys in the `ipseckeys` file.

During testing, the administrator creates keys by using the `ipseckey` command in interactive mode. When the `ipseckey` command is typed, the > prompt indicates interactive mode.

```
# ipseckey
>
```

To replace existing SAs, the administrator flushes the current SAs.

```
> flush
>
```

To create SAs for outbound packets, the administrator types the following command:

```
> add esp spi 0x8bcd1407 \
src 192.168.116.16 dst 192.168.13.213 \
encr_alg 3des \
auth_alg md5 \
encrkey d41fb74470271826a8e7a80d343cc5aae9e2a7f05f13730d \
authkey e896f8df7f78d6cab36c94ccf293f031
>
```

The administrator types the following command for inbound packets:

```
> add esp spi 0x122a43e4 \
src 192.168.13.213 dst 192.168.116.16 \
encr_alg 3des \
auth_alg md5 \
encrkey dd325c5c137fb4739a55c9b3a1747baa06359826a5e4358e \
authkey ad9ced7ad5f255c9a8605fba5eb4d2fd
>
```

To exit `ipseckey` interactive mode, the administrator types the `quit` command.

```
> quit
#
```

To change keys on the communicating system, the administrator follows the same steps. On that system, the first set of keys protects inbound packets and the second set of keys protects outbound packets.

▼ How to Verify That Packets Are Protected With IPsec

To verify that packets are protected, test the connection with the `snoop` command. The following prefixes can appear in the `snoop` output:

- `AH:` Prefix indicates that AH is protecting the headers. You see `AH:` if you used `auth_alg` to protect the traffic.
- `ESP:` Prefix indicates that encrypted data is being sent. You see `ESP:` if you used `encr_auth_alg` or `encr_alg` to protect the traffic.

Before You Begin You must be superuser or have assumed an equivalent role to create the `snoop` output. You must have access to both systems to test the connection.

1 On one system, such as `partym`, become superuser.

```
% su -
Password:       Type root password
#
```

2 From the `partym` system, prepare to snoop packets from a remote system.

In a terminal window on `partym`, snoop the packets from the `enigma` system.

```
# snoop -v enigma
Using device /dev/hme (promiscuous mode)
```

3 Send a packet from the remote system.

In another terminal window, remotely log in to the `enigma` system. Provide your password. Then, become superuser and send a packet from the `enigma` system to the `partym` system. The packet should be captured by the `snoop -v enigma` command.

```
% ssh enigma
Password:       Type your password
% su -
Password:       Type root password
# ping partym
```

4 Examine the `snoop` output.

On the `partym` system, you should see output that includes `AH` and `ESP` information after the initial `IP` header information. `AH` and `ESP` information that resembles the following shows that packets are being protected:

```
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 51 (AH)
IP:   Header checksum = 4e0e
IP:   Source address = 192.168.116.16, enigma
IP:   Destination address = 192.168.13.213, partym
IP:   No options
IP:
AH:  ----- Authentication Header -----
AH:
AH:  Next header = 50 (ESP)
AH:  AH length = 4 (24 bytes)
AH:  <Reserved field = 0x0>
AH:  SPI = 0xb3a8d714
AH:  Replay = 52
AH:  ICV = c653901433ef5a7d77c76eaa
AH:
ESP:  ----- Encapsulating Security Payload -----
ESP:
ESP:  SPI = 0xd4f40a61
ESP:  Replay = 52
ESP:     ....ENCRYPTED DATA....

ETHER:  ----- Ether Header -----
...
```

▼ How to Configure a Role for Network Security

If you are using role-based access control (RBAC) to administer your systems, you use this procedure to provide a network management role or network security role.

1 Find the Network rights profiles in the local `prof_attr` database.

```
% cd /etc/security
% grep Network prof_attr
Network IPsec Management:::Manage IPsec and IKE...
Network Link Security:::Manage network link security...
```

```
Network Management:::Manage the host and network configuration...
Network Security:::Manage network and host security...
Network Wifi Management:::Manage wifi network configuration...
Network Wifi Security:::Manage wifi network security...
```

The Network Management profile is a supplementary profile in the System Administrator profile. If you have included the System Administrator rights profile in a role, then that role can execute the commands in the Network Management profile.

2 Determine which commands are in the Network Management rights profile.

```
% grep "Network Management" /etc/security/exec_attr
Network Management:solaris:cmd:::/usr/sbin/ifconfig:privs=sys_net_config
...
Network Management:suser:cmd:::/usr/sbin/snoop:uid=0
```

The `solaris` policy commands run with privilege (`privs=sys_net_config`). The `suser` policy commands run as superuser (`uid=0`).

3 Decide the scope of the network security roles at your site.

Use the definitions of the rights profiles in Step 1 to guide your decision.

- **To create a role that handles all network security, use the Network Security rights profile.**
- **To create a role that handles IPsec and IKE only, use the Network IPsec Management rights profile.**

4 Create a network security role that includes the Network Management rights profile.

A role with the Network Security or the Network IPsec Management rights profile, in addition to the Network Management profile, can execute the `ipadm`, `ifconfig`, `snoop`, `ipsecconf`, and `ipseckey` commands, among others, with appropriate privilege.

To create the role, assign the role to a user, and register the changes with the name service, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

Example 19–4 Dividing Network Security Responsibilities Between Roles

In this example, the administrator divides network security responsibilities between two roles. One role administers wifi and link security and another role administers IPsec and IKE. Each role is assigned to three people, one person per shift.

The roles are created by the administrator as follows:

- The administrator names the first role LinkWifi.
  - The administrator assigns the Network Wifi, Network Link Security, and Network Management rights profiles to the role.
  - Then, the administrator assigns the LinkWifi role to the appropriate users.

- The administrator names the second role IPsec Administrator.
  - The administrator assigns the Network IPsec Management and the Network Management rights profiles to the role.
  - Then, the administrator assigns the IPsec Administrator role to the appropriate users.

How to Manage IKE and IPsec Services

The following steps provide the most likely uses of the SMF services for IPsec, IKE, and manual key management. By default, the `policy` and `ipsecalgs` services are enabled. Also by default, the `ike` and `manual-key` services are disabled.

1 **To manage IPsec policy, do one of the following:**

- **After adding new policies to the `ipsecinit.conf` file, refresh the `policy` service.**

```
# svcadm refresh svc:/network/ipsec/policy
```

- **After changing the value of a service property, view the property value, then refresh and restart the `policy` service.**

```
# svccfg -s policy setprop config/config_file=/etc/inet/MyIpsecinit.conf
# svcprop -p config/config_file policy
/etc/inet/MyIpsecinit.conf
# svcadm refresh svc:/network/ipsec/policy
# svcadm restart svc:/network/ipsec/policy
```

2 **To automatically manage keys, do one of the following:**

- **After adding entries to the `/etc/inet/ike/config` file, enable the `ike` service.**

```
# svcadm enable svc:/network/ipsec/ike
```

- **After changing entries in the `/etc/inet/ike/config` file, refresh the `ike` service.**

```
# svcadm refresh svc:/network/ipsec/ike
```

- **After changing the value of a service property, view the property value, then refresh and restart the service.**

```
# svccfg -s ike setprop config/admin_privilege=modkeys
# svcprop -p config/admin_privilege ike
modkeys
# svcadm refresh svc:/network/ipsec/ike
# svcadm restart svc:/network/ipsec/ike
```

- **To stop the `ike` service, disable it.**

```
# svcadm disable svc:/network/ipsec/ike
```

3 **To manually manage keys, do one of the following:**

- **After adding entries to the `/etc/inet/secret/ipseckeys` file, enable the `manual-key` service.**

  ```
  # svcadm enable svc:/network/ipsec/manual-key
  ```

- **After changing the `ipseckeys` file, refresh the service.**

  ```
  # svcadm refresh manual-key
  ```

- **After changing the value of a service property, view the property value, then refresh and restart the service.**

  ```
  # svccfg -s manual-key setprop config/config_file=/etc/inet/secret/MyIpseckeyfile
  # svcprop -p config/config_file manual-key
  /etc/inet/secret/MyIpseckeyfile
  # svcadm refresh svc:/network/ipsec/manual-key
  # svcadm restart svc:/network/ipsec/manual-key
  ```

- **To prevent manual key management, disable the `manual-key` service.**

  ```
  # svcadm disable svc:/network/ipsec/manual-key
  ```

4 **If you modify the IPsec protocols and algorithms table, refresh the `ipsecalgs` service.**

```
# svcadm refresh svc:/network/ipsec/ipsecalgs
```

Troubleshooting Use the `svcs` *service* command to find the status of a service. If the service is in `maintenance` mode, follow the debugging suggestions in the output of the `svcs -x` *service* command.

Protecting a VPN With IPsec

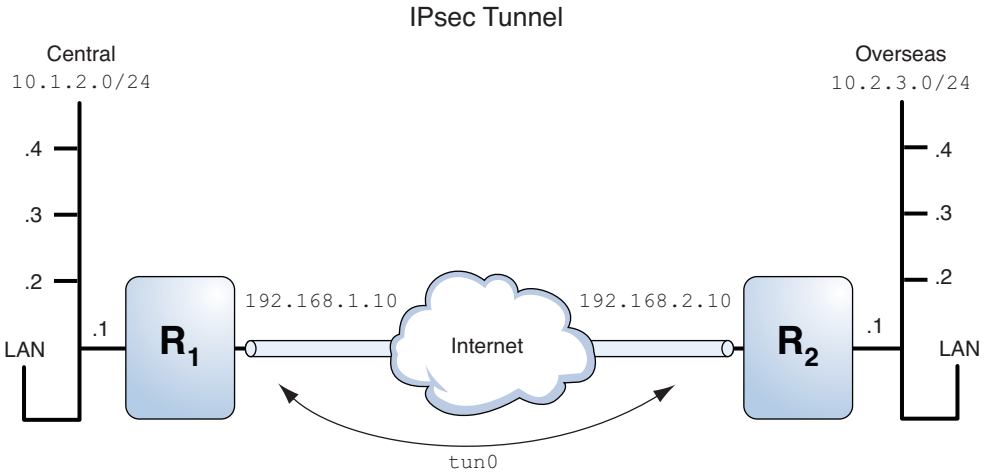
IPsec tunnels can protect a VPN. Tunnels have two modes. *Tunnel mode* is interoperable with the implementation of IPsec by other vendors. *Transport mode* is interoperable with earlier versions of the Solaris OS. For a discussion of tunnel modes, see "Transport and Tunnel Modes in IPsec" on page 401.

Tunnels in tunnel mode offer more fine-grained control of the traffic. In tunnel mode, for an inner IP address, you can specify the particular protection you want, down to a single port.

- For examples of IPsec policies for tunnels in tunnel mode, see "Examples of Protecting a VPN With IPsec by Using Tunnels in Tunnel Mode" on page 425.
- For the procedures that protect VPNs, see "Protecting a VPN With IPsec (Task Map) " on page 427.

Examples of Protecting a VPN With IPsec by Using Tunnels in Tunnel Mode

FIGURE 19–1 IPsec Tunnel Diagram



The following examples assume that the tunnel is configured for all subnets of the LANs:

```
## Tunnel configuration ##
# Tunnel name is tun0
# Intranet point for the source is 10.1.2.1
# Intranet point for the destination is 10.2.3.1
# Tunnel source is 192.168.1.10
# Tunnel destination is 192.168.2.10

# Tunnel name address object is tun0/to-central
# Tunnel name address object is tun0/to-overseas
```

EXAMPLE 19–5 Creating a Tunnel That All Subnets Can Use

In this example, all traffic from the local LANs of the Central LAN in Figure 19–1 can be tunneled through Router 1 to Router 2, and then delivered to all local LANs of the Overseas LAN. The traffic is encrypted with AES.

```
## IPsec policy ##
{tunnel tun0 negotiate tunnel}
 ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

EXAMPLE 19–6 Creating a Tunnel That Connects Two Subnets Only

In this example, only traffic between subnet `10.1.2.0/24` of the Central LAN and subnet `10.2.3.0/24` of the Overseas LAN is tunneled and encrypted. In the absence of other IPsec policies for Central, if the Central LAN attempts to route any traffic for other LANs over this tunnel, the traffic is dropped at Router 1.

```
## IPsec policy ##
{tunnel tun0 negotiate tunnel laddr 10.1.2.0/24 raddr 10.2.3.0/24}
 ipsec {encr_algs aes encr_auth_algs md5 sha1 shared}
```

EXAMPLE 19–7 Creating a Tunnel for Email Traffic Only Between Two Subnets

In this example, a tunnel is created for email traffic only. The traffic is delivered from subnet `10.1.2.0/24` of the Central LAN to the email server on the `10.2.3.0/24` subnet of the Overseas LAN. The email is encrypted with Blowfish. The policies apply to the remote and local email ports. The `rport` policy protects email that Central sends to the remote email port of Overseas. The `lport` policy protects email that Central receives from Overseas on local port 25.

```
## IPsec policy for email from Central to Overseas ##
{tunnel tun0 negotiate tunnel ulp tcp rport 25
 laddr 10.1.2.0/24 raddr 10.2.3.0/24}
 ipsec {encr_algs blowfish encr_auth_algs sha1 sa shared}

## IPsec policy for email from Overseas to Central ##
{tunnel tun0 negotiate tunnel ulp tcp lport 25
 laddr 10.1.2.0/24 raddr 10.2.3.0/24}
 ipsec {encr_algs blowfish encr_auth_algs sha1 sa shared}
```

EXAMPLE 19–8 Creating a Tunnel for FTP Traffic for All Subnets

In this example, IPsec policy protects the FTP ports in Figure 19–1 with AES for all subnets of the Central LAN to all subnets of the Overseas LAN. This configuration works for the active mode of FTP.

```
## IPsec policy for outbound FTP from Central to Overseas ##
{tunnel tun0 negotiate tunnel ulp tcp rport 21}
  ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
{tunnel tun0 negotiate tunnel ulp tcp lport 20}
  ipsec {encr_algs aes encr_auth_algs sha1 sa shared}

## IPsec policy for inbound FTP from Central to Overseas ##
{tunnel tun0 negotiate tunnel ulp tcp lport 21}
  ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
{tunnel tun0 negotiate tunnel ulp tcp rport 20}
  ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

Protecting a VPN With IPsec (Task Map)

The following task map points to procedures that configure IPsec to protect traffic across the Internet. These procedures set up a secure virtual private network (VPN) between two systems that are separated by the Internet. One common use of this technology is to protect traffic between home workers and their corporate office.

Task	Description	For Instructions
Protect tunnel traffic in tunnel mode.	Protects traffic in tunnel mode between two Oracle Solaris systems. Also, protects traffic in tunnel mode between an Oracle Solaris Express system, and a system that is running on another platform.	"How to Protect a VPN With an IPsec Tunnel in Tunnel Mode" on page 429
Protect tunnel traffic in transport mode.	Protects traffic in transport mode between two Solaris systems. Also, protects traffic in transport mode between a system that is running an earlier version of the Solaris OS and an Oracle Solaris system.	"How to Protect a VPN With an IPsec Tunnel in Transport Mode" on page 438
	Protects traffic by using an older, deprecated syntax. This method is useful when you are communicating with a system that is running an earlier version of the Solaris OS. This method simplifies comparing the configuration files on the two systems.	Example 19–10
Prevent IP spoofing.	Creates an SMF service to prevent the system from forwarding packets across a VPN without decrypting the packets.	"How to Prevent IP Spoofing" on page 442

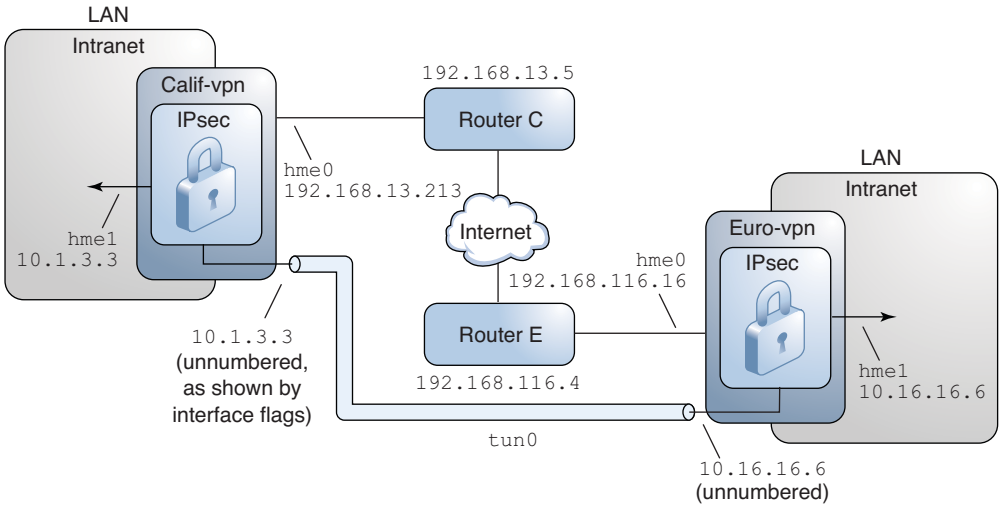
Description of the Network Topology for the IPsec Tasks to Protect a VPN

The procedures that follow this section assume the following setup. For a depiction of the network, see Figure 19–2.

- Each system is using an IPv4 address space. The procedures include examples of IPv6 syntax.
- Each system has two interfaces. The `hme0` interface connects to the Internet. In this example, Internet IP addresses begin with `192.168`. The `hme1` interface connects to the company's LAN, its intranet. In this example, intranet IP addresses begin with the number `10`.
- Each system requires ESP authentication with the SHA–1 algorithm. The SHA–1 algorithm requires a 160-bit key.

- Each system requires ESP encryption with the AES algorithm. The AES algorithm uses a 128-bit or 256–bit key.
- Each system can connect to a router that has direct access to the Internet.
- Each system uses shared security associations.

FIGURE 19–2 Sample VPN Between Offices Separated by the Internet



As the preceding illustration shows, the procedures for the IPv4 network use the following configuration parameters.

Parameter	Europe	California
System name	`enigma`	`partym`
System intranet interface	`hme1`	`hme1`
System intranet address, also the *-point* address in Step 6	`10.16.16.6`	`10.1.3.3`
System intranet address object	`hme1/LAN`	`hme1/LAN`
System Internet interface	`hme0`	`hme0`
System Internet address, also the *tsrc* address in Step 6	`192.168.116.16`	`192.168.13.213`
System Internet address object	`hme0/WAN`	`hme0/WAN`
Name of Internet router	`router-E`	`router-C`
Address of Internet router	`192.168.116.4`	`192.168.13.5`

Parameter	Europe	California
Tunnel name	`tun0`	`tun0`
Tunnel name address object	`tun0/to-calif`	`tun0/to-europe`

The following IPv6 addresses are used in the procedures. The tunnel names, tunnel address objects, and Internet and intranet address objects are the same. For information about tunnel names, see "Tunnel Configuration and Administration With the `dladm` Command" on page 159. For information about address objects, see "How to Configure an IP Interface" in *System Administration Guide: Network Interfaces and Network Virtualization*. Also see the `ipadm`(1M) man page.

Parameter	Europe	California
System intranet address	`6000:6666::aaaa:1116`	`6000:3333::eeee:1113`
System Internet address	`2001::aaaa:6666:6666`	`2001::eeee:3333:3333`
Address of Internet router	`2001::aaaa:0:4`	`2001::eeee:0:1`

How to Protect a VPN With an IPsec Tunnel in Tunnel Mode

In tunnel mode, the inner IP packet determines the IPsec policy that protects its contents.

This procedure extends the procedure "How to Secure Traffic Between Two Systems With IPsec" on page 411. The setup is described in "Description of the Network Topology for the IPsec Tasks to Protect a VPN" on page 427.

Note – Perform the steps in this procedure on both systems.

In addition to connecting two systems, you are connecting two intranets that connect to these two systems. The systems in this procedure function as gateways.

Note – To use IPsec in tunnel mode with labels on a Trusted Extensions system, see the extension of this procedure in "How to Configure a Tunnel Across an Untrusted Network" in *Oracle Solaris Trusted Extensions Configuration and Administration*.

Before You Begin You must be in the global zone to configure IPsec policy for the system or for a shared-IP zone. For an exclusive-IP zone, you configure IPsec policy in the non-global zone.

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session. Use the `ssh` command for a secure remote login. For an example, see Example 19–1.

---

2 **Control the flow of packets before configuring IPsec.**

a. **Ensure that IP forwarding and IP dynamic routing are disabled.**

```
# routeadm
```

On an IPv4 system, the output appears similar to the following:

```
# routeadm
Configuration       Current         Current
       Option       Configuration  System State
---------------------------------------------------
IPv4 forwarding     disabled           disabled
   IPv4 routing     default (enabled)   enabled
...
```

On an IPv6 system, the output shows IPv6 routing and configuration settings.

b. **If IP forwarding and IP dynamic routing are enabled, disable them.**

- **For an IPv4 network, type the following commands:**

```
# routeadm -d ipv4-routing
# ipadm set-prop -p forwarding=off ipv4
# routeadm -u
```

- **For an IPv6 network, type the following commands:**

```
# ipadm set-prop -p forwarding=off ipv6
# routeadm -d ipv6-routing
# routeadm -u
```

Turning off IP forwarding prevents packets from being forwarded from one network to another network through this system. For a description of the `routeadm` command, see the `routeadm`(1M) man page.

c. **Turn on IP strict destination multihoming.**

- **For an IPv4 network, type the following command:**

```
# ndd -set /dev/ip ip_strict_dst_multihoming 1
```

- **For an IPv6 network, type the following command:**

  ```
  # ndd -set /dev/ip ip6_strict_dst_multihoming 1
  ```

  Turning on IP strict destination multihoming ensures that packets for one of the system's destination addresses arrive at the correct destination address.

  When strict destination multihoming is enabled, packets that arrive on a particular interface must be addressed to one of the local IP addresses of that interface. All other packets, even packets that are addressed to other local addresses of the system, are dropped.

Caution – The multihoming value reverts to the default when the system is booted. To make the changed value persistent, see "How to Prevent IP Spoofing" on page 442.

d. Verify that most network services are disabled.

Verify that loopback mounts and the `ssh` service are running.

```
# svcs | grep network
online         Aug_02   svc:/network/loopback:default
...
online         Aug_09   svc:/network/ssh:default
```

3 Add a pair of SAs between the two systems.

Choose one of the following options:

- Configure IKE to manage the keys for the SAs. Use one of the procedures in "Configuring IKE (Task Map)" on page 459 to configure IKE for the VPN.
- If you have an overriding reason to manually manage the keys, see "How to Manually Create IPsec Security Associations" on page 417.

4 Add IPsec policy.

Edit the `/etc/inet/ipsecinit.conf` file to add the IPsec policy for the VPN. To strengthen the policy, see Example 19–11. For additional examples, see "Examples of Protecting a VPN With IPsec by Using Tunnels in Tunnel Mode" on page 425.

In this policy, IPsec protection is not required between systems on the local LAN and the internal IP address of the gateway, so a `bypass` statement is added.

a. On the `enigma` system, type the following entry into the `ipsecinit.conf` file:

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 10.16.16.6 dir both} bypass {}

# WAN traffic uses ESP with AES and SHA-1.
{tunnel tun0 negotiate tunnel}
 ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

For an IPv6 network, the entry appears similar to the following:

```
# IPv6 Neighbor Discovery messages bypass IPsec.
{ulp ipv6-icmp type 133-137 dir both} pass {}
```

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 6000:6666::aaaa:1116 dir both} bypass {}

# WAN traffic uses ESP with AES and SHA-1.
{tunnel tun0 negotiate tunnel}
  ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

b. On the `partym` system, type the following entry into the `ipsecinit.conf` file:

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 10.1.3.3 dir both} bypass {}

# WAN traffic uses ESP with AES and SHA-1.
{tunnel tun0 negotiate tunnel}
 ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

For an IPv6 network, the entry appears similar to the following:

```
# IPv6 Neighbor Discovery messages bypass IPsec.
{ulp ipv6-icmp type 133-137 dir both} pass {}

# LAN traffic to and from this host can bypass IPsec.
{laddr 6000:3333::eeee:1113 dir both} bypass {}

# WAN traffic uses ESP with AES and SHA-1.
{tunnel tun0 negotiate tunnel}
  ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

5 (Optional) Verify the syntax of the IPsec policy file.

```
# ipsecconf -c -f /etc/inet/ipsecinit.conf
```

6 Create and configure the tunnel, *tunnel-name*.

For example, name the tunnel `tun0`, analogous to the `hme0` interface.

a. On the `enigma` system, create the tunnel and configure it.

- **For IPv4 addresses, the syntax appears similar to the following:**

```
# dladm create-iptun -T ipv4 -a local=192.168.116.16,remote=192.168.13.213 tun0
# ipadm create-addr -T static -a local=10.16.16.6,remote=10.1.3.3 tun0/to-calif
# ipadm set-ifprop -p forwarding=on -m ipv4 tun0
```

- **For IPv6 addresses, the syntax appears similar to the following:**

```
# dladm create-iptun -T ipv6
-a local=2001::aaaa:6666:6666,remote=2001::eeee:3333:3333 tun0
# ipadm create-addr -T static \
-a local=2001::aaaa:6666:6666,remote=2001::eeee:3333:3333 tun0/to-calif
# ipadm set-ifprop -p forwarding=on -m ipv6 tun0
```

For information about these commands, see the `dladm`(1M) and `ipadm`(1M) man pages, and "How to Configure an IP Interface" in *System Administration Guide: Network Interfaces and*

Network Virtualization. For information about creating customized names, such as `tun0`, see "Assigning Names to Datalinks" in *System Administration Guide: Network Interfaces and Network Virtualization*.

b. On the `partym` system, create the tunnel and configure it:

- **For IPv4 addresses, the syntax appears similar to the following:**

```
# dladm create-iptun -T ipv4 -a local=192.168.13.213,remote=192.168.116.16 tun0
# ipadm create-addr -T static -a local=10.1.3.3,remote=10.16.16.6 tun0/to-europe
# ipadm set-ifprop -p forwarding=on -m ipv4 tun0
```

- **For IPv6 addresses, the syntax appears similar to the following:**

```
# dladm create-iptun -T ipv6
-a local=2001::eeee:3333:3333,remote=2001::aaaa:6666:6666 tun0
# ipadm create-addr -T static
-a local=6000:3333::eeee:1113,remote=6000:6666::aaaa:1116 tun0/to-europe
# ipadm set-ifprop -p forwarding=on -m ipv6 tun0
```

7 Protect the tunnel with the IPsec policy that you created.

```
# svcadm refresh svc:/network/ipsec/policy:default
```

8 Restart the network services.

```
# svcadm restart svc:/network/initial:default
```

9 Turn on IP forwarding for the `hme1` interface.

a. On the `enigma` system, identify the interface and turn on forwarding:

- **For IPv4 addresses, the syntax appears similar to the following:**

```
# ipadm create-addr -T static -a 192.168.116.16 hme1/LAN
# ipadm set-ifprop -p forwarding=on -m ipv4 hme1
```

b. On the `partym` system, identify the interface and turn on forwarding:

- **For IPv4 addresses, the syntax appears similar to the following:**

```
# ipadm create-addr -T static -a 192.168.13.213 hme1/LAN
# ipadm set-ifprop -p forwarding=on -m ipv4 hme1
```

- **For IPv6 addresses, the syntax appears similar to the following:**

```
# ipadm create-addr -T static -a 2001::eeee:3333:3333 hme1/LAN
# ipadm set-ifprop -p forwarding=on -m ipv6 hme1
```

IP forwarding means that packets that arrive from somewhere else can be forwarded. IP forwarding also means that packets that leave this interface might have originated somewhere else. To successfully forward a packet, both the receiving interface and the transmitting interface must have IP forwarding turned on.

Because the hme1 interface is *inside* the intranet, IP forwarding must be turned on for hme1. Because ip.tun0 connects the two systems through the Internet, IP forwarding must be turned on for tun0.

The hme0 interface has its IP forwarding turned off to prevent an *outside* adversary from injecting packets into the protected intranet. The *outside* refers to the Internet.

10 Ensure that the routing protocols do not advertise the default route within the intranet.

a. On the enigma system, turn off advertising within the intranet.

```
# ipadm create-addr -T static -a 10.16.16.6 -p private=on hme0/WAN
```

For an IPv6 system, the syntax appears similar to the following:

```
# ipadm create-addr -T static -a 6000:6666::aaaa:1116 -p private=on hme0/WAN
```

b. On the partym system, turn off advertising within the intranet.

```
# ipadm create-addr -T static -a 10.1.3.3 -p private=on hme0/WAN
```

For an IPv6 system, the syntax appears similar to the following:

```
# ipadm create-addr -T static -a 6000:3333::eeee:1113 -p private=on hme0/WAN
```

Even if hme0 has IP forwarding turned off, a routing protocol implementation might still advertise the interface. For example, the in.routed protocol might still advertise that hme0 is available to forward packets to its peers inside the intranet. By setting the interface's *private* flag, these advertisements are prevented.

11 Manually add a default route over the hme0 interface.

The default route must be a router with direct access to the Internet.

a. On the enigma system, add the following route:

```
# route add default 192.168.116.4
```

For an IPv6 network, the syntax appears similar to the following:

```
# route add -inet6 default 2001::aaaa:0:4
```

b. On the partym system, add the following route:

```
# route add default 192.168.13.5
```

For an IPv6 network, the syntax appears similar to the following:

```
# route add -inet6 default 2001::eeee:0:1
```

Even though the hme0 interface is not part of the intranet, hme0 does need to reach across the Internet to its peer system. To find its peer, hme0 needs information about Internet routing. The VPN system appears to be a host, rather than a router, to the rest of the Internet.

Therefore, you can use a default router or run the router discovery protocol to find a peer system. For more information, see the `route(1M)` and `in.routed(1M)` man pages.

12 Run a routing protocol.

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

For an IPv6 system, use the `ipv6` prefix:

```
# routeadm -e ipv6-routing
# routeadm -u
```

You might need to configure the routing protocol before running the routing protocol. For more information, see "Routing Protocols in Oracle Solaris" on page 193. For a procedure, see "How to Configure an IPv4 Router" on page 91.

Example 19–9 Creating Temporary Tunnels When Testing

In this example, the administrator tests tunnel creation. Later, the administrator will use the procedure "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode" on page 429 to make the tunnels permanent. During testing, the administrator performs the following series of steps on the systems `system1` and `system2`:

- On both systems, the administrator completes the first three steps of "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode" on page 429. The administrator names the tunnel `test.tun0`.
- The administrator sets IPsec policy on both systems in the `ipsecinit.conf` file.

```
## SYSTEM 1 ipsecinit.conf file
##
## LAN traffic on system1.
{laddr 10.16.16.6 dir both} bypass {}

## WAN traffic on system1.
{tunnel test.tun0 negotiate tunnel}
 ipsec {encr_algs aes encr_auth_algs sha1 sa shared}


## SYSTEM 2 ipsecinit.conf file
##
## LAN traffic on system2.
{laddr 10.1.3.3 dir both} bypass {}

## WAN traffic on system2.
{tunnel test.tun0 negotiate tunnel}
 ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

  Note that the administrator names this tunnel `test.tun0`.

- The administrator verifies the syntax of the file.

```
# ipsecconf -c -f /etc/inet/ipsecinit.conf
```

- The administrator uses the `dladm` command to configure the `test.tun0` tunnel and the `ipadm` command to configure the tunnel IP interface over the link.

```
system1 # dladm create-iptun -T ipv4 \
-a local=192.168.116.16,remote=192.168.13.213 test.tun0
system1 # ipadm create-addr -T static \
-a local=192.168.116.16,remote=192.168.13.213 test.tun0/tunaddr

# ssh system2
Password:      admin-password-on-system2
system2 # dladm create-iptun -T ipv4 \
-a local=192.168.13.213,remote=192.168.116.16 test.tun0
system2 # ipadm create-addr -T static \
-a local=10.1.3.3,remote=10.1.3.3 test.tun0/tunaddr

system1 # dladm create-iptun -T ipv4 \
-a local=192.168.116.16,remote=192.168.13.213 test.tun0
system1 # ipadm create-addr -T static \
-a local=10.16.16.6,remote=10.1.3.3 test.tun0/tunaddr

# ssh system2
Password:      admin-password-on-system2
system2 # dladm create-iptun -T ipv4 \
-a local=192.168.13.213,remote=192.168.116.16 test.tun0
system2 # ipadm create-addr -T static \
-a local=10.1.3.3,remote=10.16.16.6 test.tun0/tunaddr
```

  For the syntax of the `ipadm` command, see the `ipadm(1M)` man page. For the syntax of the `dladm` command, see the `dladm(1M)` man page. For examples, see "How to Create and Configure an IP Tunnel" on page 161.

- The administrator enables IPsec policy on the tunnel.

```
system1 # svcadm refresh svc:/network/ipsec/policy:default
system2 # svcadm refresh svc:/network/ipsec/policy:default
```

- The administrator makes the Internet interface a router and prevents routing protocols from going over the intranet interface.

```
system1 # ipadm set-ifprop -p forwarding=on -m ipv4 hme1
system1 # ipadm set-addrpop -p private=on hme0/WAN


system2 # ipadm set-ifprop -p forwarding=on -m ipv4 hme1
system2 # ipadm set-addprop -p private=on ipv4 hme0/WAN
```

  For information about address objects, such as hme0/WAN, see the `ipadm(1M)` man page and "How to Configure an IP Interface" in *System Administration Guide: Network Interfaces and Network Virtualization*. For sample procedures that set IP address properties which references address objects, see "Setting IP Address Properties" in *System Administration Guide: Network Interfaces and Network Virtualization*.

- The administrator manually adds routing and runs the routing protocol by completing Step 11 and Step 12 of "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode" on page 429 on both systems.

Example 19–10 Creating a Tunnel to an Earlier Version of a Solaris System by Using the Command Line

In the Solaris 10 7/07 release, the syntax of the `ifconfig` command was simplified. In this example, the administrator tests tunnel creation to a system that is running a version of Solaris prior to the Solaris 10 7/07 release. By using the original syntax of the `ifconfig` command, the administrator can use identical commands on the two communicating systems. Later, the administrator will use "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode" on page 429 to make the tunnels permanent.

During testing, the administrator performs the following steps on the systems `system1` and `system2`:

- On both systems, the administrator completes the first five steps of "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode" on page 429.
- The administrator plumbs and configures the tunnel.

```
system1 # ifconfig ip.tun0 plumb
system1 # ifconfig ip.tun0 10.16.16.6 10.1.3.3 \
          tsrc 192.168.116.16 tdst 192.168.13.213 \
          encr_algs aes encr_auth_algs sha1
system1 # ifconfig ip.tun0 router up
```

```
# ssh system2
Password:      admin-password-on-system2
system2 # ifconfig ip.tun0 plumb
system2 # ifconfig ip.tun0 10.1.3.3  10.16.16.6 \
          tsrc 192.168.13.213 tdst 192.168.116.16 \
          encr_algs aes encr_auth_algs sha1
system2 # ifconfig ip.tun0 router up
```

- The administrator enables IPsec policy on the tunnel. The policy was created in Step 4 of "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode" on page 429.

```
system1 # svcadm refresh svc:/network/ipsec/policy:default
system2 # svcadm refresh svc:/network/ipsec/policy:default
```

- The administrator makes the Internet interface a router and prevents routing protocols from going over the intranet interface.

```
system1 # ifconfig hme1 router ; ifconfig hme0 private
system2 # ifconfig hme1 router ; ifconfig hme0 private
```

- The administrator adds routing by completing Step 11 and Step 12 of "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode" on page 429 on both systems.

Example 19–11 Requiring IPsec Policy on All Systems on a LAN

In this example, the administrator comments out the `bypass` policy that was configured in Step 4, thereby strengthening the protection. With this policy configuration, each system on the LAN must activate IPsec to communicate with the router.

```
# LAN traffic must implement IPsec.
# {laddr 10.1.3.3 dir both} bypass {}

# WAN traffic uses ESP with AES and SHA-1.
{tunnel ip.tun0 negotiate tunnel} ipsec {encr_algs aes encr_auth_algs sha1}
```

Example 19–12 Using IPsec to Protect Telnet Traffic Differently From SMTP Traffic

In this example, the first rule protects `telnet` traffic on port 23 with Blowfish and SHA-1. The second rule protects SMTP traffic on port 25 with AES and MD5.

```
{laddr 10.1.3.3 ulp tcp dport 23 dir both}
  ipsec {encr_algs blowfish encr_auth_algs sha1 sa unique}
{laddr 10.1.3.3 ulp tcp dport 25 dir both}
 ipsec {encr_algs aes encr_auth_algs md5 sa unique}
```

Example 19–13 Using an IPsec Tunnel in Tunnel Mode to Protect a Subnet Differently From Other Network Traffic

The following tunnel configuration protects all traffic from subnet `10.1.3.0/24` across the tunnel:

```
{tunnel ip.tun0 negotiate tunnel laddr 10.1.3.0/24}
  ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

The following tunnel configurations protect traffic from subnet `10.1.3.0/24` to different subnets across the tunnel. Subnets that begin with `10.2.`*x.x* are across the tunnel.

```
{tunnel ip.tun0 negotiate tunnel laddr 10.1.3.0/24 raddr 10.2.1.0/24}
  ipsec {encr_algs blowfish encr_auth_algs sha1 sa shared}

{tunnel ip.tun0 negotiate tunnel laddr 10.1.3.0/24 raddr 10.2.2.0/24}
  ipsec {encr_algs blowfish encr_auth_algs sha1 sa shared}

{tunnel ip.tun0 negotiate tunnel laddr 10.1.3.0/24 raddr 10.2.3.0/24}
  ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

▼ How to Protect a VPN With an IPsec Tunnel in Transport Mode

In transport mode, the outer header determines the IPsec policy that protects the inner IP packet.

This procedure extends the procedure "How to Secure Traffic Between Two Systems With IPsec" on page 411. In addition to connecting two systems, you are connecting two intranets that connect to these two systems. The systems in this procedure function as gateways.

This procedure uses the setup that is described in "Description of the Network Topology for the IPsec Tasks to Protect a VPN" on page 427. For a fuller description of the reasons for running particular commands, see the corresponding steps in "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode" on page 429.

Note – Perform the steps in this procedure on both systems.

1 Become an administrator.

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session. Use the `ssh` command for a secure remote login. For an example, see Example 19–1.

2 Control the flow of packets before configuring IPsec.

For a list of the steps, see Step 2 in "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode" on page 429.

3 Add a pair of SAs between the two systems.

Choose one of the following options:

- Configure IKE to manage the keys for the SAs. Use one of the procedures in "Configuring IKE (Task Map)" on page 459 to configure IKE for the VPN.
- If you have an overriding reason to manually manage the keys, see "How to Manually Create IPsec Security Associations" on page 417.

4 Add IPsec policy.

Edit the `/etc/inet/ipsecinit.conf` file to add the IPsec policy for the VPN. To strengthen the policy, see Example 19–14.

a. On the `enigma` system, type the following entry into the `ipsecinit.conf` file:

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 10.16.16.6 dir both} bypass {}

# WAN traffic uses ESP with AES and SHA-1.
{tunnel tun0 negotiate transport}
 ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

For an IPv6 system, the syntax appears similar to the following:

```
# IPv6 Neighbor Discovery messages bypass IPsec.
{ulp ipv6-icmp type 133-137 dir both} pass {}
```

```
# LAN traffic can bypass IPsec.
{laddr 6000:6666::aaaa:1116 dir both} bypass {}

# WAN traffic uses ESP with AES and SHA-1.
{tunnel tun0 negotiate transport}
 ipsec {encr_algs aes encr_auth_algs sha1}
```

b. On the `partym` system, type the following entry into the `ipsecinit.conf` file:

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 10.1.3.3 dir both} bypass {}

# WAN traffic uses ESP with AES and SHA-1.
{tunnel tun0 negotiate transport}
 ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

For an IPv6 system, the syntax appears similar to the following:

```
# IPv6 Neighbor Discovery messages bypass IPsec.
{ulp ipv6-icmp type 133-137 dir both} pass {}

# LAN traffic can bypass IPsec.
{laddr 6000:3333::eeee:1113 dir both} bypass {}

# WAN traffic uses ESP with AES and SHA-1.
{tunnel tun0 negotiate transport}
 ipsec {encr_algs aes encr_auth_algs sha1}
```

5 (Optional) Verify the syntax of the IPsec policy file.

```
# ipsecconf -c -f /etc/inet/ipsecinit.conf
```

6 Configure the tunnel.

Tunnel configuration for transport mode is the same as tunnel configuration for tunnel mode. For the steps, see Step 6 in "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode" on page 429.

7 Protect the tunnel with the IPsec policy that you created.

```
# svcadm refresh svc:/network/ipsec/policy:default
```

8 Restart the network services.

```
# svcadm restart svc:/network/initial:default
```

9 Turn on IP forwarding for the `hme1` interface.

a. On the `enigma` system, identify the interface and turn on forwarding:

```
# ipadm create-addr -T static -a 192.168.116.16 hme1/LAN
# ipadm set-ifprop -p forwarding=on -m ipv4 hme1
```

For IPv6 addresses, the syntax appears similar to the following:

```
# ipadm create-addr -T static -a 2001::aaaa:6666:6666 hme1/LAN
ipadm set-ifprop -p forwarding=on -m ipv6 hme1
```

b. On the `partym` system, identify the interface and turn on forwarding:

```
# ipadm create-addr -T static -a 192.168.13.213 hme1/LAN

# ipadm set-ifprop -p forwarding=on -m ipv4 hme1
```

For IPv6 addresses, the syntax appears similar to the following:

```
# ipadm create-addr -T static -a 2001::eeee:3333:3333 hme1/LAN
# ipadm set-ifprop -p forwarding=on -m ipv6 hme1
```

10 Ensure that routing protocols do not advertise the default route within the intranet.

a. On the `enigma` system, turn off advertising within the intranet.

```
# ipadm create-addr -T static -a 10.16.16.6 -p private=on hme0/WAN
```

For an IPv6 system, the syntax appears similar to the following:

```
# ipadm create-addr -T static -a 6000:6666::aaaa:1116 -p private=on hme0/WAN
```

b. On the `partym` system, turn off advertising within the intranet.

```
# ipadm create-addr -T static -a 10.1.3.3 -p private=on hme0/WAN
```

For an IPv6 system, the syntax appears similar to the following:

```
# ipadm create-addr -T static -a 6000:3333::eeee:1113 -p private=on hme0/WAN
```

11 Manually add a default route over `hme0`.

a. On the `enigma` system, add the following route:

```
# route add default 192.168.116.4
```

For an IPv6 network, the syntax appears similar to the following:

```
# route add -inet6 default 2001::eeee:0:1
```

b. On the `partym` system, add the following route:

```
# route add default 192.168.13.5
```

For an IPv6 network, the syntax appears similar to the following:

```
# route add -inet6 default 2001::eeee:0:1
```

12 Run a routing protocol.

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

For an IPv6 network, use the `ipv6` prefix:

```
# routeadm -e ipv6-routing
# routeadm -u
```

Example 19–14 Requiring IPsec Policy on All Systems in Transport Mode

In this example, the administrator comments out the `bypass` policy that was configured in Step 4, thereby strengthening the protection. With this policy configuration, each system on the LAN must activate IPsec to communicate with the router.

```
# LAN traffic must implement IPsec.
# {laddr 10.1.3.3 dir both} bypass {}

# WAN traffic uses ESP with AES and SHA-1.
{tunnel tun0 negotiate transport} ipsec {encr_algs aes encr_auth_algs sha1}
```

▼ How to Prevent IP Spoofing

To prevent the system from forwarding packets to another interface without trying to decrypt them, the system needs to check for IP spoofing. One method of prevention is to set the IP strict destination multihoming parameter by using the `ndd` command. When this parameter is set in an SMF manifest, the parameter is set when the system reboots.

Note – Perform the steps in this procedure on both systems.

---

1 Become an administrator.

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

2 Create the site-specific SMF manifest to check for IP spoofing.

Use the following sample script, `/var/svc/manifest/site/spoof_check.xml`.

```
<?xml version="1.0"?>
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">

<service_bundle type='manifest' name='Custom:ip_spoof_checking'>

<!--    This is a custom smf(5) manifest for this system. Place this
        file in /var/svc/manifest/site, the directory for local
        system customizations. The exec method uses an unstable
        interface to provide a degree of protection against IP
        spoofing attacks when this system is acting as a router.

        IP spoof protection can also be achieved by using ipfilter(5).
        If ipfilter is configured, this service can be disabled.

        Note: Unstable interfaces might be removed in later
        releases.  See attributes(5).
-->

<service
        name='site/ip_spoofcheck'
        type='service'
```

```
        version='1'>

        <create_default_instance enabled='false' />
        <single_instance />

        <!--    Don't enable spoof protection until the
                network is up.
        -->
        <dependency
                name='basic_network'
                grouping='require_all'
                restart_on='none'
                type='service'>
        <service_fmri value='svc:/milestone/network' />
        </dependency>

        <exec_method
                type='method'
                name='start'
                exec='/usr/sbin/ndd -set /dev/ip ip_strict_dst_multihoming 1'
<!--
     For an IPv6 network, use the IPv6 version of this command, as in:
                exec='/usr/sbin/ndd -set /dev/ip ip6_strict_dst_multihoming 1
-->
                timeout_seconds='60'
        />

        <exec_method
                type='method'
                name='stop'
                exec=':true'
                timeout_seconds='3'
        />

        <property_group name='startd' type='framework'>
                <propval
                        name='duration'
                        type='astring'
                        value='transient'
                />
        </property_group>

        <stability value='Unstable' />

</service>
</service_bundle>
```

3 Import this manifest into the SMF repository.

```
# svccfg import /var/svc/manifest/site/spoof_check.xml
```

4 Enable the `ip_spoofcheck` service.

Use the name that is defined in the manifest, `/site/ip_spoofcheck`.

```
# svcadm enable /site/ip_spoofcheck
```

5 Verify that the `ip_spoofcheck` service is online.

```
# svcs /site/ip_spoofcheck
```

◆ ◆ ◆ CHAPTER 20

IP Security Architecture (Reference)

This chapter contains the following reference information:

For instructions on how to implement IPsec on your network, see Chapter 19, "Configuring IPsec (Tasks)." For an overview of IPsec, see Chapter 18, "IP Security Architecture (Overview)."

IPsec Service Management Facility

The service management facility (SMF) provides the following services for IPsec:

- `svc:/network/ipsec/policy` **service** – Manages IPsec policy. By default, this service is enabled. The value of the `config_file` property determines the location of the `ipsecinit.conf` file. The initial value is `/etc/inet/ipsecinit.conf`.
- `svc:/network/ipsec/ipsecalgs` **service** – Manages the algorithms that are available to IPsec. By default, this service is enabled.
- `svc:/network/ipsec/manual-key` **service** – Activates manual key management. By default, this service is disabled. The value of the `config_file` property determines the location of the `ipseckeys`configuration file. The initial value is `/etc/inet/secret/ipseckeys`.
- `svc:/network/ipsec/ike` **service** – Manages IKE. By default, this service is disabled. For the configurable properties, see "IKE Service Management Facility" on page 501.

For information about SMF, see Chapter 11, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*. Also see the `smf`(5), `svcadm`(1M), and `svccfg`(1M) man pages.

ipsecconf Command

You use the ipsecconf command to configure the IPsec policy for a host. When you run the command to configure the policy, the system creates the IPsec policy entries in the kernel. The system uses these entries to check the policy on all inbound and outbound IP datagrams. Forwarded datagrams are not subjected to policy checks that are added by using this command. The ipsecconf command also configures the security policy database (SPD).

- For information about how to protect forwarded packets, see the ifconfig(1M) and ipadm(1M) man pages.
- For IPsec policy options, see the ipsecconf(1M) man page.
- For instructions about how to use the ipsecconf command to protect traffic between systems, see "Configuring IKE (Task Map)" on page 459.

You must become superuser or assume an equivalent role to invoke the ipsecconf command. The command accepts entries that protect traffic in both directions. The command also accepts entries that protect traffic in only one direction.

Policy entries with a format of local address and remote address can protect traffic in both directions with a single policy entry. For example, entries that contain the patterns laddr host1 and raddr host2 protect traffic in both directions, if no direction is specified for the named host. Thus, you need only one policy entry for each host.

Policy entries with a format of source address to destination address protect traffic in only one direction. For example, a policy entry of the pattern saddr host1 daddr host2 protects inbound traffic or outbound traffic, not both directions. Thus, to protect traffic in both directions, you need to pass the ipsecconf command another entry, as in saddr host2 daddr host1.

To ensure that the IPsec policy is active when the machine boots, you can create an IPsec policy file, /etc/inet/ipsecinit.conf. This file is read when the network services are started. For instructions on how to create an IPsec policy file, see "Protecting Traffic With IPsec (Task Map)" on page 409.

With the -c option, the ipsecconf command checks the syntax of the IPsec policy file that you provide as an argument.

Policy entries that are added by the ipsecconf command are not persistent over a system reboot. To ensure that the IPsec policy is active when the system boots, add the policy entries to the /etc/inet/ipsecinit.conf file, then refresh or enable the policy service. For examples, see "Protecting Traffic With IPsec (Task Map)" on page 409.

ipsecinit.conf File

To invoke IPsec security policies when you start the Solaris Operating System (Solaris OS), you create a configuration file to initialize IPsec with your specific IPsec policy entries. The default name for this file is /etc/inet/ipsecinit.conf. See the ipsecconf(1M) man page for details about policy entries and their format. After policies are configured, you can use the ipsecconf command to view the existing configuration.

Sample ipsecinit.conf File

The Solaris software includes a sample IPsec policy file, ipsecinit.sample. You can use the file as a template to create your own ipsecinit.conf file. The ipsecinit.sample file contains the following examples:

```
#
# For example,
#
#     {rport 23} ipsec {encr_algs des encr_auth_algs md5}
#
# will protect the telnet traffic originating from the host with ESP using
# DES and MD5. Also:
#
#     {raddr 10.5.5.0/24} ipsec {auth_algs any}
#
# will protect traffic to or from the 10.5.5.0 subnet with AH
# using any available algorithm.
#
#
# To do basic filtering, a drop rule may be used. For example:
#
#     {lport 23 dir in} drop {}
#     {lport 23 dir out} drop {}
# will disallow any remote system from telnetting in.
#
# If you are using IPv6, it may be useful to bypass neighbor discovery
# to allow in.iked to work properly with on-link neighbors. To do that,
# add the following lines:
#
#        {ulp ipv6-icmp type 133-137 dir both } pass { }
#
# This will allow neighbor discovery to work normally.
```

Security Considerations for ipsecinit.conf and ipsecconf

Use extreme caution if transmitting a copy of the ipsecinit.conf file over a network. An adversary can read a network-mounted file as the file is being read. If, for example, the /etc/inet/ipsecinit.conf file is accessed or is copied from an NFS-mounted file system, an adversary can change the policy that is contained in the file.

Ensure that you set up IPsec policies before starting any communications, because existing connections might be affected by the addition of new policy entries. Similarly, do not change policies in the middle of a communication.

Specifically, IPsec policy cannot be changed for SCTP, TCP, or UDP sockets on which a `connect()` or `accept()` function call has been issued. A socket whose policy cannot be changed is called a *latched socket*. New policy entries do not protect sockets that are already latched. For more information, see the `connect`(3SOCKET) and `accept`(3SOCKET) man pages.

Protect your naming system. If the following two conditions are met, then your host names are no longer trustworthy:

- Your source address is a host that can be looked up over the network.
- Your naming system is compromised.

Security weaknesses often arise from the misapplication of tools, not from the actual tools. You should be cautious when using the `ipsecconf` command. Use a console or other hard-connected TTY for the safest mode of operation.

`ipsecalgs` Command

The Solaris cryptographic framework provides authentication and encryption algorithms to IPsec. The `ipsecalgs` command can list the algorithms that each IPsec protocol supports. The `ipsecalgs` configuration is stored in the `/etc/inet/ipsecalgs` file. Typically, this file does not need to be modified. However, if the file needs to be modified, use the `ipsecalgs` command. The file must never be edited directly. The supported algorithms are synchronized with the kernel at system boot by the `svc:/network/ipsec/ipsecalgs:default` service.

The valid IPsec protocols and algorithms are described by the ISAKMP domain of interpretation (DOI), which is covered by RFC 2407. In a general sense, 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.

Specifically, the ISAKMP DOI defines the naming and numbering conventions for the valid IPsec algorithms and for their protocols, `PROTO_IPSEC_AH` and `PROTO_IPSEC_ESP`. Each algorithm is associated with exactly one protocol. These ISAKMP DOI definitions are in the `/etc/inet/ipsecalgs` file. The algorithm and protocol numbers are defined by the Internet Assigned Numbers Authority (IANA). The `ipsecalgs` command makes the list of algorithms for IPsec extensible.

For more information about the algorithms, refer to the `ipsecalgs`(1M) man page. For more information on the Solaris cryptographic framework, see Chapter 13, "Oracle Solaris Cryptographic Framework (Overview)," in *System Administration Guide: Security Services*.

Security Associations Database for IPsec

Information on key material for IPsec security services is maintained in a security associations database (SADB). Security associations (SAs) protect inbound packets and outbound packets. The SADBs are maintained by a user process, or possibly multiple cooperating processes, that send messages over a special kind of socket. This method of maintaining SADBs is analogous to the method that is described in the `route(7P)` man page. Only superuser or a user who has assumed an equivalent role can access the database.

The `in.iked` daemon and the `ipseckey` command use the `PF_KEY` socket interface to maintain SADBs. For more information on how SADBs handle requests and messages, see the `pf_key(7P)` man page.

Utilities for Key Generation in IPsec

The IKE protocol provides automatic key management for IPv4 and IPv6 addresses. See Chapter 22, "Configuring IKE (Tasks)," for instructions on how to set up IKE. The manual keying utility is the `ipseckey` command, which is described in the `ipseckey(1M)` man page.

You use the `ipseckey` command to manually populate the security associations database (SADB). Typically, manual SA generation is used when IKE is unavailable for some reason. However, if the SPI values are unique, manual SA generation and IKE can be used at the same time.

The `ipseckey` command can be used to view all SAs that are known to the system, whether the keys were added manually or by IKE. With the `-c` option, the `ipseckey` command checks the syntax of the keys file that you provide as an argument.

IPsec SAs that are added by the `ipseckey` command are not persistent over system reboot. To enable manually added SAs at system boot, add entries to the `/etc/inet/secret/ipseckeys` file, then enable the `svc:/network/ipsec/manual-key:default` service. For the procedure, see "How to Manually Create IPsec Security Associations" on page 417.

While the `ipseckey` command has only a limited number of general options, the command supports a rich command language. You can specify that requests be delivered by means of a programmatic interface specific for manual keying. For additional information, see the `pf_key(7P)` man page.

Security Considerations for `ipseckey`

The `ipseckey` command enables superuser or a role with the Network Security or Network IPsec Management rights profile to enter sensitive cryptographic keying information. If an adversary gains access to this information, the adversary can compromise the security of IPsec traffic.

You should consider the following issues when you handle keying material and use the `ipseckey` command:

- Have you refreshed the keying material? Periodic key refreshment is a fundamental security practice. Key refreshment guards against potential weaknesses of the algorithm and keys, and limits the damage of an exposed key.
- Is the TTY going over a network? Is the `ipseckey` command in interactive mode?
  - In interactive mode, the security of the keying material is the security of the network path for this TTY's traffic. You should avoid using the `ipseckey` command over a clear-text telnet or rlogin session.
  - Even local windows might be vulnerable to attacks by a concealed program that reads window events.
- Have you used the `-f` option? Is the file being accessed over the network? Can the file be read by the world?
  - An adversary can read a network-mounted file as the file is being read. You should avoid using a world-readable file that contains keying material.
  - Protect your naming system. If the following two conditions are met, then your host names are no longer trustworthy:
    - Your source address is a host that can be looked up over the network.
    - Your naming system is compromised.

Security weaknesses often arise from the misapplication of tools, not from the actual tools. You should be cautious when using the `ipseckey` command. Use a console or other hard-connected TTY for the safest mode of operation.

`snoop` Command and IPsec

The `snoop` command can parse AH and ESP headers. Because ESP encrypts its data, the `snoop` command cannot see encrypted headers that are protected by ESP. AH does not encrypt data. Therefore, traffic that is protected by AH can be inspected with the `snoop` command. The `-V` option to the command shows when AH is in use on a packet. For more details, see the `snoop`(1M) man page.

For a sample of verbose `snoop` output on a protected packet, see "How to Verify That Packets Are Protected With IPsec" on page 420.

◆ ◆ ◆ CHAPTER 21

Internet Key Exchange (Overview)

Internet Key Exchange (IKE) automates key management for IPsec. This chapter contains the following information about IKE:

For instructions on implementing IKE, see Chapter 22, "Configuring IKE (Tasks)." For reference information, see Chapter 23, "Internet Key Exchange (Reference)." For information about IPsec, see Chapter 18, "IP Security Architecture (Overview)."

Key Management With IKE

The management of keying material for IPsec security associations (SAs) is called *key management*. Automatic key management requires a secure channel of communication for the creation, authentication, and exchange of keys. The Solaris Operating System uses Internet Key Exchange (IKE) to automate key management. IKE easily scales to provide a secure channel for a large volume of traffic. IPsec SAs on IPv4 and IPv6 packets can take advantage of IKE.

When IKE is used on a system with a Sun Crypto Accelerator 1000 board or a Sun Crypto Accelerator 4000 or Sun Crypto Accelerator 6000 board, the public key operations can be offloaded to the accelerator. Operating system resources are not used for public key operations. When IKE is used on a system with a Sun Crypto Accelerator 4000 or Sun Crypto Accelerator 6000 board, the certificates, public keys, and private keys can be stored on the board. Key storage that is off the system provides an additional layer of protection.

IKE Key Negotiation

The IKE daemon, `in.iked`, negotiates and authenticates keying material for SAs in a protected manner. The daemon uses random seeds for keys from internal functions provided by the Solaris Operating System. IKE provides perfect forward secrecy (PFS). In PFS, the keys that protect data transmission are not used to derive additional keys. Also, seeds used to create data transmission keys are not reused. See the `in.iked(1M)` man page.

When the IKE daemon discovers a remote system's public encryption key, the local system can then use that key. The system encrypts messages by using the remote system's public key. The messages can be read only by that remote system. The IKE daemon performs its job in two phases. The phases are called *exchanges*.

IKE Key Terminology

The following table lists terms that are used in key negotiation, provides their commonly used acronyms, and gives a definition and use for each term.

TABLE 21–1 Key Negotiation Terms, Acronyms, and Uses

Key Negotiation Term	Acronym	Definition and Use
Key exchange		The process of generating keys for asymmetric cryptographic algorithms. The two main methods are RSA protocols and the Diffie-Hellman protocol.
Diffie-Hellman protocol	DH	A key exchange protocol that involves key generation and key authentication. Often called *authenticated key exchange*.
RSA protocol	RSA	A key exchange protocol that involves key generation and key transport. The protocol is named for its three creators, Rivest, Shamir, and Adleman.
Perfect forward secrecy	PFS	Applies to authenticated key exchange only. PFS ensures that long-term secret material for keys does not compromise the secrecy of the exchanged keys from previous communications. 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.
Oakley method		A method for establishing keys for Phase 2 in a secure manner. This protocol is analogous to the Diffie-Hellman method of key exchange. Similar to Diffie-Hellman, Oakley group key exchange involves key generation and key authentication. The Oakley method is used to negotiate PFS.

IKE Phase 1 Exchange

The Phase 1 exchange is known as *Main Mode*. In the Phase 1 exchange, IKE uses public key encryption methods to authenticate itself with peer IKE entities. The result is an Internet Security Association and Key Management Protocol (ISAKMP) security association (SA). An ISAKMP SA is a secure channel for IKE to negotiate keying material for the IP datagrams. Unlike IPsec SAs, the ISAKMP SAs are bidirectional, so only one security association is needed.

How IKE negotiates keying material in the Phase 1 exchange is configurable. IKE reads the configuration information from the `/etc/inet/ike/config` file. Configuration information includes the following:

- Global parameters, such as the names of public key certificates
- Whether perfect forward secrecy (PFS) is used
- The interfaces that are affected
- The security protocols and their algorithms
- The authentication method

The two authentication methods are preshared keys and public key certificates. The public key certificates can be self-signed. Or, the certificates can be issued by a certificate authority (CA) from a public key infrastructure (PKI) organization. Organizations include beTrusted, Entrust, GeoTrust, RSA Security, and Verisign.

IKE Phase 2 Exchange

The Phase 2 exchange is known as *Quick Mode*. In the Phase 2 exchange, IKE creates and manages the IPsec SAs between systems that are running the IKE daemon. IKE uses the secure channel that was created in the Phase 1 exchange to protect the transmission of keying material. The IKE daemon creates the keys from a random number generator by using the `/dev/random` device. The daemon refreshes the keys at a configurable rate. The keying material is available to algorithms that are specified in the configuration file for IPsec policy, `ipsecinit.conf`.

IKE Configuration Choices

The `/etc/inet/ike/config` configuration file contains IKE policy entries. For two IKE daemons to authenticate each other, the entries must be valid. Also, keying material must be available. The entries in the configuration file determine the method for using the keying material to authenticate the Phase 1 exchange. The choices are preshared keys or public key certificates.

The entry `auth_method preshared` indicates that preshared keys are used. Values for `auth_method` other than `preshared` indicate that public key certificates are to be used. Public key certificates can be self-signed, or the certificates can be installed from a PKI organization. For more information, see the `ike.config(4)` man page.

IKE With Preshared Keys

Preshared keys are created by an administrator on one system. The keys are then shared out of band with administrators of remote systems. You should take care to create large random keys and to protect the file and the out-of-band transmission. The keys are placed in the `/etc/inet/secret/ike.preshared` file on each system. The `ike.preshared` file is for IKE as the `ipseckeys` file is for IPsec. Any compromise of the keys in the `ike.preshared` file compromises all keys that are derived from the keys in the file.

One system's preshared key must be identical to its remote system's key. The keys are tied to a particular IP address or range of addresses. Keys are most secure when one administrator controls the communicating systems or subnets. For more information, see the `ike.preshared(4)` man page.

IKE With Public Key Certificates

Public key certificates eliminate the need for communicating systems to share secret keying material out of band. Public keys use the Diffie-Hellman protocol (DH) for authenticating and negotiating keys. Public key certificates come in two flavors. The certificates can be self-signed, or the certificates can be certified by a certificate authority (CA).

Self-signed public key certificates are created by you, the administrator. The `ikecert certlocal -ks` command creates the private part of the public-private key pair for the system. You then get the self-signed certificate output in X.509 format from the remote system. The remote system's certificate is input to the `ikecert certdb` command for the public part of the key pair. The self-signed certificates reside in the `/etc/inet/ike/publickeys` directory on the communicating systems. When you use the `-T` option, the certificates reside on attached hardware.

Self-signed certificates are a halfway point between preshared keys and CAs. Unlike preshared keys, a self-signed certificate can be used on a mobile machine or on a system that might be renumbered. To self-sign a certificate for a system without a fixed number, use a `DNS` (`www.example.org`) or `email` (`root@domain.org`) alternative name.

Public keys can be delivered by a PKI or a CA organization. You install the public keys and their accompanying CAs in the `/etc/inet/ike/publickeys` directory. When you use the `-T` option, the certificates reside on attached hardware. Vendors also issue certificate revocation lists (CRLs). Along with installing the keys and CAs, you are responsible for installing the CRL in the `/etc/inet/ike/crls` directory.

CAs have the advantage of being certified by an outside organization, rather than by the site administrator. In a sense, CAs are notarized certificates. As with self-signed certificates, CAs can be used on a mobile machine or on a system that might be renumbered. Unlike self-signed certificates, CAs can very easily scale to protect a large number of communicating systems.

IKE and Hardware Acceleration

IKE algorithms are computationally expensive, particularly in the Phase 1 exchange. Systems that handle a large number of exchanges can use a Sun Crypto Accelerator 1000 board to handle the public key operations. The Sun Crypto Accelerator 6000 and Sun Crypto Accelerator 4000 boards can also be used to handle expensive Phase 1 computations.

For information on how to configure IKE to offload its computations to the accelerator board, see "How to Configure IKE to Find the Sun Crypto Accelerator 1000 Board" on page 495. For information on how to store keys, see "How to Configure IKE to Find the Sun Crypto Accelerator 4000 Board" on page 496, and the `cryptoadm`(1M) man page.

IKE and Hardware Storage

Public key certificates, private keys, and public keys can be stored on a Sun Crypto Accelerator 6000 or Sun Crypto Accelerator 4000 board. For RSA encryption, the Sun Crypto Accelerator 4000 board supports keys up to 2048 bits. For DSA encryption, the board supports keys up to 1024 bits. The Sun Crypto Accelerator 6000 board supports the SHA-512 and ECC algorithms.

For information on how to configure IKE to access the board, see "How to Configure IKE to Find the Sun Crypto Accelerator 1000 Board" on page 495. For information on how to add certificates and public keys to the board, see "How to Generate and Store Public Key Certificates on Hardware" on page 482.

IKE Utilities and Files

The following table summarizes the configuration files for IKE policy, the storage locations for IKE keys, and the various commands and services that implement IKE. For more about services, see Chapter 11, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*.

TABLE 21–2 IKE Configuration Files, Key Storage Locations, Commands, and Services

File, Location, Command, or Service	Description	For More Information
`svc:/network/ipsec/ike`	The SMF service that manages IKE.	`smf`(5)
`/usr/lib/inet/in.iked` daemon	Internet Key Exchange (IKE) daemon. Activates automated key management when the `ike` service is enabled.	`in.iked`(1M)
`/usr/sbin/ikeadm` command	IKE administration command for viewing and modifying the IKE policy. Enables you to view IKE administrative objects, such as Phase 1 algorithms and available Diffie-Hellman groups.	`ikeadm`(1M)

TABLE 21–2 IKE Configuration Files, Key Storage Locations, Commands, and Services *(Continued)*

File, Location, Command, or Service	Description	For More Information
`/usr/sbin/ikecert` command	Certificate database management command for manipulating local databases that hold public key certificates. The databases can also be stored on an attached Sun Crypto Accelerator 4000 board.	`ikecert`(1M)
`/etc/inet/ike/config` file	Default configuration file for the IKE policy in the `/etc/inet` directory. Contains the site's rules for matching inbound IKE requests and preparing outbound IKE requests. If this file exists, the `in.iked` daemon starts when the `ike` service is enabled. The location of this file can be changed by the `svccfg` command.	`ike.config`(4)
`ike.preshared` file	Preshared keys file in the `/etc/inet/secret` directory. Contains secret keying material for authentication in the Phase 1 exchange. Used when configuring IKE with preshared keys.	`ike.preshared`(4)
`ike.privatekeys` directory	Private keys directory in the `/etc/inet/secret` directory. Contains the private keys that are part of a public-private key pair.	`ikecert`(1M)
`publickeys` directory	Directory in the `/etc/inet/ike` directory that holds public keys and certificate files. Contains the public key part of a public-private key pair.	`ikecert`(1M)
`crls` directory	Directory in the `/etc/inet/ike` directory that holds revocation lists for public keys and certificate files.	`ikecert`(1M)
Sun Crypto Accelerator 1000 board	Hardware that accelerates public key operations by offloading the operations from the operating system.	`ikecert`(1M)
Sun Crypto Accelerator 4000 board	Hardware that accelerates public key operations by offloading the operations from the operating system. The board also stores public keys, private keys, and public key certificates.	`ikecert`(1M)

◆ ◆ ◆ CHAPTER 22

Configuring IKE (Tasks)

This chapter describes how to configure the Internet Key Exchange (IKE) for your systems. After IKE is configured, it automatically generates keying material for IPsec on your network. This chapter contains the following information:

For overview information about IKE, see Chapter 21, "Internet Key Exchange (Overview)." For reference information about IKE, see Chapter 23, "Internet Key Exchange (Reference)." For more procedures, see the Examples sections of the `ikeadm`(1M), `ikecert`(1M), and `ike.config`(4) man pages.

Displaying IKE Information

You can view the algorithms and groups that can be used in Phase 1 IKE negotiations.

▼ How to Display Available Groups and Algorithms for Phase 1 IKE Exchanges

In this procedure, you determine which Diffie-Hellman groups are available for use in Phase 1 IKE exchanges. You also view the encryption and authentication algorithms that are available for IKE Phase 1 exchanges. The numeric values match the values that are specified for these algorithms by the Internet Assigned Numbers Authority (IANA).

1 Display the list of Diffie-Hellman groups that IKE can use in Phase 1.

Diffie-Hellman groups set up IKE SAs.

```
# ikeadm dump groups
Value Strength Description
1     66       ietf-ike-grp-modp-768
2     77       ietf-ike-grp-modp-1024
5     91       ietf-ike-grp-modp-1536
14    110      ietf-ike-grp-modp-2048
15    130      ietf-ike-grp-modp-3072
16    150      ietf-ike-grp-modp-4096
17    170      ietf-ike-grp-modp-6144
18    190      ietf-ike-grp-modp-8192

Completed dump of groups
```

You would use one of these values as the argument to the `oakley_group` parameter in an IKE Phase 1 transform, as in:

```
p1_xform
  { auth_method preshared oakley_group 15 auth_alg sha encr_alg des }
```

2 Display the list of authentication algorithms that IKE can use in Phase 1.

```
# ikeadm dump authalgs
Value Name
1     md5
2     sha1
4     sha256
5     sha384
6     sha512

Completed dump of authalgs
```

You would use one of these names as the argument to the `auth_alg` parameter in an IKE Phase 1 transform, as in:

```
p1_xform
  { auth_method preshared oakley_group 15 auth_alg sha256 encr_alg des }
```

3 Display the list of encryption algorithms that IKE can use in Phase 1.

```
# ikeadm dump encralgs
Value Name
3     blowfish-cbc
5     3des-cbc
1     des-cbc
7     aes-cbc

Completed dump of encralgs
```

You would use one of these names as the argument to the `encr_alg` parameter in an IKE Phase 1 transform, as in:

```
p1_xform
  { auth_method preshared oakley_group 15 auth_alg sha encr_alg aes-cbc }
```

See Also For tasks to configure IKE rules that require these values, see "Configuring IKE (Task Map)" on page 459.

Configuring IKE (Task Map)

You can use preshared keys, self-signed certificates, and certificates from a Certificate Authority (CA) to authenticate IKE. A rule links the particular IKE authentication method with the end points that are being protected. Therefore, you can use one or all IKE authentication methods on a system. A pointer to a PKCS #11 library enables certificates to use an attached hardware accelerator.

After configuring IKE, complete the IPsec task that uses the IKE configuration. The following table refers you to task maps that focus on a specific IKE configuration.

Task	Description	For Instructions
Configure IKE with preshared keys	Protects communications between two systems by having the systems share a secret key.	"Configuring IKE With Preshared Keys (Task Map)" on page 459
Configure IKE with public key certificates	Protects communications with public key certificates. The certificates can be self-signed, or they can be vouched for by a PKI organization.	"Configuring IKE With Public Key Certificates (Task Map)" on page 470
Cross a NAT boundary	Configures IPsec and IKE to communicate with a mobile system	"Configuring IKE for Mobile Systems (Task Map)" on page 487
Configure IKE to generate and store public key certificates on attached hardware	Enables a Sun Crypto Accelerator 1000 board or a Sun Crypto Accelerator 4000 board to accelerate IKE operations. Also enables the Sun Crypto Accelerator 4000 board to store public key certificates.	"Configuring IKE to Find Attached Hardware (Task Map)" on page 494
Tune Phase 1 key negotiation parameters	Changes the timing of IKE key negotiations.	"Changing IKE Transmission Parameters (Task Map)" on page 497

Configuring IKE With Preshared Keys (Task Map)

The following table points to procedures to configure and maintain IKE with preshared keys.

Task	Description	For Instructions
Configure IKE with preshared keys	Creates an IKE policy file and one key to be shared.	"How to Configure IKE With Preshared Keys" on page 460
Refresh preshared keys on a running IKE system	Adds fresh keying material for IKE on communicating systems.	"How to Refresh IKE Preshared Keys" on page 465

Task	Description	For Instructions
Add preshared keys to a running IKE system	Adds a new IKE policy entry and new keying material to a system that is currently enforcing IKE policy.	"How to Add an IKE Preshared Key for a New Policy Entry in `ipsecinit.conf`" on page 466
Check that preshared keys are identical	Displays the preshared keys on both systems to see that the keys are identical.	"How to Verify That IKE Preshared Keys Are Identical" on page 469

Configuring IKE With Preshared Keys

Preshared keys is the simplest authentication method for IKE. If you are configuring two systems to use IKE, and you are the administrator for both of the systems, using preshared keys is a good choice. However, unlike public key certificates, preshared keys are tied to particular IP addresses. You can associate preshared keys with subnets or ranges of IP addresses. Preshared keys cannot be used with mobile systems or systems that might be renumbered, unless the renumbering is within the previously determined range of IP addresses. Also, when you use preshared keys, you cannot offload IKE computations to attached hardware.

▼ How to Configure IKE With Preshared Keys

The IKE implementation offers algorithms whose keys vary in length. The key length that you choose is determined by site security. In general, longer keys provide more security than shorter keys.

In this procedure, you generate keys in hexadecimal format. You can translate a hexadecimal key into ASCII format to interoperate with other vendors, as shown in Example 22–1. Alternatively, you can use an ASCII passphrase, as shown in Example 22–2. Also, you can use one key with a range of systems by using an IP prefix in the IKE rule. For an example, see Example 22–3.

These procedures use the system names `enigma` and `partym`. Substitute the names of your systems for the names `enigma` and `partym`.

Note – To use IPsec with labels on a Trusted Extensions system, see the extension of this procedure in "How to Apply IPsec Protections in a Multilevel Trusted Extensions Network" in *Oracle Solaris Trusted Extensions Configuration and Administration*.

---

1 Become an administrator.

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session. Use the `ssh` command for a secure remote login. For an example, see Example 19–1.

2 On each system, copy the file `/etc/inet/ike/config.sample` to the file `/etc/inet/ike/config`.

3 Enter rules and global parameters in the `ike/config` file on each system.

The rules and global parameters in this file should permit the IPsec policy in the system's `ipsecinit.conf` file to succeed. The following `ike/config` examples work with the `ipsecinit.conf` examples in "How to Secure Traffic Between Two Systems With IPsec" on page 411.

a. For example, modify the `/etc/inet/ike/config` file on the `enigma` system:

```
### ike/config file on enigma, 192.168.116.16

## Global parameters
#
## Phase 1 transform defaults
p1_lifetime_secs 14400
p1_nonce_len 40
#
## Defaults that individual rules can override.
p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg des }
p2_pfs 2
#
## The rule to communicate with partym
#  Label must be unique
{ label "enigma-partym"
  local_addr 192.168.116.16
  remote_addr 192.168.13.213
  p1_xform
   { auth_method preshared oakley_group 5 auth_alg sha1 encr_alg aes }
  p2_pfs 5
}
```

Note – All arguments to the `auth_method` parameter must be on the same line.

b. Modify the `/etc/inet/ike/config` file on the `partym` system:

```
### ike/config file on partym, 192.168.13.213
## Global Parameters
#
p1_lifetime_secs 14400
p1_nonce_len 40
#
p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg des }
```

```
p2_pfs 2

## The rule to communicate with enigma
#  Label must be unique
{ label "partym-enigma"
  local_addr 192.168.13.213
  remote_addr 192.168.116.16
p1_xform
   { auth_method preshared oakley_group 5 auth_alg sha1 encr_alg aes }
p2_pfs 5
}
```

4 On each system, verify the syntax of the file.

```
# /usr/lib/inet/in.iked -c -f /etc/inet/ike/config
```

5 Generate random numbers for use as keying material.

If your site has a random number generator, use that generator. On a Solaris system, you can use the `od` command. For example, the following command prints two lines of hexadecimal numbers:

```
% od -X -A n /dev/random | head -2
        f47cb0f4 32e14480 951095f8 2b735ba8
        0a9467d0 8f92c880 68b6a40e 0efe067d
```

For an explanation of the `od` command, see "How to Generate Random Numbers on a Solaris System" on page 416 and the `od(1)` man page.

Note – Other operating systems can require ASCII keying material. To generate the identical key in hexadecimal and ASCII formats, see Example 22–1.

---

6 From the output of Step 5, construct one key.

```
f47cb0f432e14480951095f82b735ba80a9467d08f92c88068b6a40e
```

The authentication algorithm in this procedure is SHA–1, as shown in Step 3. The size of the hash, that is, the size of the authentication algorithm's output, determines the minimum recommended size of a preshared key. The output of the SHA–1 algorithm is 160 bits, or 40 characters. The example key is 56 characters long, which provides additional keying material for IKE to use.

7 Create the file `/etc/inet/secret/ike.preshared` on each system.

Put the preshared key in each file.

a. For example, on the `enigma` system, the `ike.preshared` file would appear similar to the following:

```
# ike.preshared on enigma, 192.168.116.16
#...
{ localidtype IP
    localid 192.168.116.16
    remoteidtype IP
```

```
    remoteid 192.168.13.213
    # enigma and partym's shared key in hex (192 bits)
    key f47cb0f432e14480951095f82b735ba80a9467d08f92c88068b6a40e
    }
```

b. On the `partym` system, the `ike.preshared` file would appear similar to the following:

```
# ike.preshared on partym, 192.168.13.213
#...
{ localidtype IP
    localid 192.168.13.213
    remoteidtype IP
    remoteid 192.168.116.16
    # partym and enigma's shared key in hex (192 bits)
    key f47cb0f432e14480951095f82b735ba80a9467d08f92c88068b6a40e
    }
```

Note – The preshared keys on each system must be identical.

Example 22–1 Generating Identical Keying Material for Two Systems With Different Operating Systems

Solaris IPsec interoperates with other operating systems. If your system is communicating with a system that requires ASCII preshared keys, you need to generate one key in two formats, hexadecimal and ASCII.

In this example, the Solaris system administrator wants 56 characters of keying material. The administrator uses the following command to generate a hexadecimal key from an ASCII passphrase. The option `-tx1` prints the bytes one at a time on all Solaris systems.

```
# /bin/echo "papiermache with cashews and\c" | od -tx1 | cut -c 8-55 | \
tr -d '\n' | tr -d ' ' | awk '{print}'
7061706965726d6163686520776974682063617368657773206 16e64
```

By removing the offsets and concatenating the hexadecimal output, the hexadecimal key for the Solaris system is `7061706965726d6163686520776974682063617368657773206 16e64`. The administrator places this value in the `ike.preshared` file on the Solaris system.

```
# Shared key in hex (192 bits)
key 7061706965726d6163686520776974682063617368657773206 16e64
```

On the system that requires ASCII preshared keys, the passphrase is the preshared key. The Solaris system administrator telephones the other administrator with the passphrase, `papiermache with cashews and`.

Example 22–2 Using ASCII Keying Material for IKE Preshared Keys

In this example, the Solaris system administrator wants to use a passphrase to create the keying material for the IKE preshared key. The administrator uses special characters to complicate the passphrase. The string is typed on one line. The backslash escapes the double quotation mark that is part of the passphrase.

First, the administrator places the passphrase in the preshared rule on the `enigma` system.

```
#...
{ localidtype IP
    localid 192.168.116.16
    remoteidtype IP
    remoteid 192.168.13.213
    # enigma and partym's shared passphrase for keying material
key "LOoooong key Th@t m^st Be Ch*angEd \"reguLarLy."
    }
```

Then, the administrator places the passphrase in the preshared rule on the `partym` system.

```
#...
{ localidtype IP
    localid 192.168.13.213
    remoteidtype IP
    remoteid 192.168.116.16
    # partym and enigma's shared passphrase for keying material
key "LOoooong key Th@t m^st Be Ch*angEd \"reguLarLy."
    }
```

Example 22–3 Using an IKE Preshared Key to Protect a Range of Systems

In this example, the Solaris system administrator wants to use one IKE preshared key to protect a range of hosts. The administrator uses a passphrase to generate the keying material as well as subnet addresses to specify the range of hosts that use the IKE preshared key.

First, the administrator places the following IKE rule on every host in the `192.168.116.0/24` subnet.

```
#...
{ localidtype IP
    localid 192.168.116.0/24
    remoteidtype IP
    remoteid 192.168.13.0/24
    # enigma and partym's shared passphrase for keying material
key "LOoooong key Th@t m^st Be Ch*angEd \"reguLarLy."
    }
```

The administrator places the following IKE rule on every host in the `192.168.13.0/24` subnet.

```
#...
{ localidtype IP
    localid 192.168.116.0/24
```

```
    remoteidtype IP
    remoteid 192.168.13.0/24
    # partym and enigma's shared passphrase for keying material
key "LOooong key Th@t m^st Be Ch*angEd \"reguLarLy."
    }
```

How to Refresh IKE Preshared Keys

This procedure assumes that you want to replace an existing preshared key at regular intervals.

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session. Use the `ssh` command for a secure remote login. For an example, see Example 19–1.

---

2 **Generate random numbers and construct a key of the appropriate length.**

For details, see "How to Generate Random Numbers on a Solaris System" on page 416. If you are generating a preshared key for a Solaris system that is communicating with an operating system that requires ASCII, see Example 22–1.

Note – You can instead use an ASCII key for the preshared key, as in Example 22–2.

---

3 **Replace the current key with a new key.**

For example, on the hosts `enigma` and `partym`, you would replace the value of `key` in the `/etc/inet/secret/ike.preshared` file with a new number of the same length.

4 **Refresh the IKE keys.**

```
# svcadm refresh ike
```

How to View IKE Preshared Keys

By default, the `ikeadm` command prevents you from viewing the actual keys in a dump of a Phase 1 SA. Viewing the keys is useful during debugging.

To view the actual keys, you must increase the privilege level of the daemon. For a description of the privilege levels, see "IKE Administration Command" on page 503.

Before You Begin IKE is configured and the `ike` service is running.

1. **View the IKE preshared keys.**

```
# ikeadm
ikeadm> dump preshared
```

2. **If you get an error, increase the privilege level of the `in.iked` daemon.**

   a. **Increase the privilege level of the `in.iked` daemon in the SMF repository.**

```
# svcprop -p config/admin_privilege ike
base
# svccfg -s ike setprop config/admin_privilege=keymat
```

   b. **Increase the privilege level of the running `in.iked` daemon.**

```
# svcadm refresh ike ; svcadm restart ike
```

   c. **(Optional) Confirm that the privilege level is `keymat`.**

```
# svcprop -p config/admin_privilege ike
keymat
```

   d. **View the keys by running Step 1 again.**

3. **Return the IKE daemon to the base privilege level.**

   a. **After you view the keys, return the privilege level to the default.**

```
# svccfg -s ike setprop config/admin_privilege=base
```

   b. **Refresh and then restart IKE.**

```
# svcadm refresh ike ; svcadm restart ike
```

▼ How to Add an IKE Preshared Key for a New Policy Entry in `ipsecinit.conf`

If you add IPsec policy entries while IPsec and IKE are running, you restart the `policy` service and refresh the `ike` service after you add the new keys.

Before You Begin This procedure assumes the following:

- The `enigma` system is set up as described in "How to Configure IKE With Preshared Keys" on page 460.
- The `enigma` system is going to protect its traffic with a new system, `ada`.
- The `in.iked` daemon is running on both systems.
- The systems' interfaces are included as entries in the `/etc/hosts` file on both systems. The following entry is an example.

```
192.168.15.7 ada
192.168.116.16 enigma
```

This procedure also works with an IPv6 address. IPv6 addresses are placed in the `/etc/hosts` file.

- You have added a new policy entry to the `/etc/inet/ipsecinit.conf` file on both systems. The entries appear similar to the following:

```
# ipsecinit.conf file for enigma
{laddr enigma raddr ada} ipsec {auth_algs any encr_algs any sa shared}

# ipsecinit.conf file for ada
{laddr ada raddr enigma} ipsec {auth_algs any encr_algs any sa shared}
```

- You have verified the syntax of the `/etc/inet/ipsecinit.conf` file on both systems by using the following:

```
# ipsecconf -c -f /etc/inet/ipsecinit.conf
```

1 Become an administrator.

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session. Use the `ssh` command for a secure remote login. For an example, see Example 19–1.

---

2 On this system, generate random numbers and construct a key of 64 to 448 bits.

For details, see "How to Generate Random Numbers on a Solaris System" on page 416. If you are generating a preshared key for a Solaris system that is communicating with an operating system that requires ASCII, see Example 22–1.

Note – You can also instead an ASCII key for the preshared key, as in Example 22–2.

---

3 By some means, send the key to the administrator of the remote system.

You both need to add the same preshared key at the same time. Your key is only as safe as the safety of your transmission mechanism. An out-of-band mechanism, such as registered mail or a protected fax machine, is best. You can also use an `ssh` session to administer both systems.

4 Create a rule for IKE to manage the keys for `enigma` and `ada`.

a. On the `enigma` system, add the following rule to the `/etc/inet/ike/config` file:

```
### ike/config file on enigma, 192.168.116.16

## The rule to communicate with ada
```

```
{label "enigma-to-ada"
 local_addr 192.168.116.16
 remote_addr 192.168.15.7
 p1_xform
 {auth_method preshared oakley_group 5 auth_alg sha1 encr_alg blowfish}
 p2_pfs 5
    }
```

b. On the `ada` system, add the following rule:

```
### ike/config file on ada, 192.168.15.7

## The rule to communicate with enigma

{label "ada-to-enigma"
 local_addr 192.168.15.7
 remote_addr 192.168.116.16
 p1_xform
 {auth_method preshared oakley_group 5 auth_alg sha1 encr_alg blowfish}
 p2_pfs 5
}
```

5 Ensure that IKE preshared keys are available at reboot.

a. On the `enigma` system, add the following information to the `/etc/inet/secret/ike.preshared` file:

Note – If you created an ASCII key, place it in quotation marks as the argument to the `key` parameter.

```
# ike.preshared on enigma for the ada interface
#
{ localidtype IP
  localid 192.168.116.16
  remoteidtype IP
  remoteid 192.168.15.7
  # enigma and ada's shared key in hex (32 - 448 bits required)
  key 8d1fb4ee500e2bea071deb2e781cb48374411af5a9671714672bb1749ad9364d
}
```

b. On the `ada` system, add the following information to the `ike.preshared` file:

```
# ike.preshared on ada for the enigma interface
#
{ localidtype IP
  localid 192.168.15.7
  remoteidtype IP
  remoteid 192.168.116.16
  # ada and enigma's shared key in hex (32 - 448 bits required)
  key 8d1fb4ee500e2bea071deb2e781cb48374411af5a9671714672bb1749ad9364d
}
```

6 On each system, restart the IPsec policy service to secure the added interface.

```
# svcadm restart policy
```

7 **On each system, refresh the `ike` service.**

```
# svcadm refresh ike
```

8 **Verify that the systems can communicate.**

See "How to Verify That IKE Preshared Keys Are Identical" on page 469.

Example 22–4 Adding an IKE Preshared Key for a New IPsec Policy Entry

In the following example, the administrator is adding preshared key to a Solaris system that is not running the current Solaris release. The administrator follows the preceding procedure to modify the `ike/config` and `ike.preshared` files, and to generate keys and contact the remote system. The administrator uses different commands to read the new IPsec policy and IKE rules into the kernel.

- Before generating the new key, the administrator sets the privilege level of the `in.iked` daemon to 2.

```
# pkill in.iked
# /usr/lib/inet/in.iked -p 2
Setting privilege level to 2
```

- After sending the key to the other system and adding the new key to the system, the administrator lowers the privilege level.

```
# ikeadm set priv base
```

- Then, the administrator reads the new IPsec policy into the kernel.

```
# ipsecconf -a /etc/inet/ipsecinit.conf
```

- Finally, the administrator reads the new IKE rules into the kernel.

```
# ikeadm read rules
```

▼ How to Verify That IKE Preshared Keys Are Identical

If the preshared keys on the communicating systems are not identical, the systems cannot authenticate.

Before You Begin IPsec has been configured and is enabled between the two systems that you are testing.

1 **On each system, check the privilege level of the `in.iked` daemon.**

```
# svcprop -p config/admin_privilege ike
base
```

- **If the privilege level is `keymat`, continue with Step 2.**

- **If the privilege level is `base` or `modkeys`, increase the privilege level.**

  Then, refresh and restart the `ike` service.

  ```
  # svccfg -s ike setprop config/admin_privilege=keymat
  # svcadm refresh ike ; svcadm restart ike
  # svcprop -p config/admin_privilege ike
  keymat
  ```

2. **On each system, view the preshared key information.**

   ```
   # ikeadm dump preshared
   PSKEY: Preshared key (24 bytes): f47cb.../192
   LOCIP: AF_INET: port 0, 192.168.116.16 (enigma).
   REMIP: AF_INET: port 0, 192.168.13.213 (partym).
   ```

3. **Compare the two dumps.**

   If the preshared keys are not identical, replace one key with the other key in the `/etc/inet/secret/ike.preshared` file.

4. **When the verification is complete, return the privilege level to the default on each system.**

   ```
   # svccfg -s ike setprop config/admin_privilege=base
   # svcadm restart ike
   ```

Configuring IKE With Public Key Certificates (Task Map)

The following table provides pointers to procedures for creating public key certificates for IKE. The procedures include how to accelerate and store the certificates on attached hardware.

Task	Description	For Instructions
Configure IKE with self-signed public key certificates	Creates and places two certificates on each system: ■ A self-signed certificate ■ The public key certificate from the remote system	"How to Configure IKE With Self-Signed Public Key Certificates" on page 471
Configure IKE with a PKI Certificate Authority	Creates a certificate request, and then places three certificates on each system: ■ The certificate that the Certificate Authority (CA) creates from your request ■ The public key certificate from the CA ■ The CRL from the CA	"How to Configure IKE With Certificates Signed by a CA" on page 477

Task	Description	For Instructions
Configure public key certificates on local hardware	Involves one of: ■ Generating a self-signed certificate on the local hardware and then adding the public key from a remote system to the hardware. ■ Generating a certificate request on the local hardware and then adding the public key certificates from the CA to the hardware.	"How to Generate and Store Public Key Certificates on Hardware" on page 482
Update the certificate revocation list (CRL) from a PKI	Accesses the CRL from a central distribution point.	"How to Handle a Certificate Revocation List" on page 485

Note – To label packets and IKE negotiations on a Trusted Extensions system, follow the procedures in "Configuring Labeled IPsec (Task Map)" in *Oracle Solaris Trusted Extensions Configuration and Administration*.

Public key certificates are managed in the global zone on Trusted Extensions systems. Trusted Extensions does not change how certificates are managed and stored.

Configuring IKE With Public Key Certificates

Public key certificates eliminate the need for communicating systems to share secret keying material out of band. Unlike preshared keys, a public key certificate can be used on a mobile machine or on a system that might be renumbered.

Public key certificates can also be stored on attached hardware. For the procedure, see "Configuring IKE to Find Attached Hardware (Task Map)" on page 494.

▼ How to Configure IKE With Self-Signed Public Key Certificates

Self-signed certificates require less overhead than public certificates from a CA, but do not scale very easily.

1 Add a self-signed certificate to the `ike.privatekeys` database.

```
# ikecert certlocal -ks|-kc -m keysize -t keytype \
-D dname -A altname \
[-S validity-start-time] [-F validity-end-time] [-T token-ID]
```

`-ks` Creates a self-signed certificate.

`-kc`	Creates a certificate request. For the procedure, see "How to Configure IKE With Certificates Signed by a CA" on page 477.
`-m` *keysize*	Is the size of the key. The *keysize* can be 512, 1024, 2048, 3072, or 4096.
`-t` *keytype*	Specifies the type of algorithm to use. The *keytype* can be `rsa-sha1`, `rsa-md5`, or `dsa-sha1`.
`-D` *dname*	Is the X.509 distinguished name for the certificate subject. The *dname* typically has the form: C=country, O=organization, OU=organizational unit, CN=common name. Valid tags are C, O, OU, and CN.
`-A` *altname*	Is the alternate name for the certificate. The *altname* is in the form of `tag=value`. Valid tags are `IP`, `DNS`, `email`, and `DN`.
`-S` *validity-start-time*	Provides an absolute or relative valid start time for the certificate.
`-F` *validity-end-time*	Provides an absolute or relative valid end time for the certificate.
`-T` *token-ID*	Enables a PKCS #11 hardware token to generate the keys. The certificates are then stored in the hardware.

a. For example, the command on the `partym` system would appear similar to the following:

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
-D "C=US, O=PartyCompany, OU=US-Partym, CN=Partym" \
-A IP=192.168.13.213
Creating software private keys.
  Writing private key to file /etc/inet/secret/ike.privatekeys/0.
Enabling external key providers - done.
Acquiring private keys for signing - done.
Certificate:
 Proceeding with the signing operation.
 Certificate generated successfully (.../publickeys/0)
Finished successfully.
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
MIICLTCCAZagAwIBAgIBATANBgkqhkiG9w0BAQQFADBNMQswCQYDVQQGEwJVUzEX
...
6sKTxpg4GP3GkQGcd0r1rhW/3yaWBkDwOdFCqEUyffzU
-----END X509 CERTIFICATE-----
```

b. The command on the `enigma` system would appear similar to the following:

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
-D "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax" \
-A IP=192.168.116.16
Creating software private keys.
  ...
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
MIICKDCCAZGgAwIBAgIBATANBgkqhkiG9w0BAQQFADBJMQswCQYDVQQGEwJVUzEV
...
```

```
jpxfLM98xyFVyLCbkr3dZ3Tvxvi732BXePKF2A==
-----END X509 CERTIFICATE-----
```

2 Save the certificate and send it to the remote system.

You can paste the certificate into an email.

a. For example, you would send the following `partym` certificate to the `enigma` administrator:

```
To: admin@ja.enigmaexample.com
From: admin@us.partyexample.com
Message: -----BEGIN X509 CERTIFICATE-----
MIICLTCCAZagAwIBAgIBATANBgkqhkiG9w0BAQQFADBNMQswCQYDVQQGEwJVUzEX
...
6sKTxpg4GP3GkQGcd0r1rhW/3yaWBkDwOdFCqEUyffzU
-----END X509 CERTIFICATE-----
```

b. The `enigma` administrator would send you the following `enigma` certificate:

```
To: admin@us.partyexample.com
From: admin@ja.enigmaexample.com
Message: -----BEGIN X509 CERTIFICATE-----
MIICKDCCAZGgAwIBAgIBATANBgkqhkiG9w0BAQQFADBJMQswCQYDVQQGEwJVUzEV
...
jpxfLM98xyFVyLCbkr3dZ3Tvxvi732BXePKF2A==
-----END X509 CERTIFICATE-----
```

3 On each system, add the certificate that you received.

a. Copy the public key from the administrator's email.

b. Type the `ikecert certdb -a` command and press the Return key.

No prompts display when you press the Return key.

```
# ikecert certdb -a       Press the Return key
```

c. Paste the public key. Then press the Return key. To end the entry, press Control-D.

```
-----BEGIN X509 CERTIFICATE-----
MIIC...
...
----END X509 CERTIFICATE-----       Press the Return key
<Control>-D
```

4 Verify with the other administrator that the certificate is from that administrator.

For example, you can telephone the other administrator to compare the values of the public key hash. The public key hash for the shared certificate must be identical on the two systems.

a. List the stored certificate on your system.

For example, on the `partym` system, the public certificate is in slot 1, and the private certificate is in slot 0.

```
partym # ikecert certdb -l
Certificate Slot Name: 0   Type: rsa-md5      Private Key
    Subject Name: <C=US, O=PartyCompany, OU=US-Partym, CN=Partym>
    Key Size: 1024
    Public key hash: B2BD13FCE95FD27ECE6D2DCD0DE760E2

Certificate Slot Name: 1   Type: rsa-md5      Public Certificate
    (Private key in certlocal slot 0)      Points to certificate's private key
    Subject Name: <C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax>
    Key Size: 1024
    Public key hash: 2239A6A127F88EE0CB40F7C24A65B818
```

b. Compare this value with the public key hash on the `enigma` system.

You can read the public key hash over the telephone.

```
enigma # ikecert certdb -l
Certificate Slot Name: 4   Type: rsa-md5      Private Key
    Subject Name: <C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax>
    Key Size: 1024
    Public key hash: DF3F108F6AC669C88C6BD026B0FCE3A0

Certificate Slot Name: 5   Type: rsa-md5      Public Certificate
    (Private key in certlocal slot 4)
    Subject Name: <C=US, O=PartyCompany, OU=US-Partym, CN=Partym>
    Key Size: 1024
    Public key hash: 2239A6A127F88EE0CB40F7C24A65B818
```

5 On each system, trust both certificates.

Edit the `/etc/inet/ike/config` file to recognize the certificates.

The administrator of the remote system provides the values for the `cert_trust`, `remote_addr`, and `remote_id` parameters.

a. For example, on the `partym` system, the `ike/config` file would appear similar to the following:

```
# Explicitly trust the following self-signed certs
# Use the Subject Alternate Name to identify the cert

# Verified remote address and remote ID
```

```
# Verified public key hash per telephone call from administrator
cert_trust "192.168.13.213"      Local system's certificate Subject Alt Name
cert_trust "192.168.116.16"      Remote system's certificate Subject Alt Name

## Parameters that may also show up in rules.

p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg des }
p2_pfs 5

{
 label "US-partym to JA-enigmax"
 local_id_type dn
 local_id "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"
 remote_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"

 local_addr  192.168.13.213
 remote_addr 192.168.116.16

 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg sha1 encr_alg aes}
}
```

b. On the `enigma` system, add `enigma` values for local parameters in the `ike/config` file.

For the remote parameters, use `partym` values. Ensure that the value for the `label` keyword is unique. This value must be different from the remote system's `label` value.

```
...
{
 label "JA-enigmax to US-partym"
 local_id_type dn
 local_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 local_addr  192.168.116.16
 remote_addr 192.168.13.213
...
```

Example 22–5 Verifying That a Certificate From Another Administrator is Valid

In this example, the administrators use the Subject Name to verify that the certificates are identical.

The first administrator saves the output of generating and listing the certificate to a file. Because the output of the `ikecert` command prints to standard error, the administrator redirects standard error to the file.

```
sys1# cd /
sys1# ikecert certlocal -ks -m1024 -t rsa-md5 \
```

```
-D"C=US, O=TestCo, CN=Co2Sys" 2>/tmp/for_co2sys
Certificate added to database.
sys1# ikecert certdb -l "C=US, O=TestCo, CN=Co2Sys" 2>>/tmp/for_co2sys
```

The administrator verifies the contents of the file.

```
sys1# cat /tmp/for_co2sys
Creating private key.
-----BEGIN X509 CERTIFICATE-----
MIIB7TCCAVagAwIBAgIEZkHfOTANBgkqhkiG9w0BAQQFADAxMQwwCgYDVQQGEwNV
U0ExEDAOBgNVBAoMB3Rlc3RfY28xDzANBgNVBAMTBkVuaWdtYTAeFw0wODAxMTUx
OTI1MjBaFw0xMjAxMTUxOTI1MjBaMDExDDAKBgNVBAYTA1VTQTEQMA4GA1UECgwH
dGVzdF9jbzEPMA0GA1UEAxMGRW5pZ21hMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB
iQKBgQCPxGv0rUzHMnFtkx9uwYuPiWbftmWfa9iDt6ELOEuw3zlboy2qtuRUZohz
FIbCxAJevdCY6a+pktvYy3/2nJL0WATObO5T0FKn3F0bphajinLYbyCrYhEzD9E2
gkiT2D9/ttbSiMvi9usphprEDcLAFaWgCJiHnKPBEkjC0vhA3wIDAQABoxIwEDAO
BgNVHQ8BAf8EBAMCBaAwDQYJKoZIhvcNAQEEBQADgYEAL/q6xgweylGQylqLCwzN
5PIpjfzsNPf3saTyh3VplwEOW6WTHwRQT17IO/1Oc6Jnz9Mr0ZrbHWDXq+1sx180
F8+DMW1Qv1UR/lGMq3ufDG3qedmSN6txDF8qLlPCUML0YL8m4oGdewqGb+78aPyE
Y/cJRsK1hWbYyseqcIkjj5k=
-----END X509 CERTIFICATE-----
Certificate Slot Name: 2   Key Type: rsa
        (Private key in certlocal slot 2)
        Subject Name: <C=US, O=TestCo, CN=Co2Sys>
        Key Size: 1024
        Public key hash: C46DE77EF09084CE2B7D9C70479D77FF
```

Then, the administrator sends the file in an email to the second administrator.

The second administrator places the file in a secure directory, then imports the certificate from the file.

```
sys2# cd /
sys2# ikecert certdb -a < /sec/co2sys
```

The `ikecert` command imports only the text between the `-----BEGIN` and `-----END` lines. The administrator verifies that the local certificate has the same public key hash as the public key hash in the `co2sys` file.

```
sys2# ikecert certdb -l
Certificate Slot Name: 1   Key Type: rsa
        (Private key in certlocal slot 1)
        Subject Name: <C=US, O=TestCo, CN=Co2Sys>
        Key Size: 1024
        Public key hash: C46DE77EF09084CE2B7D9C70479D77FF
```

To ensure that the first administrator sent this email, the second administrator telephones the first administrator to verify the Subject Name of the certificate.

Example 22–6 Specifying a Start Time and an End Time for a Certificate

In this example, the administrator on the `partym` system establishes dates within which the certificate is valid. The certificate is backdated by 2 1/2 days, and is valid for 4 years and 6 months from the date of creation.

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
-D "C=US, O=PartyCompany, OU=US-Partym, CN=Partym" \
-A IP=192.168.13.213 \
-S -2d12h -F +4y6m
```

The administrator on the `enigma` system establishes dates within which the certificate is valid. The certificate is backdated by 2 days and is valid until midnight of December 31, 2010.

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
-D "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax" \
-A IP=192.168.116.16 \
-S -2d -F "12/31/2010 12:00 AM"
```

How to Configure IKE With Certificates Signed by a CA

Public certificates from a Certificate Authority (CA) require negotiation with an outside organization. The certificates very easily scale to protect a large number of communicating systems.

1 Use the `ikecert certlocal -kc` command to create a certificate request.

For a description of the arguments to the command, see Step 1 in "How to Configure IKE With Self-Signed Public Key Certificates" on page 471.

```
# ikecert certlocal -kc -m keysize -t keytype \
-D dname -A altname
```

a. For example, the following command creates a certificate request on the `partym` system:

```
# ikecert certlocal -kc -m 1024 -t rsa-md5 \
> -D "C=US, O=PartyCompany\, Inc., OU=US-Partym, CN=Partym" \
> -A "DN=C=US, O=PartyCompany\, Inc., OU=US-Partym"
Creating software private keys.
  Writing private key to file /etc/inet/secret/ike.privatekeys/2.
Enabling external key providers - done.
Certificate Request:
  Proceeding with the signing operation.
  Certificate request generated successfully (.../publickeys/0)
Finished successfully.
-----BEGIN CERTIFICATE REQUEST-----
MIIByjCCATMCAQAwUzELMAkGA1UEBhMCVVMxHTAbBgNVBAoTFEV4YW1wbGVDb21w
...
lcM+tw0ThRrfuJX9t/Qa1R/KxRlMA3zckO80mO9X
-----END CERTIFICATE REQUEST-----
```

b. The following command creates a certificate request on the `enigma` system:

```
# ikecert certlocal -kc -m 1024 -t rsa-md5 \
> -D "C=JA, O=EnigmaCo\, Inc., OU=JA-Enigmax, CN=Enigmax" \
> -A "DN=C=JA, O=EnigmaCo\, Inc., OU=JA-Enigmax"
Creating software private keys.
...
Finished successfully.
-----BEGIN CERTIFICATE REQUEST-----
MIIBuDCCASECAQAwSTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDFBhcnR5Q29tcGFu
...
8qlqdjaStLGfhDOO
-----END CERTIFICATE REQUEST-----
```

2 Submit the certificate request to a PKI organization.

The PKI organization can tell you how to submit the certificate request. Most organizations have a web site with a submission form. The form requires proof that the submission is legitimate. Typically, you paste your certificate request into the form. When your request has been checked by the organization, the organization issues you the following two certificate objects and a list of revoked certificates:

- Your public key certificate – This certificate is based on the request that you submitted to the organization. The request that you submitted is part of this public key certificate. The certificate uniquely identifies you.
- A Certificate Authority – The organization's signature. The CA verifies that your public key certificate is legitimate.
- A Certificate Revocation List (CRL) – The latest list of certificates that the organization has revoked. The CRL is not sent separately as a certificate object if access to the CRL is embedded in the public key certificate.

  When a URI for the CRL is embedded in the public key certificate, IKE can automatically retrieve the CRL for you. Similarly, when a DN (directory name on an LDAP server) entry is embedded in the public key certificate, IKE can retrieve and cache the CRL from an LDAP server that you specify.

  See "How to Handle a Certificate Revocation List" on page 485 for an example of an embedded URI and an embedded DN entry in a public key certificate.

3 Add each certificate to your system.

The `-a` option to the `ikecert certdb -a` adds the pasted object to the appropriate certificate database on your system. For more information, see "IKE With Public Key Certificates" on page 454.

a. Add the public key certificate that you received from the PKI organization.

```
# ikecert certdb -a
    Press the Return key
    Paste the certificate:
-----BEGIN X509 CERTIFICATE-----
```

```
...
-----END X509 CERTIFICATE----
```
Press the Return key
```
<Control>-D
```

b. Add the CA from the PKI organization.

```
# ikecert certdb -a
```
Press the Return key

Paste the CA:
```
-----BEGIN X509 CERTIFICATE-----
...
-----END X509 CERTIFICATE----
```
Press the Return key
```
<Control>-D
```

c. If the PKI organization has sent a list of revoked certificates, add the CRL to the `certrldb` database:

```
# ikecert certrldb -a
```
Press the Return key

Paste the CRL:
```
-----BEGIN CRL-----
...
-----END CRL----
```
Press the Return key
```
<Control>-D
```

4 Use the `cert_root` keyword to identify the PKI organization in the `/etc/inet/ike/config` file.

Use the name that the PKI organization provides.

a. For example, the `ike/config` file on the `partym` system might appear similar to the following:

```
# Trusted root cert
# This certificate is from Example PKI
# This is the X.509 distinguished name for the CA that it issues.

cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"

## Parameters that may also show up in rules.

p1_xform
 { auth_method rsa_sig oakley_group 1 auth_alg sha1 encr_alg des }
p2_pfs 2

{
 label "US-partym to JA-enigmax - Example PKI"
 local_id_type dn
 local_id  "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"
 remote_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
```

```
 local_addr  192.168.13.213
 remote_addr 192.168.116.16

 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg sha1 encr_alg aes}
}
```

Note – All arguments to the `auth_method` parameter must be on the same line.

b. On the `enigma` system, create a similar file.

Specifically, the `enigma` `ike/config` file should do the following:

- Include the same `cert_root` value.
- Use `enigma` values for local parameters.
- Use `partym` values for remote parameters.
- Create a unique value for the `label` keyword. This value must be different from the remote system's `label` value.

```
...
cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"
...
{
 label "JA-enigmax to US-partym - Example PKI"
 local_id_type dn
 local_id   "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id  "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 local_addr  192.168.116.16
 remote_addr 192.168.13.213
...
```

5 Tell IKE how to handle CRLs.

Choose the appropriate option:

- **No CRL available**

  If the PKI organization does not provide a CRL, add the keyword `ignore_crls` to the `ike/config` file.

  ```
  # Trusted root cert
  ...
  cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example,...
  ignore_crls
  ...
  ```

  The `ignore_crls` keyword tells IKE not to search for CRLs.

- **CRL available**

  If the PKI organization provides a central distribution point for CRLs, you can modify the `ike/config` file to point to that location.

  See "How to Handle a Certificate Revocation List" on page 485 for examples.

Example 22–7 Using `rsa_encrypt` When Configuring IKE

When you use `auth_method rsa_encrypt` in the `ike/config` file, you must add the peer's certificate to the `publickeys` database.

1. Send the certificate to the remote system's administrator.

   You can paste the certificate into an email.

   For example, the `partym` administrator would send the following email:

   ```
   To: admin@ja.enigmaexample.com
   From: admin@us.partyexample.com
   Message: -----BEGIN X509 CERTIFICATE-----
   MII...
   ----END X509 CERTIFICATE-----
   ```

   The `enigma` administrator would send the following email:

   ```
   To: admin@us.partyexample.com
   From: admin@ja.enigmaexample.com
   Message: -----BEGIN X509 CERTIFICATE-----
   MII
   ...
   -----END X509 CERTIFICATE-----
   ```

2. On each system, add the emailed certificate to the local `publickeys` database.

   ```
   # ikecert certdb -a
       Press the Return key
   -----BEGIN X509 CERTIFICATE-----
   MII...
   -----END X509 CERTIFICATE-----
       Press the Return key
   <Control>-D
   ```

The authentication method for RSA encryption hides identities in IKE from eavesdroppers. Because the `rsa_encrypt` method hides the peer's identity, IKE cannot retrieve the peer's certificate. As a result, the `rsa_encrypt` method requires that the IKE peers know each other's public keys.

Therefore, when you use an `auth_method` of `rsa_encrypt` in the `/etc/inet/ike/config` file, you must add the peer's certificate to the `publickeys` database. The `publickeys` database then holds three certificates for each communicating pair of systems:

- Your public key certificate
- The CA certificate
- The peer's public key certificate

Troubleshooting – The IKE payload, which includes the three certificates, can become too large for `rsa_encrypt` to encrypt. Errors such as "authorization failed" and "malformed payload" can indicate that the `rsa_encrypt` method cannot encrypt the total payload. Reduce the size of the payload by using a method, such as `rsa_sig`, that requires only two certificates.

How to Generate and Store Public Key Certificates on Hardware

Generating and storing public key certificates on hardware is similar to generating and storing public key certificates on your system. On hardware, the `ikecert certlocal` and `ikecert certdb` commands must identify the hardware. The `-T` option with the token ID identifies the hardware to the commands.

Before You Begin

- The hardware must be configured.
- The hardware uses the `/usr/lib/libpkcs11.so` library, unless the `pkcs11_path` keyword in the `/etc/inet/ike/config` file points to a different library. The library must be implemented according to the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki), that is, a PKCS #11 library.

  See "How to Configure IKE to Find the Sun Crypto Accelerator 4000 Board" on page 496 for setup instructions.

1 Generate a self-signed certificate or a certificate request, and specify the token ID.

Choose one of the following options:

Note – The Sun Crypto Accelerator 4000 board supports keys up to 2048 bits for RSA. For DSA, this board supports keys up to 1024 bits.

- **For a self-signed certificate, use this syntax.**

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
> -D "C=US, O=PartyCompany, OU=US-Partym, CN=Partym" \
> -a -T dca0-accel-stor IP=192.168.116.16
Creating hardware private keys.
Enter PIN for PKCS#11 token:      Type user:password
```

The argument to the `-T` option is the token ID from the attached Sun Crypto Accelerator 4000 board.

- **For a certificate request, use this syntax.**

```
# ikecert certlocal -kc -m 1024 -t rsa-md5 \
> -D "C=US, O=PartyCompany, OU=US-Partym, CN=Partym" \
> -a -T dca0-accel-stor IP=192.168.116.16
Creating hardware private keys.
Enter PIN for PKCS#11 token:      Type user:password
```

For a description of the arguments to the `ikecert` command, see the `ikecert(1M)` man page.

2 At the prompt for a PIN, type the Sun Crypto Accelerator 4000 user, a colon, and the user's password.

If the Sun Crypto Accelerator 4000 board has a user `ikemgr` whose password is `rgm4tigt`, you would type the following:

```
Enter PIN for PKCS#11 token: ikemgr:rgm4tigt
```

Note – The PIN response is stored on disk *as clear text*.

After you type the password, the certificate prints out:

```
Enter PIN for PKCS#11 token: ikemgr:rgm4tigt
-----BEGIN X509 CERTIFICATE-----
MIIBuDCCASECAQAwSTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDFBhcnR5Q29tcGFu
...
oKUDBbZ9O/pLWYGr
-----END X509 CERTIFICATE-----
```

3 Send your certificate for use by the other party.

Choose one of the following options:

- **Send the self-signed certificate to the remote system.**

  You can paste the certificate into an email.

- **Send the certificate request to an organization that handles PKI.**

  Follow the instructions of the PKI organization to submit the certificate request. For a more detailed discussion, see Step 2 of "How to Configure IKE With Certificates Signed by a CA" on page 477.

4 On your system, edit the `/etc/inet/ike/config` file to recognize the certificates.

Choose one of the following options.

- **Self-signed certificate**

  Use the values that the administrator of the remote system provides for the `cert_trust`, `remote_id`, and `remote_addr` parameters. For example, on the `enigma` system, the `ike/config` file would appear similar to the following:

```
# Explicitly trust the following self-signed certs
# Use the Subject Alternate Name to identify the cert

cert_trust "192.168.116.16"       Local system's certificate Subject Alt Name
cert_trust "192.168.13.213"       Remote system's certificate Subject Alt name


...
{
 label "JA-enigmax to US-partym"
 local_id_type dn
 local_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 local_addr  192.168.116.16
 remote_addr 192.168.13.213

 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg sha1 encr_alg aes}
}
```

- **Certificate request**

  Type the name that the PKI organization provides as the value for the `cert_root` keyword. For example, the `ike/config` file on the `enigma` system might appear similar to the following:

```
# Trusted root cert
# This certificate is from Example PKI
# This is the X.509 distinguished name for the CA that it issues.

cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"

...
{
 label "JA-enigmax to US-partym - Example PKI"
 local_id_type dn
 local_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id  "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 local_addr  192.168.116.16
 remote_addr 192.168.13.213
```

```
 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg sha1 encr_alg aes}
}
```

5 **Place the certificates from the other party in the hardware.**

Respond to the PIN request as you responded in Step 2.

Note – You *must* add the public key certificates to the same attached hardware that generated your private key.

- **Self-signed certificate.**

  Add the remote system's self-signed certificate. In this example, the certificate is stored in the file, `DCA.ACCEL.STOR.CERT`.

  ```
  # ikecert certdb -a -T dca0-accel-stor < DCA.ACCEL.STOR.CERT
  Enter PIN for PKCS#11 token:      Type user:password
  ```

  If the self-signed certificate used `rsa_encrypt` as the value for the `auth_method` parameter, add the peer's certificate to the hardware store.

- **Certificates from a PKI organization.**

  Add the certificate that the organization generated from your certificate request, and add the certificate authority (CA).

  ```
  # ikecert certdb -a -T dca0-accel-stor < DCA.ACCEL.STOR.CERT
  Enter PIN for PKCS#11 token:      Type user:password

  # ikecert certdb -a -T dca0-accel-stor < DCA.ACCEL.STOR.CA.CERT
  Enter PIN for PKCS#11 token:      Type user:password
  ```

  To add a certificate revocation list (CRL) from the PKI organization, see "How to Handle a Certificate Revocation List" on page 485.

▼ How to Handle a Certificate Revocation List

A certificate revocation list (CRL) contains outdated or compromised certificates from a Certificate Authority. You have four ways to handle CRLs.

- You must instruct IKE to ignore CRLs if your CA organization does not issue CRLs. This option is shown in Step 5 in "How to Configure IKE With Certificates Signed by a CA" on page 477.
- You can instruct IKE to access the CRLs from a URI (uniform resource indicator) whose address is embedded in the public key certificate from the CA.
- You can instruct IKE to access the CRLs from an LDAP server whose DN (directory name) entry is embedded in the public key certificate from the CA.

- You can provide the CRL as an argument to the `ikecert certrldb` command. For an example, see Example 22–8.

The following procedure describes how to instruct IKE to use CRLs from a central distribution point.

1 Display the certificate that you received from the CA.

```
# ikecert certdb -lv certspec
```

`-l` Lists certificates in the IKE certificate database.

`-v` Lists the certificates in verbose mode. Use this option with care.

certspec Is a pattern that matches a certificate in the IKE certificate database.

For example, the following certificate was issued by Sun Microsystems. Details have been altered.

```
# ikecert certdb -lv example-protect.sun.com
Certificate Slot Name: 0   Type: dsa-sha1
   (Private key in certlocal slot 0)
 Subject Name: <O=Sun Microsystems Inc, CN=example-protect.sun.com>
 Issuer Name: <CN=Sun Microsystems Inc CA (Cl B), O=Sun Microsystems Inc>
 SerialNumber: 14000D93
   Validity:
      Not Valid Before: 2002 Jul 19th, 21:11:11 GMT
      Not Valid After:  2005 Jul 18th, 21:11:11 GMT
   Public Key Info:
      Public Modulus  (n) (2048 bits): C575A...A5
      Public Exponent (e) (  24 bits): 010001
   Extensions:
      Subject Alternative Names:
              DNS = example-protect.sun.com
      Key Usage: DigitalSignature KeyEncipherment
      [CRITICAL]
   CRL Distribution Points:
      Full Name:
         URI = #Ihttp://www.sun.com/pki/pkismica.crl#i
         DN = <CN=Sun Microsystems Inc CA (Cl B), O=Sun Microsystems Inc>
      CRL Issuer:
      Authority Key ID:
      Key ID:              4F ... 6B
      SubjectKeyID:        A5 ... FD
      Certificate Policies
      Authority Information Access
```

Notice the `CRL Distribution Points` entry. The URI entry indicates that this organization's CRL is available on the web. The DN entry indicates that the CRL is available on an LDAP server. Once accessed by IKE, the CRL is cached for further use.

To access the CRL, you need to reach a distribution point.

2 **Choose one of the following methods to access the CRL from a central distribution point.**

- **Use the URI.**

  Add the keyword `use_http` to the host's `/etc/inet/ike/config` file. For example, the `ike/config` file would appear similar to the following:

  ```
  # Use CRL from organization's URI
  use_http
  ...
  ```

- **Use a web proxy.**

  Add the keyword `proxy` to the `ike/config` file. The `proxy` keyword takes a URL as an argument, as in the following:

  ```
  # Use own web proxy
  proxy "http://proxy1:8080"
  ```

- **Use an LDAP server.**

  Name the LDAP server as an argument to the `ldap-list` keyword in the host's `/etc/inet/ike/config` file. Your organization provides the name of the LDAP server. The entry in the `ike/config` file would appear similar to the following:

  ```
  # Use CRL from organization's LDAP
  ldap-list "ldap1.sun.com:389,ldap2.sun.com"
  ...
  ```

IKE retrieves the CRL and caches the CRL until the certificate expires.

Example 22–8 Pasting a CRL Into the Local `certrldb` Database

If the PKI organization's CRL is not available from a central distribution point, you can add the CRL manually to the local `certrldb` database. Follow the PKI organization's instructions for extracting the CRL into a file, then add the CRL to the database with the `ikecert certrldb -a` command.

```
# ikecert certrldb -a < Sun.Cert.CRL
```

Configuring IKE for Mobile Systems (Task Map)

The following table points to procedures to configure IKE to handle systems that log in remotely to a central site.

Task	Description	For Instructions
Communicate with a central site from off-site	Enables off-site systems to communicate with a central site. The off-site systems might be mobile.	"How to Configure IKE for Off-Site Systems" on page 488

Task	Description	For Instructions
Use a root certificate and IKE on a central system that accepts traffic from mobile systems	Configures a gateway system to accept IPsec traffic from a system that does not have a fixed IP address.	Example 22–9
Use a root certificate and IKE on a system that does not have a fixed IP address	Configures a mobile system to protect its traffic to a central site, such as company headquarters.	Example 22–10
Use self-signed certificates and IKE on a central system that accepts traffic from mobile systems	Configures a gateway system with self-signed certificates to accept IPsec traffic from a mobile system.	Example 22–11
Use self-signed certificates and IKE on a system that does not have a fixed IP address	Configures a mobile system with self-signed certificates to protect its traffic to a central site.	Example 22–12

Configuring IKE for Mobile Systems

When configured properly, home offices and mobile laptops can use IPsec and IKE to communicate with their company's central computers. A blanket IPsec policy that is combined with a public key authentication method enables off-site systems to protect their traffic to a central system.

▼ How to Configure IKE for Off-Site Systems

IPsec and IKE require a unique ID to identify source and destination. For off-site or mobile systems that do not have a unique IP address, you must use another ID type. ID types such as `DNS`, `DN`, or `email` can be used to uniquely identify a system.

Off-site or mobile systems that have unique IP addresses are still best configured with a different ID type. For example, if the systems attempt to connect to a central site from behind a NAT box, their unique addresses are not used. A NAT box assigns an arbitrary IP address, which the central system would not recognize.

Preshared keys also do not work well as an authentication mechanism for mobile systems, because preshared keys require fixed IP addresses. Self-signed certificates, or certificates from a PKI enable mobile systems to communicate with the central site.

1 Configure the central system to recognize mobile systems.

a. Set up the `/etc/hosts` file.

The central system does not have to recognize specific addresses for the mobile systems.

```
# /etc/hosts on central
central 192.xxx.xxx.x
```

b. Set up the `ipsecinit.conf` file.

The central system needs a policy that allows a wide range of IP addresses. Later, certificates in the IKE policy ensure that the connecting systems are legitimate.

```
# /etc/inet/ipsecinit.conf on central
# Keep everyone out unless they use this IPsec policy:
{} ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

c. Set up the `ike.config` file.

DNS identifies the central system. Certificates are used to authenticate the system.

```
## /etc/inet/ike/ike.config on central
# Global parameters
#
# Find CRLs by URI, URL, or LDAP
# Use CRL from organization's URI
use_http
#
# Use web proxy
proxy "http://somecache.domain:port/"
#
# Use LDAP server
ldap_server   "ldap-server1.domain.org,ldap2.domain.org:port"
#
# List CA-signed certificates
cert_root    "C=US, O=Domain Org, CN=Domain STATE"
#
# List self-signed certificates - trust server and enumerated others
#cert_trust    "DNS=central.domain.org"
#cert_trust    "DNS=mobile.domain.org"
#cert_trust    "DN=CN=Domain Org STATE (CLASS), O=Domain Org
#cert_trust    "email=root@central.domain.org"
#cert_trust    "email=user1@mobile.domain.org"
#

# Rule for mobile systems with certificate
{
  label "Mobile systems with certificate"
  local_id_type DNS

# Any mobile system who knows my DNS or IP can find me.

  local_id "central.domain.org"
  local_addr 192.xxx.xxx.x

# Root certificate ensures trust,
# so allow any remote_id and any remote IP address.
  remote_id ""
  remote_addr 0.0.0.0/0

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg sha1 }
}
```

2 Log in to each mobile system, and configure the system to find the central system.

a. Set up the `/etc/hosts` file.

The `/etc/hosts` file does not need an address for the mobile system, but can provide one. The file must contain a public IP address for the central system.

```
# /etc/hosts on mobile
mobile 10.x.x.xx
central 192.xxx.xxx.x
```

b. Set up the `ipsecinit.conf` file.

The mobile system needs to find the central system by its public IP address. The systems must configure the same IPsec policy.

```
# /etc/inet/ipsecinit.conf on mobile
# Find central
{raddr 192.xxx.xxx.x} ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

c. Set up the `ike.config` file.

The identifier cannot be an IP address. The following identifiers are valid for mobile systems:

- `DN=`*ldap-directory-name*
- `DNS=`*domain-name-server-address*
- `email=`*email-address*

Certificates are used to authenticate the mobile system.

```
## /etc/inet/ike/ike.config on mobile
# Global parameters
#
# Find CRLs by URI, URL, or LDAP
# Use CRL from organization's URI
use_http
#
# Use web proxy
proxy "http://somecache.domain:port/"
#
# Use LDAP server
ldap_server    "ldap-server1.domain.org,ldap2.domain.org:port"
#
# List CA-signed certificates
cert_root     "C=US, O=Domain  Org, CN=Domain STATE"
#
# Self-signed certificates - trust me and enumerated others
#cert_trust     "DNS=mobile.domain.org"
#cert_trust     "DNS=central.domain.org"
#cert_trust     "DN=CN=Domain Org STATE (CLASS), O=Domain Org
#cert_trust     "email=user1@domain.org"
#cert_trust     "email=root@central.domain.org"
#
# Rule for off-site systems with root certificate
{
    label "Off-site mobile with certificate"
```

```
        local_id_type DNS

    # NAT-T can translate local_addr into any public IP address
    # central knows me by my DNS

        local_id "mobile.domain.org"
        local_addr 0.0.0.0/0

    # Find central and trust the root certificate
        remote_id "central.domain.org"
        remote_addr 192.xxx.xxx.x

    p2_pfs 5

    p1_xform
    {auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg sha1 }
    }
```

3 Enable the `ike` service.

```
# svcadm enable svc:/network/ipsec/ike
```

Example 22–9 Configuring a Central Computer to Accept IPsec Traffic From a Mobile System

IKE can initiate negotiations from behind a NAT box. However, the ideal setup for IKE is without an intervening NAT box. In the following example, root certificates have been issued by a CA. The CA certificates have been placed on the mobile system and the central system. A central system accepts IPsec negotiations from a system behind a NAT box. `main1` is the company system that can accept connections from off-site systems. To set up the off-site systems, see Example 22–10.

```
## /etc/hosts on main1
main1 192.168.0.100

## /etc/inet/ipsecinit.conf on main1
# Keep everyone out unless they use this IPsec policy:
{} ipsec {encr_algs aes encr_auth_algs sha1 sa shared}

## /etc/inet/ike/ike.config on main1
# Global parameters
#
# Find CRLs by URI, URL, or LDAP
# Use CRL from organization's URI
use_http
#
# Use web proxy
proxy "http://cache1.domain.org:8080/"
#
# Use LDAP server
ldap_server   "ldap1.domain.org,ldap2.domain.org:389"
#
# List CA-signed certificate
cert_root "C=US, O=ExamplePKI Inc, OU=PKI-Example, CN=Example PKI"
#
# Rule for off-site systems with root certificate
```

```
{
  label "Off-site system with root certificate"
  local_id_type DNS
  local_id "main1.domain.org"
  local_addr 192.168.0.100

# Root certificate ensures trust,
# so allow any remote_id and any remote IP address.
  remote_id ""
  remote_addr 0.0.0.0/0

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg sha1}
p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg aes auth_alg sha1}
p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg sha1}
p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg aes auth_alg sha1}
}
```

Example 22–10 Configuring a System Behind a NAT With IPsec

In the following example, root certificates have been issued by a CA and placed on the mobile system and the central system. `mobile1` is connecting to the company headquarters from home. The Internet service provider (ISP) network uses a NAT box to enable the ISP to assign `mobile1` a private address. The NAT box then translates the private address into a public IP address that is shared with other ISP network nodes. Company headquarters is not behind a NAT. For setting up the computer at company headquarters, see Example 22–9.

```
## /etc/hosts on mobile1
mobile1 10.1.3.3
main1 192.168.0.100

## /etc/inet/ipsecinit.conf on mobile1
# Find main1
{raddr 192.168.0.100} ipsec {encr_algs aes encr_auth_algs sha1 sa shared}

## /etc/inet/ike/ike.config on mobile1
# Global parameters
#
# Find CRLs by URI, URL, or LDAP
# Use CRL from organization's URI
use_http
#
# Use web proxy
proxy "http://cache1.domain.org:8080/"
#
# Use LDAP server
ldap_server   "ldap1.domain.org,ldap2.domain.org:389"
#
# List CA-signed certificate
cert_root "C=US, O=ExamplePKI Inc, OU=PKI-Example, CN=Example PKI"
```

```
#
# Rule for off-site systems with root certificate
{
  label "Off-site mobile1 with root certificate"
  local_id_type DNS
  local_id "mobile1.domain.org"
  local_addr 0.0.0.0/0

# Find main1 and trust the root certificate
  remote_id "main1.domain.org"
  remote_addr 192.168.0.100

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg sha1 }
}
```

Example 22–11 Accepting Self-Signed Certificates From a Mobile System

In the following example, self-signed certificates have been issued and are on the mobile and the central system. `main1` is the company system that can accept connections from off-site systems. To set up the off-site systems, see Example 22–12.

```
## /etc/hosts on main1
main1 192.168.0.100

## /etc/inet/ipsecinit.conf on main1
# Keep everyone out unless they use this IPsec policy:
{} ipsec {encr_algs aes encr_auth_algs sha1 sa shared}

## /etc/inet/ike/ike.config on main1
# Global parameters
#
# Self-signed certificates - trust me and enumerated others
cert_trust    "DNS=main1.domain.org"
cert_trust    "jdoe@domain.org"
cert_trust    "user2@domain.org"
cert_trust    "user3@domain.org"
#
# Rule for off-site systems with trusted certificate
{
  label "Off-site systems with trusted certificates"
  local_id_type DNS
  local_id "main1.domain.org"
  local_addr 192.168.0.100

# Trust the self-signed certificates
# so allow any remote_id and any remote IP address.
  remote_id ""
  remote_addr 0.0.0.0/0

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg sha1 }
}
```

Example 22–12 Using Self-Signed Certificates to Contact a Central System

In the following example, `mobile1` is connecting to the company headquarters from home. The certificates have been issued and placed on the mobile and the central system. The ISP network uses a NAT box to enable the ISP to assign `mobile1` a private address. The NAT box then translates the private address into a public IP address that is shared with other ISP network nodes. Company headquarters is not behind a NAT. To set up the computer at company headquarters, see Example 22–11.

```
## /etc/hosts on mobile1
mobile1 10.1.3.3
main1 192.168.0.100

## /etc/inet/ipsecinit.conf on mobile1
# Find main1
{raddr 192.168.0.100} ipsec {encr_algs aes encr_auth_algs sha1 sa shared}

## /etc/inet/ike/ike.config on mobile1
# Global parameters

# Self-signed certificates - trust me and the central system
cert_trust    "jdoe@domain.org"
cert_trust    "DNS=main1.domain.org"
#
# Rule for off-site systems with trusted certificate
{
  label "Off-site mobile1 with trusted certificate"
  local_id_type email
  local_id "jdoe@domain.org"
  local_addr 0.0.0.0/0

# Find main1 and trust the certificate
  remote_id "main1.domain.org"
  remote_addr 192.168.0.100

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg sha1 }
}
```

Configuring IKE to Find Attached Hardware (Task Map)

The following table points to procedures that inform IKE about attached hardware. You must inform IKE about attached hardware before IKE can use the hardware. To use the hardware, follow the hardware procedures in "Configuring IKE With Public Key Certificates" on page 471.

Note – You do not have to inform IKE about on-chip hardware. For example, the UltraSPARC® T2 processor provides cryptographic acceleration. You do not need to configure IKE to find the on-chip accelerators.

Task	Description	For Instructions
Offload IKE key operations to the Sun Crypto Accelerator 1000 board	Links IKE to the PKCS #11 library.	"How to Configure IKE to Find the Sun Crypto Accelerator 1000 Board" on page 495
Offload IKE key operations and store the keys on the Sun Crypto Accelerator 4000 board	Links IKE to the PKCS #11 library and lists the name of the attached hardware.	"How to Configure IKE to Find the Sun Crypto Accelerator 4000 Board" on page 496

Configuring IKE to Find Attached Hardware

Public key certificates can also be stored on attached hardware. The Sun Crypto Accelerator 1000 board provides storage only. The Sun Crypto Accelerator 4000 and the Sun Crypto Accelerator 6000 board provide storage, and enable public key operations to be offloaded from the system to the board.

▼ How to Configure IKE to Find the Sun Crypto Accelerator 1000 Board

Before You Begin The following procedure assumes that a Sun Crypto Accelerator 1000 board is attached to the system. The procedure also assumes that the software for the board has been installed and that the software has been configured. For instructions, see the *Sun Crypto Accelerator 1000 Board Version 2.0 Installation and User's Guide*.

- **Check that the PKCS #11 library is linked.**

  Type the following command to determine whether a PKCS #11 library is linked:

```
# ikeadm get stats
Phase 1 SA counts:
Current:   initiator:          0   responder:          0
Total:     initiator:          0   responder:          0
Attempted: initiator:          0   responder:          0
Failed:    initiator:          0   responder:          0
           initiator fails include 0 time-out(s)
PKCS#11 library linked in from /usr/lib/libpkcs11.so
#
```

▼ How to Configure IKE to Find the Sun Crypto Accelerator 4000 Board

Before You Begin The following procedure assumes that a Sun Crypto Accelerator 4000 board is attached to the system. The procedure also assumes that the software for the board has been installed and that the software has been configured. For instructions, see the *Sun Crypto Accelerator 4000 Board Version 1.1 Installation and User's Guide*.

If you are using a Sun Crypto Accelerator 6000 board, see the *Sun Crypto Accelerator 6000 Board Version 1.1 User's Guide* for instructions.

1 Check that the PKCS #11 library is linked.

IKE uses the library's routines to handle key generation and key storage on the Sun Crypto Accelerator 4000 board. Type the following command to determine whether a PKCS #11 library has been linked:

```
$ ikeadm get stats
...
PKCS#11 library linked in from /usr/lib/libpkcs11.so
$
```

Note – The Sun Crypto Accelerator 4000 board supports keys up to 2048 bits for RSA. For DSA, this board supports keys up to 1024 bits.

---

2 Find the token ID for the attached Sun Crypto Accelerator 4000 board.

```
$ ikecert tokens
Available tokens with library "/usr/lib/libpkcs11.so":

"Sun Metaslot                    "
```

The library returns a token ID, also called a keystore name, of 32 characters. In this example, you could use the `Sun Metaslot` token with the `ikecert` commands to store and accelerate IKE keys.

For instructions on how to use the token, see "How to Generate and Store Public Key Certificates on Hardware" on page 482.

The trailing spaces are automatically padded by the `ikecert` command.

Example 22–13 Finding and Using Metaslot Tokens

Tokens can be stored on disk, on an attached board, or in the softtoken keystore that the Solaris encryption framework provides. The softtoken keystore token ID might resemble the following.

```
$ ikecert tokens
Available tokens with library "/usr/lib/libpkcs11.so":

"Sun Metaslot                    "
```

To create a passphrase for the softtoken keystore, see the `pktool(1)` man page.

A command that resembles the following would add a certificate to the softtoken keystore. `Sun.Metaslot.cert` is a file that contains the CA certificate.

```
# ikecert certdb -a -T "Sun Metaslot" < Sun.Metaslot.cert
Enter PIN for PKCS#11 token:      Type user:passphrase
```

Changing IKE Transmission Parameters (Task Map)

The following table points to procedures to configure transmission parameters for IKE.

Task	Description	For Instructions
Make key negotiation more efficient	Changes the key negotiation parameters.	"How to Change the Duration of Phase 1 IKE Key Negotiation" on page 498
Configure key negotiation to allow for delays in transmission	Lengthens the key negotiation parameters.	Example 22–14
Configure key negotiation to succeed quickly, or to show failures quickly	Shortens the key negotiation parameters.	Example 22–15

Changing IKE Transmission Parameters

When IKE negotiates keys, the speed of transmission can affect the success of the negotiation. Normally, you would not need to change the default values for IKE transmission parameters. However, when optimizing key negotiation over very dirty lines, or when reproducing a problem, you might want to change the transmission values.

Longer duration times enable IKE to negotiate keys over unreliable transmission lines. You can lengthen certain parameters so that initial attempts succeed. If the initial attempt does not succeed, you can space subsequent attempts to offer more time for success.

Shorter duration times enable you to take advantage of reliable transmission lines. You can more quickly retry a failed negotiation to speed up the negotiation. When diagnosing a problem, you might also want to speed up the negotiation for a quick failure. Shorter durations also enable the Phase 1 SAs to be used for their lifetime.

▼ How to Change the Duration of Phase 1 IKE Key Negotiation

1 Change the default values of the global transmission parameters on each system.

On each system, modify Phase 1 duration parameters the `/etc/inet/ike/config` file.

```
### ike/config file on     system

## Global parameters
#
## Phase 1 transform defaults
#
#expire_timer      300
#retry_limit         5
#retry_timer_init    0.5 (integer or float)
#retry_timer_max    30   (integer or float)
```

`expire_timer` — The number of seconds to let a not-yet-complete IKE Phase I negotiation linger before deleting the negotiation attempt. By default, the attempt lingers for 30 seconds.

`retry_limit` — The number of retransmits before any IKE negotiation is aborted. By default, IKE tries five times.

`retry_timer_init` — The initial interval between retransmits. This interval is doubled until the `retry_timer_max` value is reached. The initial interval is 0.5 seconds.

`retry_timer_max` — The maximum interval in seconds between retransmits. The retransmit interval stops growing at this limit. By default, the limit is 30 seconds.

2 Refresh the `ike` service.

```
# svcadm refresh svc:/network/ipsec/ike
```

Example 22–14 Lengthening IKE Phase 1 Negotiation Times

In the following example, a system is connected to its IKE peers by a high-traffic transmission line. The original settings are in comments in the file. The new settings lengthen the negotiation time.

```
### ike/config file on partym
## Global Parameters
#
## Phase 1 transform defaults
#expire_timer   300
#retry_limit      5
#retry_timer_init 0.5 (integer or float)
#retry_timer_max 30   (integer or float)
```

```
#
expire_timer  600
retry_limit  10
retry_timer_init  2.5
retry_timer_max  180
```

Example 22–15 Shortening IKE Phase 1 Negotiation Times

In the following example, a system is connected to its IKE peers by a high-speed line with little traffic. The original settings are in comments in the file. The new settings shorten the negotiation time.

```
### ike/config file on partym
## Global Parameters
#
## Phase 1 transform defaults
#expire_timer   300
#retry_limit      5
#retry_timer_init 0.5 (integer or float)
#retry_timer_max 30   (integer or float)
#
expire_timer  120
retry_timer_init  0.20
```

◆ ◆ ◆ CHAPTER 23

Internet Key Exchange (Reference)

This chapter contains the following reference information about IKE:

For instructions on implementing IKE, see Chapter 22, "Configuring IKE (Tasks)." For overview information, see Chapter 21, "Internet Key Exchange (Overview)."

IKE Service Management Facility

`svc:/network/ipsec/ike:default` **service** – The service management facility (SMF) provides the `ike` service to manage IKE. By default, this service is disabled. Before enabling this service, you must create an IKE configuration file, `/etc/inet/ike/config`.

The following `ike` service properties are configurable:

- `config_file` **property** – Is the location of the IKE configuration file. The initial value is `/etc/inet/ike/config`.
- `debug level` **property** – Is the debugging level of the `in.iked` daemon. The initial value is `op`, or operational. For possible values, see the table on debug levels under *Object Types* in the `ikeadm(1M)` man page.
- `admin_privilege` **property** – Is the level of privilege of the `in.iked` daemon. The initial value is `base`. Other values are `modkeys` and `keymat`. For details, see "IKE Administration Command" on page 503.

For information about SMF, see Chapter 11, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*. Also see the `smf(5)`, `svcadm(1M)`, and `svccfg(1M)` man pages.

IKE Daemon

The `in.iked` daemon automates the management of cryptographic keys for IPsec on a Solaris system. The daemon negotiates with a remote system that is running the same protocol to provide authenticated keying materials for security associations (SAs) in a protected manner. The daemon must be running on all systems that plan to communicate securely.

By default, the `svc:/network/ipsec/ike:default` service is not enabled. After you have configured the `/etc/inet/ike/config` file and enabled the `ike` service, the `in.iked` daemon runs at system boot.

When the IKE daemon runs, the system authenticates itself to its peer IKE entity in the Phase 1 exchange. The peer is defined in the IKE policy file, as are the authentication methods. The daemon then establishes the keys for the Phase 2 exchange. At an interval specified in the policy file, the IKE keys are refreshed automatically. The `in.iked` daemon listens for incoming IKE requests from the network and for requests for outbound traffic through the `PF_KEY` socket. For more information, see the `pf_key(7P)` man page.

Two commands support the IKE daemon. The `ikeadm` command can be used to view and temporarily modify the IKE policy. To permanently modify the IKE policy, you modify properties of the `ike` service. For the procedure, see "How to View IKE Preshared Keys" on page 465. The `ikeadm` command can also be used to view Phase 1 SAs, policy rules, preshared keys, available Diffie-Hellman groups, Phase 1 encryption and authentication algorithms, and the certificate cache.

The `ikecert` command enables you to view and manage the public key databases. This command manages the local databases, `ike.privatekeys` and `publickeys`. This command also manages public key operations and the storage of public keys on hardware.

IKE Policy File

The configuration file for the IKE policy, `/etc/inet/ike/config`, manages the keys for the interfaces that are being protected in the IPsec policy file, `/etc/inet/ipsecinit.conf`. The IKE policy file manages keys for IKE, and for the IPsec SAs. The IKE daemon itself requires keying material in the Phase 1 exchange.

Key management with IKE includes rules and global parameters. An IKE rule identifies the systems or networks that the keying material secures. The rule also specifies the authentication method. Global parameters include such items as the path to an attached hardware accelerator. For examples of IKE policy files, see "Configuring IKE With Preshared Keys (Task Map)" on page 459. For examples and descriptions of IKE policy entries, see the `ike.config(4)` man page.

The IPsec SAs that IKE supports protect the IP datagrams according to policies that are set up in the configuration file for the IPsec policy, `/etc/inet/ipsecinit.conf`. The IKE policy file determines if perfect forward security (PFS) is used when creating the IPsec SAs.

The `ike/config` file can include the path to a library that is implemented according to the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki). IKE uses this PKCS #11 library to access hardware for key acceleration and key storage.

The security considerations for the `ike/config` file are similar to the considerations for the `ipsecinit.conf` file. For details, see "Security Considerations for `ipsecinit.conf` and `ipsecconf`" on page 447.

IKE Administration Command

You can use the `ikeadm` command to do the following:

- View aspects of the IKE state.
- Change the properties of the IKE daemon.
- Display statistics on SA creation during the Phase 1 exchange.
- Debug IKE protocol exchanges.
- Display IKE daemon objects, such as all Phase 1 SAs, policy rules, preshared keys, available Diffie-Hellman groups, Phase 1 encryption and authentication algorithms, and the certificate cache.

For examples and a full description of this command's options, see the `ikeadm(1M)` man page.

The privilege level of the running IKE daemon determines which aspects of the IKE daemon can be viewed and modified. Three levels of privilege are possible.

`base` level	You cannot view or modify keying material. The `base` level is the default level of privilege.
`modkeys` level	You can remove, change, and add preshared keys.
`keymat` level	You can view the actual keying material with the `ikeadm` command.

For a temporary privilege change, you can use the `ikeadm` command. For a permanent change, change the `admin_privilege` property of the `ike` service. For the procedure, see "How to Manage IKE and IPsec Services" on page 423.

The security considerations for the `ikeadm` command are similar to the considerations for the `ipseckey` command. For details, see "Security Considerations for `ipseckey`" on page 449.

IKE Preshared Keys Files

When you create preshared keys manually, the keys are stored in files in the `/etc/inet/secret` directory. The `ike.preshared` file contains the preshared keys for Internet Security Association and Key Management Protocol (ISAKMP) SAs. The `ipseckeys` file contains the preshared keys for IPsec SAs. The files are protected at `0600`. The `secret` directory is protected at `0700`.

- You create an `ike.preshared` file when you configure the `ike/config` file to require preshared keys. You enter keying material for ISAKMP SAs, that is, for IKE authentication, in the `ike.preshared` file. Because the preshared keys are used to authenticate the Phase 1 exchange, the file must be valid before the `in.iked` daemon starts.
- The `ipseckeys` file contains keying material for IPsec SAs. For examples of manually managing the file, see "How to Manually Create IPsec Security Associations" on page 417. The IKE daemon does not use this file. The keying material that IKE generates for IPsec SAs is stored in the kernel.

Note – Preshared keys cannot take advantage of hardware storage. Preshared keys are generated and are stored on the system.

IKE Public Key Databases and Commands

The `ikecert` command manipulates the local system's public key databases. You use this command when the `ike/config` file requires public key certificates. Because IKE uses these databases to authenticate the Phase 1 exchange, the databases must be populated before activating the `in.iked` daemon. Three subcommands handle each of the three databases: `certlocal`, `certdb`, and `certrldb`.

The `ikecert` command also handles key storage. Keys can be stored on disk, on an attached Sun Crypto Accelerator 6000 or Sun Crypto Accelerator 4000 board, or in a softtoken keystore. The softtoken keystore is available when the metaslot in the Solaris cryptographic framework is used to communicate with the hardware device. The `ikecert` command uses the PKCS #11 library to locate key storage.

For more information, see the `ikecert(1M)` man page. For information about metaslot and the softtoken keystore, see the `cryptoadm(1M)` man page.

`ikecert tokens` Command

The `tokens` argument lists the token IDs that are available. Token IDs enable the `ikecert certlocal` and `ikecert certdb` commands to generate public key certificates and certificate requests. The certificates and certificate requests can also be stored by the cryptographic

framework in the softtoken keystore, or on an attached Sun Crypto Accelerator 6000 or Sun Crypto Accelerator 4000 board. The `ikecert` command uses the PKCS #11 library to locate certificate storage.

`ikecert certlocal` Command

The `certlocal` subcommand manages the private key database. Options to this subcommand enable you to add, view, and remove private keys. This subcommand also creates either a self-signed certificate or a certificate request. The `-ks` option creates a self-signed certificate. The `-kc` option creates a certificate request. Keys are stored on the system in the `/etc/inet/secret/ike.privatekeys` directory, or on attached hardware with the `-T` option.

When you create a private key, the options to the `ikecert certlocal` command must have related entries in the `ike/config` file. The correspondences between `ikecert` options and `ike/config` entries are shown in the following table.

TABLE 23–1 Correspondences Between `ikecert` Options and `ike/config` Entries

`ikecert` Option	`ike/config` Entry	Description
`-A` *subject-alternate-name*	`cert_trust` *subject-alternate-name*	A nickname that uniquely identifies the certificate. Possible values are an IP address, an email address, or a domain name.
`-D` *X.509-distinguished-name*	*X.509-distinguished-name*	The full name of the certificate authority that includes the country (C), organization name (ON), organizational unit (OU), and common name (CN).
`-t dsa-sha1`	`auth_method dss_sig`	An authentication method that is slightly slower than RSA.
`-t rsa-md5` and `-t rsa-sha1`	`auth_method rsa_sig`	An authentication method that is slightly faster than DSA. The RSA public key must be large enough to encrypt the biggest payload. Typically, an identity payload, such as the X.509 distinguished name, is the biggest payload.
`-t rsa-md5` and `-t rsa-sha1`	`auth_method rsa_encrypt`	RSA encryption hides identities in IKE from eavesdroppers, but requires that the IKE peers know each other's public keys.
`-T`	`pkcs11_path`	The PKCS #11 library handles key acceleration on the Sun Crypto Accelerator 1000 board, the Sun Crypto Accelerator 6000 board, and the Sun Crypto Accelerator 4000 board. The library also provides the tokens that handle key storage on the Sun Crypto Accelerator 6000 and Sun Crypto Accelerator 4000 boards.

If you issue a certificate request with the `ikecert certlocal -kc` command, you send the output of the command to a PKI organization or to a certificate authority (CA). If your

company runs its own PKI, you send the output to your PKI administrator. The PKI organization, the CA, or your PKI administrator then creates certificates. The certificates that the PKI or CA returns to you are input to the `certdb` subcommand. The certificate revocation list (CRL) that the PKI returns to you is input for the `certrldb` subcommand.

`ikecert certdb` Command

The `certdb` subcommand manages the public key database. Options to this subcommand enable you to add, view, and remove certificates and public keys. The command accepts, as input, certificates that were generated by the `ikecert certlocal -ks` command on a remote system. For the procedure, see "How to Configure IKE With Self-Signed Public Key Certificates" on page 471. This command also accepts the certificate that you receive from a PKI or CA as input. For the procedure, see "How to Configure IKE With Certificates Signed by a CA" on page 477.

The certificates and public keys are stored on the system in the `/etc/inet/ike/publickeys` directory. The `-T` option stores the certificates, private keys, and public keys on attached hardware.

`ikecert certrldb` Command

The `certrldb` subcommand manages the certificate revocation list (CRL) database, `/etc/inet/ike/crls`. The CRL database maintains the revocation lists for public keys. Certificates that are no longer valid are on this list. When PKIs provide you with a CRL, you can install the CRL in the CRL database with the `ikecert certrldb` command. For the procedure, see "How to Handle a Certificate Revocation List" on page 485.

`/etc/inet/ike/publickeys` Directory

The `/etc/inet/ike/publickeys` directory contains the public part of a public-private key pair and its certificate in files, or *slots*. The directory is protected at `0755`. The `ikecert certdb` command populates the directory. The `-T` option stores the keys on the Sun Crypto Accelerator 6000 or Sun Crypto Accelerator 4000 board rather than in the `publickeys` directory.

The slots contain, in encoded form, the X.509 distinguished name of a certificate that was generated on another system. If you are using self-signed certificates, you use the certificate that you receive from the administrator of the remote system as input to the command. If you are using certificates from a CA, you install two signed certificates from the CA into this database. You install a certificate that is based on the certificate signing request that you sent to the CA. You also install a certificate of the CA.

/etc/inet/secret/ike.privatekeys Directory

The /etc/inet/secret/ike.privatekeys directory holds private key files that are part of a public-private key pair, which is keying material for ISAKMP SAs. The directory is protected at 0700. The ikecert certlocal command populates the ike.privatekeys directory. Private keys are not effective until their public key counterparts, self-signed certificates or CAs, are installed. The public key counterparts are stored in the /etc/inet/ike/publickeys directory or on a Sun Crypto Accelerator 6000 or Sun Crypto Accelerator 4000 board.

/etc/inet/ike/crls Directory

The /etc/inet/ike/crls directory contains certificate revocation list (CRL) files. Each file corresponds to a public certificate file in the /etc/inet/ike/publickeys directory. PKI organizations provide the CRLs for their certificates. You can use the ikecert certrldb command to populate the database.

◆ ◆ ◆ CHAPTER 24

IP Filter in Oracle Solaris (Overview)

This chapter provides an overview of IP Filter, an Oracle Solaris feature. For IP Filter tasks, see Chapter 25, " IP Filter (Tasks)."

This chapter contains the following information:

Introduction to Filter

The IP Filter feature of Oracle Solaris replaces the SunScreen firewall in the OS. Like the SunScreen firewall, IP Filter provides stateful packet filtering and network address translation (NAT). IP Filter also includes stateless packet filtering and the ability to create and manage address pools.

Packet filtering provides basic protection against network-based attacks. IP Filter can filter by IP address, port, protocol, network interface, and traffic direction. IP Filter can also filter by an individual source IP address, a destination IP address, by a range of IP addresses, or by address pools.

IP Filter is derived from open source IPFilter software. To view license terms, attribution, and copyright statements for open source IPFilter, the default path is `/usr/lib/ipf/IPFILTER.LICENCE`. If Oracle Solaris has been installed anywhere other than the default, modify the given path to access the file at the installed location.

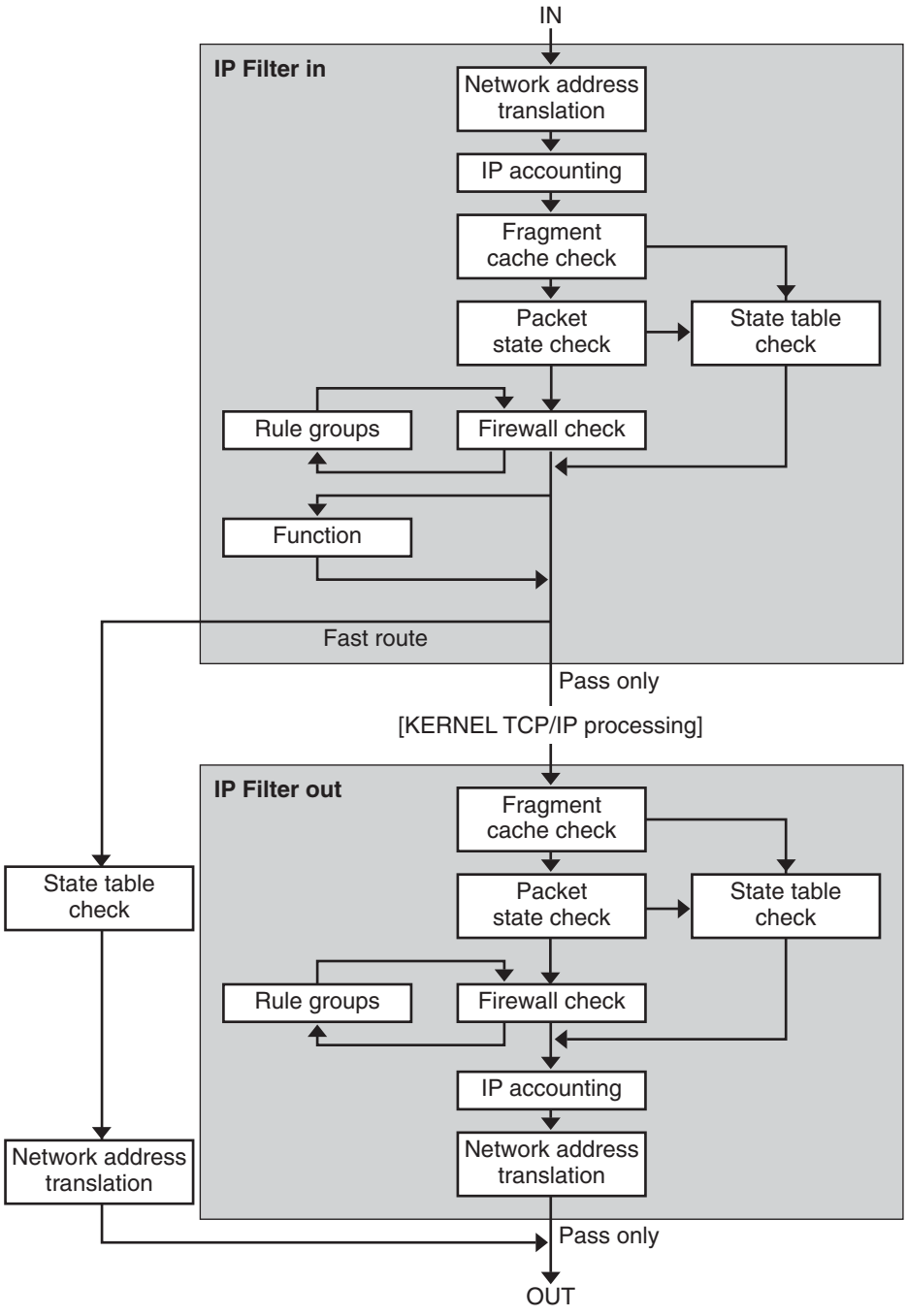
Information Sources for Open Source IPFilter

The home page for the open source IPFilter software by Darren Reed is found at `http://coombs.anu.edu.au/~avalon/ip-filter.html`. This site includes information for open source IPFilter, including a link to a tutorial entitled "IP Filter Based Firewalls HOWTO" (Brendan Conoboy and Erik Fichtner, 2002). This tutorial provides step-by-step instructions for building firewalls in a BSD UNIX environment. Although written for a BSD UNIX environment, the tutorial is also relevant for the configuration of the IP Filter feature of Oracle Solaris.

IP Filter Packet Processing

IP Filter executes a sequence of steps as a packet is processed. The following diagram illustrates the steps of packet processing and how filtering integrates with the TCP/IP protocol stack.

FIGURE 24–1 Packet Processing Sequence



The packet processing sequence includes the following:

- **Network Address Translation (NAT)**

  The translation of a private IP address to a different public address, or the aliasing of multiple private addresses to a single public one. NAT allows an organization to resolve the problem of IP address depletion when the organization has existing networks and needs to access the Internet.

- **IP Accounting**

  Input and output rules can be separately set up, recording the number of bytes that pass through. Each time a rule match occurs, the byte count of the packet is added to the rule and allows for collection of cascading statistics.

- **Fragment Cache Check**

  If the next packet in the current traffic is a fragment and the previous packet was allowed, the packet fragment is also allowed, bypassing state table and rule checking.

- **Packet State Check**

  If `keep state` is included in a rule, all packets in a specified session are passed or blocked automatically, depending on whether the rule says `pass` or `block`.

- **Firewall Check**

  Input and output rules can be separately set up, determining whether or not a packet will be allowed through IP Filter, into the kernel's TCP/IP routines, or out onto the network.

- **Groups**

  Groups allow you to write your rule set in a tree fashion.

- **Function**

  A function is the action to be taken. Possible functions include `block`, `pass`, `literal`, and `send ICMP response`.

- **Fast-route**

  Fast-route signals IP Filter to not pass the packet into the UNIX IP stack for routing, which results in a TTL decrement.

- **IP Authentication**

  Packets that are authenticated are only passed through the firewall loops once to prevent double-processing.

Guidelines for Using IP Filter

- IP Filter is managed by the SMF services svc:/network/pfil and svc:/network/ipfilter. For a complete overview of SMF, see Chapter 11, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*. For information on the step-by-step procedures that are associated with SMF, see Chapter 12, "Managing Services (Tasks)," in *System Administration Guide: Basic Administration*.
- IP Filter requires direct editing of configuration files.
- IP Filter is installed as part of Oracle Solaris. By default, IP Filter is not activated after a fresh install. To configure filtering, you must edit configuration files and manually activate IP Filter. You can activate filtering by either rebooting the system or by plumbing the interfaces using the `ipadm` command. For more information, see the `ipadm`(1M) man page. For the tasks associated with enabling IP Filter, see "Configuring IP Filter" on page 523.
- To administer IP Filter, you must be able to assume a role that includes the IP Filter Management rights profile, or become superuser. You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.
- IP Network Multipathing (IPMP) supports stateless filtering only.
- Sun Cluster configurations do not support filtering with IP Filter.
- Filtering between zones is not currently supported with IP Filter.

Using IP Filter Configuration Files

IP Filter can be used to provide firewall services or network address translation (NAT). IP Filter can be implemented using loadable configuration files. IP Filter includes a directory called `/etc/ipf`. You can create and store configuration files called `ipf.conf`, `ipnat.conf` and `ippool.conf` in the `/etc/ipf` directory. These files are loaded automatically during the boot process when they reside in the `/etc/ipf` directory. You can also store the configuration files in another location and load the files manually. For example configuration files, see "Creating and Editing IP Filter Configuration Files" on page 546.

Working With IP Filter Rule Sets

To manage your firewall, you use IP Filter to specify rule sets that you use to filter your network traffic. You can create the following types of rule sets:

- Packet filtering rule sets
- Network Address Translation (NAT) rule sets

Additionally, you can create address pools to reference groups of IP addresses. You can then use these pools later in a rule set. The address pools help to speed up rule processing. Address pools also make managing large groups of addresses easier.

Using IP Filter's Packet Filtering Feature

You set up packet filtering by using packet filtering rule sets. Use the `ipf` command to work with packet filtering rule sets. For more information on the `ipf` command, see the `ipf(1M)` command.

You can create packet filtering rules either at the command line, using the `ipf` command, or in a packet filtering configuration file. If you want the packet filtering rules to be loaded at boot time, create a configuration file called `/etc/ipf/ipf.conf` in which to put packet filtering rules. If you do not want the packet filtering rules loaded at boot time, put the `ipf.conf` file in a location of your choice, and manually activate packet filtering by using the `ipf` command.

You can maintain two sets of packet filtering rule sets with IP Filter, the active rule set and the inactive rule set. In most cases, you work with the active rule set. However, the `ipf -I` command enables you to apply the command action to the inactive rule list. The inactive rule list is not used by IP Filter unless you select it. The inactive rule list provides you with a place to store rules without affecting active packet filtering.

IP Filter processes the rules in the rules list from the beginning of the configured rules list to the end of the rules list before passing or blocking a packet. IP Filter maintains a flag that determines whether it will or will not pass a packet. It goes through the entire rule set and determines whether to pass or block the packet based on the last matching rule.

There are two exceptions to this process. The first exception is if the packet matches a rule containing the `quick` keyword. If a rule includes the `quick` keyword, the action for that rule is taken, and no subsequent rules are checked. The second exception is if the packet matches a rule containing the `group` keyword. If a packet matches a group, only rules tagged with the group are checked.

Configuring Packet Filtering Rules

Use the following syntax to create packet filtering rules:

action `[in|out]` *option keyword, keyword...*

1. Each rule begins with an action. IP Filter applies the action to the packet if the packet matches the rule. The following list includes the commonly used actions applied to a packet.

   `block` Prevents the packet from passing through the filter.

   `pass` Allows the packet through the filter.

   `log` Logs the packet but does not determine if the packet is blocked or passed. Use the `ipmon` command to view the log.

`count` Includes the packet in the filter statistics. Use the `ipfstat` command to view the statistics.

`skip` *number* Makes the filter skip over *number* filtering rules.

`auth` Requests that packet authentication be performed by a user program that validates packet information. The program determines whether the packet is passed or blocked.

2. Following the action, the next word must be either `in` or `out`. Your choice determines whether the packet filtering rule is applied to an incoming packet or to an outgoing packet.
3. Next, you can choose from a list of options. If you use more than one option, they must be in the order shown here.

   `log` Logs the packet if the rule is the last matching rule. Use the `ipmon` command to view the log.

   `quick` Executes the rule containing the `quick` option if there is a packet match. All further rule checking stops.

   `on` *interface-name* Applies the rule only if the packet is moving in or out of the specified interface.

   `dup-to` *interface-name* Copies the packet and sends the duplicate out on *interface-name* to an optionally specified IP address.

   `to` *interface-name* Moves the packet to an outbound queue on *interface-name*.
4. After specifying the options, you can choose from a variety of keywords that determine whether the packet matches the rule. The following keywords must be used in the order shown here.

Note – By default, any packet that does not match any rule in the configuration file is passed through the filter.

`tos` Filters the packet based on the type-of-service value expressed as either a hexadecimal or a decimal integer.

`ttl` Matches the packet based on its time-to-live value. The time-to-live value stored in a packet indicates the length of time a packet can be on the network before being discarded.

`proto` Matches a specific protocol. You can use any of the protocol names specified in the `/etc/protocols` file, or use a decimal number to represent the protocol. The keyword `tcp/udp` can be used to match either a TCP or a UDP packet.

`from/to/all/any`	Matches any or all of the following: the source IP address, the destination IP address, and the port number. The `all` keyword is used to accept packets from all sources and to all destinations.
`with`	Matches specified attributes associated with the packet. Insert either the word `not` or the word `no` in front of the keyword in order to match the packet only if the option is not present.
`flags`	Used for TCP to filter based on TCP flags that are set. For more information on the TCP flags, see the `ipf(4)` man page.
`icmp-type`	Filters according to ICMP type. This keyword is used only when the `proto` option is set to `icmp` and is not used if the `flags` option is used.
`keep` *keep-options*	Determines the information that is kept for a packet. The *keep-options* available include the `state` option and the `frags` option. The `state` option keeps information about the session and can be kept on TCP, UDP, and ICMP packets. The `frags` option keeps information on packet fragments and applies the information to later fragments. The *keep-options* allow matching packets to pass without going through the access control list.
`head` *number*	Creates a new group for filtering rules, which is denoted by the number *number*.
`group` *number*	Adds the rule to group number *number* instead of the default group. All filtering rules are placed in group 0 if no other group is specified.

The following example illustrates how to put together the packet filtering rule syntax to create a rule. To block incoming traffic from the IP address `192.168.0.0/16`, you would include the following rule in the rule list:

```
block in quick from 192.168.0.0/16 to any
```

For the complete grammar and syntax used to write packet filtering rules, see the `ipf(4)` man page. For tasks associated with packet filtering, see "Managing Packet Filtering Rule Sets for IP Filter" on page 530. For an explanation of the IP address scheme (`192.168.0.0/16`) shown in the example, see Chapter 1, "Planning an IPv4 Addressing Scheme (Tasks)."

Using IP Filter's NAT Feature

NAT sets up mapping rules that translate source and destination IP addresses into other Internet or intranet addresses. These rules modify the source and destination addresses of incoming or outgoing IP packets and send the packets on. You can also use NAT to redirect traffic from one port to another port. NAT maintains the integrity of the packet during any modification or redirection done on the packet.

Use the `ipnat` command to work with NAT rule lists. For more information on the `ipnat` command, see the `ipnat(1M)` command.

You can create NAT rules either at the command line, using the `ipnat` command, or in a NAT configuration file. NAT configuration rules reside in the `ipnat.conf` file. If you want the NAT rules to be loaded at boot time, create a file called `/etc/ipf/ipnat.conf` in which to put NAT rules. If you do not want the NAT rules loaded at boot time, put the `ipnat.conf` file in a location of your choice, and manually activate packet filtering with the `ipnat` command.

Configuring NAT Rules

Use the following syntax to create NAT rules:

command interface-name parameters

1. Each rule begins with one of the following commands:

`map`	Maps one IP address or network to another IP address or network in an unregulated round-robin process.
`rdr`	Redirects packets from one IP address and port pair to another IP address and port pair.
`bimap`	Establishes a bidirectional NAT between an external IP address and an internal IP address.
`map-block`	Establishes static IP address-based translation. This command is based on an algorithm that forces addresses to be translated into a destination range.

2. Following the command, the next word is the interface name, such as `bge0`.
3. Next, you can choose from a variety of parameters, which determine the NAT configuration. Some of the parameters include:

`ipmask`	Designates the network mask.
`dstipmask`	Designates the address that `ipmask` is translated to.
`mapport`	Designates `tcp`, `udp`, or `tcp/udp` protocols, along with a range of port numbers.

The following example illustrates how to put together the NAT rule syntax together to create a NAT rule. To rewrite a packet that goes out on the `de0` device with a source address of `192.168.1.0/24` and to externally show its source address as `10.1.0.0/16`, you would include the following rule in the NAT rule set:

```
map de0 192.168.1.0/24 -> 10.1.0.0/16
```

For the complete grammar and syntax used to write NAT rules, see the `ipnat(4)` man page.

Using IP Filter's Address Pools Feature

Address pools establish a single reference that is used to name a group of address/netmask pairs. Address pools provide processes to reduce the time needed to match IP addresses with rules. Address pools also make managing large groups of addresses easier.

Address pool configuration rules reside in the `ippool.conf` file. If you want the address pool rules to be loaded at boot time, create a file called `/etc/ipf/ippool.conf` in which to put address pool rules. If you do not want the address pool rules loaded at boot time, put the `ippool.conf` file in a location of your choice, and manually activate packet filtering with the `ippool` command.

Configuring Address Pools

Use the following syntax to create an address pool:

`table role =` *role-name* `type =` *storage-format* `number =` *reference-number*

`table` Defines the reference for the multiple addresses.

`role` Specifies the role of the pool in IP Filter. At this time, the only role you can reference is `ipf`.

`type` Specifies the storage format for the pool.

`number` Specifies the reference number that is used by the filtering rule.

For example, to reference the group of addresses `10.1.1.1` and `10.1.1.2`, and the network `192.16.1.0` as pool number `13`, you would include the following rule in the address pool configuration file:

```
table role = ipf type = tree number = 13
{ 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24 };
```

Then, to reference pool number `13` in a filtering rule, you would construct the rule similar to the following example:

```
pass in from pool/13 to any
```

Note that you must load the pool file before loading the rules file that contains a reference to the pool. If you do not, the pool is undefined, as shown in the following output:

```
# ipfstat -io
empty list for ipfilter(out)
block in from pool/13(!) to any
```

Even if you add the pool later, the addition of the pool does not update the kernel rule set. You also need to reload the rules file that references the pool.

For the complete grammar and syntax used to write packet filtering rules, see the `ippool(4)` man page.

Packet Filter Hooks

In the current release, packet filter hooks replace the `pfil` module to enable IP filter. In previous Solaris releases, configuration of the `pfil` module was required as an additional step to set up IP Filter. This extra configuration requirement increased the risk of errors that would cause IP Filter to work improperly. The insertion of the `pfil` STREAMS module between IP and the device driver also caused performance degradation. Lastly, the `pfil` module could not perform packet interception between zones.

The use of packet filter hooks streamlines the procedure to enable IP Filter. Through these hooks, IP Filter uses pre-routing (input) and post-routing (output) filter taps to control packet flow into and out of the Oracle Solaris system.

Packet filter hooks eliminate the need for the `pfil` module. Thus the following components that are associated with the module are also removed.

- `pfil` driver
- `pfil` daemon
- `svc:/network/pfil` SMF service

For tasks associated with enabling IP Filter, see Chapter 25, " IP Filter (Tasks)."

IPv6 for IP Filter

Beginning with the Solaris 6/06 release, support for IPv6 is available with IP Filter. IPv6 packet filtering can filter based on the source/destination IPv6 address, pools containing IPv6 addresses, and IPv6 extension headers.

IPv6 is similar to IPv4 in many ways. However, header and packet size differ between the two versions of IP, which is an important consideration for IP Filter. IPv6 packets known as *jumbograms* contain a datagram longer than 65,535 bytes. IP Filter does not support IPv6 jumbograms. To learn more about other IPv6 features, see "Major Features of IPv6" on page 49.

Note – For more information on jumbograms, refer to the document IPv6 Jumbograms, RFC 2675 from the Internet Engineering Task Force (IETF). [`http://www.ietf.org/rfc/rfc2675.txt`]

IP Filter tasks associated with IPv6 do not differ substantially from IPv4. The most notable difference is the use of the `-6` option with certain commands. Both the `ipf` command and the `ipfstat` command include the `-6` option for use with IPv6 packet filtering. Use the `-6` option with the `ipf` command to load and flush IPv6 packet filtering rules. To display IPv6 statistics, use the `-6` option with the `ipfstat` command. The `ipmon` and `ippool` commands also support IPv6, although there is no associated option for IPv6 support. The `ipmon` command has been enhanced to accommodate the logging of IPv6 packets. The `ippool` command supports the pools with IPv6 addresses. You can create pools of only IPv4 or IPv6 addresses, or a pool containing both IPv4 and IPv6 addresses within the same pool.

You can use the `ipf6.conf` file to create packet filtering rule sets for IPv6. By default, the `ipf6.conf` configuration file is included in the `/etc/ipf` directory. As with the other filtering configuration files, the `ipf6.conf` file loads automatically during the boot process when it is stored in the `/etc/ipf` directory. You can also create and store an IPv6 configuration file in another location and load the file manually.

Note – Network Address Translation (NAT) does not support IPv6.

Once packet filtering rules for IPv6 have been set up, activate IPv6 packet filtering capabilities by plumbing the `inet6` version of the interface.

For more information on IPv6, see Chapter 2, “Planning an IPv6 Addressing Scheme (Overview).” For tasks associated with IP Filter, see Chapter 25, “ IP Filter (Tasks).”

IP Filter Man Pages

The following table includes the man page documentation relevant to IP Filter.

Man Page	Description
`ipf(1M)`	Use the `ipf` command to complete the following tasks: ■ Work with packet filtering rule sets. ■ Disable and enable filtering. ■ Reset statistics and resynchronize the in-kernel interface list with the current interface status list.
`ipf(4)`	Contains the grammar and syntax for creating IP Filter packet filtering rules.
`ipfilter(5)`	Provides open source IP Filter licensing information.

Man Page	Description
ipfs(1M)	Use the ipfs command to save and restore NAT information and state table information across reboots.
ipfstat(1M)	Use the ipfstat command to retrieve and display statistics on packet processing.
ipmon(1M)	Use the ipmon command to open the log device and view logged packets for both packet filtering and NAT.
ipnat(1M)	Use the ipnat command to complete the following tasks: ■ Work with NAT rules. ■ Retrieve and display NAT statistics.
ipnat(4)	Contains the grammar and syntax for creating NAT rules.
ippool(1M)	Use the ippool command to create and manage address pools.
ippool(4)	Contains the grammar and syntax for creating IP Filter address pools.
ndd(1M)	Displays current filtering parameters of the pfil STREAMS module and the current values of the tunable parameters.

◆ ◆ ◆ CHAPTER 25

IP Filter (Tasks)

This chapter provides step-by-step instructions for tasks. For overview information about IP Filter, see Chapter 24, "IP Filter in Oracle Solaris (Overview)."

This chapter contains the following information:

Configuring IP Filter

The following task map identifies the procedures associated with configuring IP Filter.

TABLE 25–1 Configuring IP Filter (Task Map)

Task	Description	For Instructions
Initially enable IP Filter.	IP Filter is not enabled by default. You must either enable it manually or use the configuration files in the `/etc/ipf/` directory and reboot the system. Packet filter hooks replace the `pfil` module to enable IP filter.	"How to Enable IP Filter" on page 524
Re-enable IP Filter.	If IP Filter is deactivated or disabled, you can re-enable IP Filter either by rebooting the system or by using the `ipf` command.	"How to Re-Enable IP Filter" on page 525

TABLE 25–1 Configuring IP Filter (Task Map) *(Continued)*

Task	Description	For Instructions
Enable loopback filtering	As an option, you can enable loopback filtering, for example, to filter traffic between zones.	"How to Enable Loopback Filtering" on page 526

How to Enable IP Filter

1 Assume a role that includes the IP Filter Management rights profile, or become superuser.

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 Create a packet filtering rule set.

The packet filtering rule set contains packet filtering rules that are used by IP Filter. If you want the packet filtering rules to be loaded at boot time, edit the `/etc/ipf/ipf.conf` file to implement IPv4 packet filtering. Use the `/etc/ipf/ipf6.conf` file for IPv6 packet filtering rules. If you do not want the packet filtering rules loaded at boot time, put the rules in a file of your choice, and manually activate packet filtering. For information about packet filtering, see "Using IP Filter's Packet Filtering Feature" on page 514. For information about working with configuration files, see "Creating and Editing IP Filter Configuration Files" on page 546.

3 (Optional) Create a network address translation (NAT) configuration file.

Note – Network Address Translation (NAT) does not support IPv6.

Create an `ipnat.conf` file if you want to use network address translation. If you want the NAT rules to be loaded at boot time, create a file called `/etc/ipf/ipnat.conf` in which to put NAT rules. If you do not want the NAT rules loaded at boot time, put the `ipnat.conf` file in a location of your choice, and manually activate the NAT rules.

For more information about NAT, see "Using IP Filter's NAT Feature" on page 516.

4 (Optional) Create an address pool configuration file.

Create an `ipool.conf` file if you want to refer to a group of addresses as a single address pool. If you want the address pool configuration file to be loaded at boot time, create a file called `/etc/ipf/ippool.conf` in which to put the address pool. If you do not want the address pool configuration file to be loaded at boot time, put the `ippool.conf` file in a location of your choice, and manually activate the rules.

An address pool can contain only IPv4 addresses or only IPv6 addresses. It can also contain both IPv4 and IPv6 addresses.

For more information about address pools, see "Using IP Filter's Address Pools Feature" on page 518.

5 **(Optional) Enable filtering of loopback traffic.**

If you intend to filter traffic between zones that are configured in your system, you must enable loopback filtering. See "How to Enable Loopback Filtering" on page 526. Make sure that you also define the appropriate rule sets that apply to the zones.

6 **Activate IP Filter.**

```
# svcadm enable network/ipfilter
```

How to Re-Enable IP Filter

You can re-enable packet filtering after it has been temporarily disabled.

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Enable IP Filter and activate filtering using one of the following methods:**

- Reboot the machine.

  ```
  # reboot
  ```

  Note – When IP Filter is enabled, after a reboot the following files are loaded if they are present: the `/etc/ipf/ipf.conf` file, the `/etc/ipf/ipf6.conf` file when using IPv6, or the `/etc/ipf/ipnat.conf`.

- Perform the following series of commands to enable IP Filter and activate filtering:

  a. Enable IP Filter.

  ```
  # ipf -E
  ```

  b. Activate packet filtering.

  ```
  # ipf -f filename
  ```

  c. (Optional) Activate NAT.

  ```
  # ipnat -f filename
  ```

  Note – Network Address Translation (NAT) does not support IPv6.

How to Enable Loopback Filtering

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Stop IP Filter if it is running.**

   ```
   # svcadm disable network/ipfilter
   ```

3. **Edit the `/etc/ipf.conf` or `/etc/ipf6.conf` file by adding the following line at the beginning of the file:**

   ```
   set intercept_loopback true;
   ```

   This line must precede all the IP filter rules that are defined in the file. However, you can insert comments before the line, similar to the following example:

   ```
   #
   # Enable loopback filtering to filter between zones
   #
   set intercept_loopback true;
   #
   # Define policy
   #
   block in all
   block out all
   <other rules>
   ...
   ```

4. **Start the IP filter.**

   ```
   # svcadm enable network/ipfilter
   ```

5. **To verify the status of loopback filtering, use the following command:**

   ```
   # ipf –T ipf_loopback
   ipf_loopback    min 0   max 0x1 current 1
   #
   ```

   If loopback filtering is disabled, the command would generate the following output:

   ```
   ipf_loopback    min 0   max 0x1 current 0
   ```

Deactivating and Disabling IP Filter

You might want to deactivate or disable packet filtering and NAT under the following circumstances:

- For testing purposes
- To troubleshoot system problems when you think the problems are caused by IP Filter

The following task map identifies the procedures associated with deactivating or disabling IP Filter features.

TABLE 25–2 Deactivating and Disabling IP Filter (Task Map)

Task	Description	For Instructions
Deactivate packet filtering.	Deactivate packet filtering using the `ipf` command.	"How to Deactivate Packet Filtering" on page 527
Deactivate NAT.	Deactivate NAT using the `ipnat` command.	"How to Deactivate NAT" on page 528
Disable packet filtering and NAT.	Disable packet filtering and NAT using the `ipf` command.	"How to Disable Packet Filtering" on page 528

▼ How to Deactivate Packet Filtering

The following procedure deactivates IP Filter packet filtering by flushing the packet filtering rules from the active filtering rule set. The procedure does not disable IP Filter. You can reactivate IP Filter by adding rules to the rule set.

1 Assume a role that includes the IP Filter Management rights profile, or become superuser.

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 Use one of the following methods to deactivate IP Filter rules:

- Remove the active rule set from the kernel.

  ```
  # ipf -Fa
  ```

  This command deactivates all packet filtering rules.
- Remove incoming packet filtering rules.

  ```
  # ipf -Fi
  ```

  This command deactivates packet filtering rules for incoming packets.
- Remove outgoing packet filtering rules.

```
# ipf -Fo
```

This command deactivates packet filtering rules for outgoing packets.

How to Deactivate NAT

The following procedure deactivates IP Filter NAT rules by flushing the NAT rules from the active NAT rules set. The procedure does not disable IP Filter. You can reactivate IP Filter by adding rules to the rule set.

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Remove NAT from the kernel.**

   ```
   # ipnat -FC
   ```

   The `-C` option removes all entries in the current NAT rule listing. The `-F` option removes all active entries in the current NAT translation table, which shows the currently active NAT mappings.

How to Disable Packet Filtering

When you run this procedure, both packet filtering and NAT are removed from the kernel. If you use this procedure, you must re-enable IP Filter in order to reactivate packet filtering and NAT. For more information, see "How to Re-Enable IP Filter" on page 525.

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Disable packet filtering and allow all packets to pass into the network.**

   ```
   # ipf –D
   ```

Note – The `ipf -D` command flushes the rules from the rule set. When you re-enable filtering, you must add rules to the rule set.

Working With IP Filter Rule Sets

The following task map identifies the procedures associated with IP Filter rule sets.

TABLE 25–3 Working With IP Filter Rule Sets (Task Map)

Task	Description	For Instructions
Manage, view and modify IP Filter packet filtering rule sets.		"Managing Packet Filtering Rule Sets for IP Filter" on page 530
	View an active packet filtering rule set.	"How to View the Active Packet Filtering Rule Set" on page 530
	View an inactive packet filtering rule set.	"How to View the Inactive Packet Filtering Rule Set" on page 531
	Activate a different active rule set.	"How to Activate a Different or Updated Packet Filtering Rule Set" on page 531
	Remove a rule set.	"How to Remove a Packet Filtering Rule Set" on page 532
	Add rules to the rule sets.	"How to Append Rules to the Active Packet Filtering Rule Set" on page 533
		"How to Append Rules to the Inactive Packet Filtering Rule Set" on page 534
	Move between active and inactive rule sets.	"How to Switch Between Active and Inactive Packet Filtering Rule Sets" on page 534
	Delete an inactive rule set from the kernel.	"How to Remove an Inactive Packet Filtering Rule Set From the Kernel" on page 535
Manage, view and modify IP Filter NAT rules.		"Managing NAT Rules for IP Filter" on page 536
	View active NAT rules.	"How to View Active NAT Rules" on page 536
	Remove NAT rules.	"How to Remove NAT Rules" on page 537
	Add additional rules to NAT rules.	"How to Append Rules to the NAT Rules" on page 537

TABLE 25–3 Working With IP Filter Rule Sets (Task Map) *(Continued)*

Task	Description	For Instructions
Manage, view and modify IP Filter address pools.		"Managing Address Pools for IP Filter" on page 538
	View active address pools.	"How to View Active Address Pools" on page 538
	Remove an address pool.	"How to Remove an Address Pool" on page 538
	Add additional rules to an address pool.	"How to Append Rules to an Address Pool" on page 539

Managing Packet Filtering Rule Sets for IP Filter

When is enabled, both active and inactive packet filtering rule sets can reside in the kernel. The active rule set determines what filtering is being done on incoming packets and outgoing packets. The inactive rule set also stores rules. These rules are not used unless you make the inactive rule set the active rule set. You can manage, view, and modify both active and inactive packet filtering rule sets.

▼ How to View the Active Packet Filtering Rule Set

1 Assume a role that includes the IP Filter Management rights profile, or become superuser.

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 View the active packet filtering rule set that is loaded in the kernel.

```
# ipfstat -io
```

Example 25–1 Viewing the Active Packet Filtering Rule Set

The following example shows output from the active packet filtering rule set that is loaded in the kernel.

```
# ipfstat -io
empty list for ipfilter(out)
pass in quick on dmfe1 from 192.168.1.0/24 to any
pass in all
block in on dmfe1 from 192.168.1.10/32 to any
```

How to View the Inactive Packet Filtering Rule Set

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **View the inactive packet filtering rule set.**

   ```
   # ipfstat -I -io
   ```

Example 25–2 Viewing the Inactive Packet Filtering Rule Set

The following example shows output from the inactive packet filtering rule set.

```
# ipfstat -I -io
pass out quick on dmfe1 all
pass in quick on dmfe1 all
```

How to Activate a Different or Updated Packet Filtering Rule Set

Use the following procedure if you want to perform either of the following tasks:

- Activate a packet filtering rule set other than the one that is currently in use by IP Filter.
- Reload the same filtering rule set that has been newly updated.

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Choose one of the following steps:**

   - Create a new rule set in a separate file of your choice if you want to activate an entirely different rule set.
   - Update the current rule set by editing the configuration file that contains that rule set.

3. **Remove the current rule set and load the new rule set.**

   `# ipf -Fa -f` *filename*

   The *filename* can either be the new file with the new rule set or the updated file that contains the active rule set.

   The active rule set is removed from the kernel. The rules in the *filename* file become the active rule set.

Note – You still need to issue the command even if you are reloading the current configuration file. Otherwise, the old rule set continues to be operative, and the modified rule set in the updated configuration file is not applied.

Do not use commands such as `ipf -D` or `svcadm restart` to load the updated rule set. Such commands expose your network by disabling the firewall first before loading the new rule set.

Example 25–3 Activating a Different Packet Filtering Rule Set

The following example shows how to replace one packet filtering rule set with another packet filtering rule set in a separate configuration file, `/etc/ipf/ipf.conf`.

```
# ipfstat -io
empty list for ipfilter(out)
pass in quick on dmfe all
# ipf -Fa -f /etc/ipf/ipf.conf
# ipfstat -io
empty list for ipfilter(out)
block in log quick from 10.0.0.0/8 to any
```

Example 25–4 Reloading an Updated Packet Filtering Rule Set

The following example shows how to reload a packet filtering rule set that is currently active and which is then updated. In this example, the file in use is `/etc/ipf/ipf.conf`.

```
# ipfstat -io (Optional)
empty list for ipfilter (out)
block in log quick from 10.0.0.0/8 to any

(Edit the /etc/ipf/ipf.conf configuration file.)

# ip -Fa -f /etc/ipf/ipf.conf
# ipfstat -io (Optional)
empty list for ipfilter (out)
block in log quick from 10.0.0.0/8 to any
block in quick on elx10 from 192.168.0.0/12 to any
```

▼ How to Remove a Packet Filtering Rule Set

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Remove the rule set.**

   ```
   # ipf -F [a|i|o]
   ```

`-a` Removes all filtering rules from the rule set.

`-i` Removes the filtering rules for incoming packets.

`-o` Removes the filtering rules for outgoing packets.

Example 25–5 Removing a Packet Filtering Rule Set

The following example shows how to remove all filtering rules from the active filtering rule set.

```
# ipfstat -io
block out log on dmf0 all
block in log quick from 10.0.0.0/8 to any
# ipf -Fa
# ipfstat -io
empty list for ipfilter(out)
empty list for ipfilter(in)
```

▼ How to Append Rules to the Active Packet Filtering Rule Set

1 Assume a role that includes the IP Filter Management rights profile, or become superuser.

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 Use one of the following methods to append rules to the active rule set:

- Append rules to the rule set at the command line using the `ipf -f -` command.

  ```
  # echo "block in on dmfe1 proto tcp from 10.1.1.1/32 to any" | ipf -f -
  ```

- Perform the following commands:

  a. Create a rule set in a file of your choice.

  b. Add the rules you have created to the active rule set.

  ```
  # ipf -f filename
  ```

  The rules in *filename* are added to the end of the active rule set. Because uses a "last matching rule" algorithm, the added rules determine filtering priorities, unless you use the `quick` keyword. If the packet matches a rule containing the `quick` keyword, the action for that rule is taken, and no subsequent rules are checked.

Example 25–6 Appending Rules to the Active Packet Filtering Rule Set

The following example shows how to add a rule to the active packet filtering rule set from the command line.

```
# ipfstat -io
empty list for ipfilter(out)
block in log quick from 10.0.0.0/8 to any
# echo "block in on dmfe1 proto tcp from 10.1.1.1/32 to any" | ipf -f -
# ipfstat -io
empty list for ipfilter(out)
block in log quick from 10.0.0.0/8 to any
block in on dmfe1 proto tcp from 10.1.1.1/32 to any
```

How to Append Rules to the Inactive Packet Filtering Rule Set

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Create a rule set in a file of your choice.**

3. **Add the rules you have created to the inactive rule set.**

   `# ipf -I -f` *filename*

   The rules in *filename* are added to the end of the inactive rule set. Because IP Filter uses a "last matching rule" algorithm, the added rules determine filtering priorities, unless you use the `quick` keyword. If the packet matches a rule containing the `quick` keyword, the action for that rule is taken, and no subsequent rules are checked.

Example 25–7 Appending Rules to the Inactive Rule Set

The following example shows how to add a rule to the inactive rule set from a file.

```
# ipfstat -I -io
pass out quick on dmfe1 all
pass in quick on dmfe1 all
# ipf -I -f /etc/ipf/ipf.conf
# ipfstat -I -io
pass out quick on dmfe1 all
pass in quick on dmfe1 all
block in log quick from 10.0.0.0/8 to any
```

How to Switch Between Active and Inactive Packet Filtering Rule Sets

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Switch the active and inactive rule sets.**

```
# ipf -s
```

This command enables you to switch between the active and inactive rule sets in the kernel. Note that if the inactive rule set is empty, there is no packet filtering.

Example 25–8 Switching Between the Active and Inactive Packet Filtering Rule Sets

The following example shows how using the `ipf -s` command results in the inactive rule set becoming the active rule set and the active rule set becoming the inactive rule set.

- Before running the `ipf -s` command, the output from the `ipfstat -I -io` command shows the rules in the inactive rule set. The output from the `ipfstat -io` command shows the rules in the active rule set.

```
# ipfstat -io
empty list for ipfilter(out)
block in log quick from 10.0.0.0/8 to any
block in on dmfe1 proto tcp from 10.1.1.1/32 to any
# ipfstat -I -io
pass out quick on dmfe1 all
pass in quick on dmfe1 all
block in log quick from 10.0.0.0/8 to any
```

- After running the `ipf -s` command, the output from the `ipfstat -I -io` and the `ipfstat -io` command show that the content of the two rules sets have switched.

```
# ipf -s
Set 1 now inactive
# ipfstat -io
pass out quick on dmfe1 all
pass in quick on dmfe1 all
block in log quick from 10.0.0.0/8 to any
# ipfstat -I -io
empty list for inactive ipfilter(out)
block in log quick from 10.0.0.0/8 to any
block in on dmfe1 proto tcp from 10.1.1.1/32 to any
```

▼ How to Remove an Inactive Packet Filtering Rule Set From the Kernel

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Specify the inactive rule set in the "flush all" command.**

```
# ipf -I -Fa
```

This command flushes the inactive rule set from the kernel.

Note – If you subsequently run `ipf -s`, the empty inactive rule set will become the active rule set. An empty active rule set means that *no* filtering will be done.

Example 25–9 Removing an Inactive Packet Filtering Rule Set From the Kernel

The following example shows how to flush the inactive packet filtering rule set so that all rules have been removed.

```
# ipfstat -I -io
empty list for inactive ipfilter(out)
block in log quick from 10.0.0.0/8 to any
block in on dmfe1 proto tcp from 10.1.1.1/32 to any
# ipf -I -Fa
# ipfstat -I -io
empty list for inactive ipfilter(out)
empty list for inactive ipfilter(in)
```

Managing NAT Rules for IP Filter

Use the following procedures to manage, view, and modify NAT rules.

▼ How to View Active NAT Rules

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **View the active NAT rules.**

   ```
   # ipnat -l
   ```

Example 25–10 Viewing Active NAT Rules

The following example shows the output from the active NAT rules set.

```
# ipnat -l
List of active MAP/Redirect filters:
map dmfe0 192.168.1.0/24 -> 20.20.20.1/32

List of active sessions:
```

How to Remove NAT Rules

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Remove the current NAT rules.**

   ```
   # ipnat -C
   ```

Example 25–11 Removing NAT Rules

The following example shows how to remove the entries in the current NAT rules.

```
# ipnat -l
List of active MAP/Redirect filters:
map dmfe0 192.168.1.0/24 -> 20.20.20.1/32

List of active sessions:
# ipnat -C
1 entries flushed from NAT list
# ipnat -l
List of active MAP/Redirect filters:

List of active sessions:
```

How to Append Rules to the NAT Rules

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Use one of the following methods to append rules to the active rule set:**

   - Append rules to the NAT rule set at the command line using the `ipnat -f -` command.

     ```
     # echo "map dmfe0 192.168.1.0/24 -> 20.20.20.1/32" | ipnat -f -
     ```

   - Perform the following commands:

     a. Create additional NAT rules in a file of your choice.

     b. Add the rules you have created to the active NAT rules.

        ```
        # ipnat -f filename
        ```

        The rules in *filename* are added to the end of the NAT rules.

Example 25–12 Appending Rules to the NAT Rule Set

The following example shows how to add a rule to the NAT rule set from the command line.

```
# ipnat -l
List of active MAP/Redirect filters:

List of active sessions:
# echo "map dmfe0 192.168.1.0/24 -> 20.20.20.1/32" | ipnat -f -
# ipnat -l
List of active MAP/Redirect filters:
map dmfe0 192.168.1.0/24 -> 20.20.20.1/32

List of active sessions:
```

Managing Address Pools for IP Filter

Use the following procedures to manage, view, and modify address pools.

▼ How to View Active Address Pools

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **View the active address pool.**

   ```
   # ippool -l
   ```

Example 25–13 Viewing the Active Address Pool

The following example shows how to view the contents of the active address pool.

```
# ippool -l
table role = ipf type = tree number = 13
        { 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24; };
```

▼ How to Remove an Address Pool

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Remove the entries in the current address pool.**

```
# ippool -F
```

Example 25–14 Removing an Address Pool

The following example shows how to remove an address pool.

```
# ippool -l
table role = ipf type = tree number = 13
        { 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24; };
# ippool -F
1 object flushed
# ippool -l
```

How to Append Rules to an Address Pool

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Use one of the following methods to append rules to the active rule set:**

- Append rules to the rule set at the command line using the `ippool -f -` command.

  ```
  # echo "table role = ipf type = tree number = 13
  {10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24};" | ippool -f -
  ```

- Perform the following commands:

  a. Create additional address pools in a file of your choice.

  b. Add the rules you have created to the active address pool.

  ```
  # ippool -f filename
  ```

  The rules in *filename* are added to the end of the active address pool.

Example 25–15 Appending Rules to an Address Pool

The following example shows how to add an address pool to the address pool rule set from the command line.

```
# ippool -l
table role = ipf type = tree number = 13
        { 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24; };
# echo "table role = ipf type = tree number = 100
 {10.0.0.0/32, 172.16.1.2/32, 192.168.1.0/24};" | ippool -f -
# ippool -l
table role = ipf type = tree number = 100
```

```
        { 10.0.0.0/32, 172.16.1.2/32, 192.168.1.0/24; };
table role = ipf type = tree number = 13
        { 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24; };
```

Displaying Statistics and Information for IP Filter

TABLE 25–4 Displaying IP Filter Statistics and Information (Task Map)

Task	Description	For Instructions
View state tables.	View state tables to obtain information about packet filtering using the `ipfstat` command.	"How to View State Tables for IP Filter" on page 540
View state statistics.	View statistics on packet state information using the `ipfstat -s` command.	"How to View State Statistics for IP Filter" on page 541
View NAT statistics.	View NAT statistics using the `ipnat -s` command.	"How to View NAT Statistics for IP Filter" on page 542
View address pool statistics.	View address pool statistics using the `ippool -s` command.	"How to View Address Pool Statistics for IP Filter" on page 542

▼ How to View State Tables for IP Filter

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **View the state table.**

```
# ipfstat
```

Note – You can use the `-t` option to view the state table in the top utility format.

Example 25–16 Viewing State Tables for IP Filter

The following example shows how to view a state table.

```
# ipfstat
bad packets:            in 0    out 0
 input packets:         blocked 160 passed 11 nomatch 1 counted 0 short 0
output packets:         blocked 0 passed 13681 nomatch 6844 counted 0 short 0
```

```
 input packets logged:  blocked 0 passed 0
output packets logged:  blocked 0 passed 0
 packets logged:        input 0 output 0
 log failures:          input 0 output 0
fragment state(in):     kept 0  lost 0
fragment state(out):    kept 0  lost 0
packet state(in):       kept 0  lost 0
packet state(out):      kept 0  lost 0
ICMP replies:   0       TCP RSTs sent:  0
Invalid source(in):     0
Result cache hits(in):  152     (out):  6837
IN Pullups succeeded:   0       failed: 0
OUT Pullups succeeded:  0       failed: 0
Fastroute successes:    0       failures:       0
TCP cksum fails(in):    0       (out):  0
IPF Ticks:      14341469
Packet log flags set: (0)
        none
```

▼ How to View State Statistics for IP Filter

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **View the state statistics.**

```
# ipfstat -s
```

Example 25–17 Viewing State Statistics for IP Filter

The following example shows how to view state statistics.

```
# ipfstat -s
IP states added:
        0 TCP
        0 UDP
        0 ICMP
        0 hits
        0 misses
        0 maximum
        0 no memory
        0 max bucket
        0 active
        0 expired
        0 closed
State logging enabled

State table bucket statistics:
        0 in use
        0.00% bucket usage
```

```
0 minimal length
0 maximal length
0.000 average length
```

How to View NAT Statistics for IP Filter

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **View NAT statistics.**

   ```
   # ipnat -s
   ```

Example 25–18 Viewing NAT Statistics for IP Filter

The following example shows how to view NAT statistics.

```
# ipnat -s
mapped  in      0       out     0
added   0       expired 0
no memory       0       bad nat 0
inuse   0
rules   1
wilds   0
```

How to View Address Pool Statistics for IP Filter

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **View address pool statistics.**

   ```
   # ippool -s
   ```

Example 25–19 Viewing Address Pool Statistics for IP Filter

The following example shows how to view address pool statistics.

```
# ippool -s
Pools:  3
Hash Tables:    0
Nodes:  0
```

Working With Log Files for IP Filter

TABLE 25–5 Working With IP Filter Log Files (Task Map)

Task	Description	For Instructions
Create a log file.	Create a separate IP filter log file.	"How to Set Up a Log File for IP Filter" on page 543
View log files.	View state, NAT, and normal log files using the `ipmon` command.	"How to View IP Filter Log Files" on page 544
Flush the packet log buffer.	Remove the contents of the packet log buffer using the `ipmon -F` command.	"How to Flush the Packet Log File" on page 545
Save logged packets to a file.	Save logged packets to a file for later reference.	"How to Save Logged Packets to a File" on page 546

▼ How to Set Up a Log File for IP Filter

By default, all log information for IP Filter is recorded in the `syslogd` file. You should set up a log file to record IP Filter traffic information separately from other data that might be logged in the default log file. Perform the following steps.

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Edit the `/etc/syslog.conf` file by adding the following two lines:**

   ```
   # Save IPFilter log output to its own file
   local0.debug             /var/log/log-name
   ```

   Note – On the second line, make sure to use the Tab key, not the Spacebar, to separate `local0.debug` from `/var/log/`*log-name*.

3. **Create the new log file.**

   ```
   # touch /var/log/log-name
   ```

4. **Restart the system-log service.**

   ```
   # svcadm restart system-log
   ```

Example 25–20 Creating a IP Filter Log

The following example shows how to create `ipmon.log` to archive IP filter information.

In `/etc/syslog.conf`:

```
# Save IPFilter log output to its own file
local0.debug             /var/log/ipmon.log
```

At the command line:

```
# touch /var/log/ipmon.log
# svcadm restart system-log
```

▼ How to View IP Filter Log Files

Before You Begin You should create a separate log file to record IP Filter data. Refer to "How to Set Up a Log File for IP Filter" on page 543.

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **View the state, NAT, or normal log files. To view a log file, type the following command, using the appropriate option:**

   `# ipmon -o [S|N|I]` *filename*

   `S` Displays the state log file.

   `N` Displays the NAT log file.

   `I` Displays the normal IP log file.

   To view all state, NAT, and normal log files, use all the options:

   `# ipmon -o SNI` *filename*

   - **Provided that you have manually stopped the `ipmon` daemon first, you can also use the following command to display state, NAT, and IP filter log files:**

     `# ipmon -a` *filename*

Note – Do not use the `ipmon -a` syntax if the `ipmon` daemon is still running. Normally, the daemon is automatically started during system boot. Issuing the `ipmon -a` command also opens another copy of `ipmon`. In such a case, both copies read the same log information, and only one gets a particular log message.

For more information about viewing log files, see the `ipmon(1M)` man page.

Example 25–21 Viewing IP Filter Log Files

The following example shows the output from `/var/ipmon.log`.

```
# ipmon -o SNI /var/ipmon.log
02/09/2004 15:27:20.606626 bge0 @0:1 p 129.146.157.149 ->
129.146.157.145 PR icmp len 20 84 icmp echo/0 IN
```

or

```
# pkill ipmon
# ipmon -aD /var/ipmon.log
02/09/2004 15:27:20.606626 bge0 @0:1 p 129.146.157.149 ->
129.146.157.145 PR icmp len 20 84 icmp echo/0 IN
```

How to Flush the Packet Log File

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Flush the pack log buffer.**

   ```
   # ipmon -F
   ```

Example 25–22 Flushing the Packet Log File

The following example shows the output when a log file is removed. The system provides a report even when there is nothing stored in the log file, as in this example.

```
# ipmon -F
0 bytes flushed from log buffer
0 bytes flushed from log buffer
0 bytes flushed from log buffer
```

▼ How to Save Logged Packets to a File

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Save the logged packets to a file.**

```
# cat /dev/ipl > filename
```

Continue logging packets to the *filename* file until you interrupt the procedure by typing `Control-C` to get the command line prompt back.

Example 25–23 Saving Logged Packets to a File

The following example shows the result when logged packets are saved to a file.

```
# cat /dev/ipl > /tmp/logfile
^C#

# ipmon -f /tmp/logfile
02/09/2004 15:30:28.708294 bge0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 52 -S IN
02/09/2004 15:30:28.708708 bge0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 40 -A IN
02/09/2004 15:30:28.792611 bge0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 70 -AP IN
02/09/2004 15:30:28.872000 bge0 @0:1 p 129.146.157.149,33923 ->
 129.146.157.145,23 PR tcp len 20 40 -A IN
02/09/2004 15:30:28.872142 bge0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 43 -AP IN
02/09/2004 15:30:28.872808 bge0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 40 -A IN
02/09/2004 15:30:28.872951 bge0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 47 -AP IN
02/09/2004 15:30:28.926792 bge0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 40 -A IN
.
.
(output truncated)
```

Creating and Editing IP Filter Configuration Files

You must directly edit the configuration files to create and modify rule sets and address pools. Configuration files follow standard UNIX syntax rules:

- The pound sign (#) indicates a line containing comments.
- Rules and comments can coexist on the same line.

- Extraneous white space is allowed to keep rules easy to read.
- Rules can be more than one line long. Use the backslash (\) at the end of a line to indicate that the rule continues on the next line.

How to Create a Configuration File for IP Filter

The following procedure describes how to set up the following:

- Packet filtering configuration files
- NAT rules configuration files
- Address pool configuration files

1 Assume a role that includes the IP Filter Management rights profile, or become superuser.

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 Start the file editor of your choice. Create or edit the configuration file for the feature you want to configure.

- To create a configuration file for packet filtering rules, edit the `ipf.conf` file.

  IP Filter uses the packet filtering rules that you put in to the `ipf.conf` file. If you locate the rules file for packet filtering in the `/etc/ipf/ipf.conf` file, this file is loaded when the system is booted. If you do not want the filtering rules to be loaded at boot time, put the in a file of your choice. You can then activate the rules with the `ipf` command, as described in "How to Activate a Different or Updated Packet Filtering Rule Set" on page 531.

  See "Using IP Filter's Packet Filtering Feature" on page 514 for information about creating packet filtering rules.

  Note – If the `ipf.conf` file is empty, there is no filtering. An empty `ipf.conf` file is the same as having a rule set that reads:

  ```
  pass in all
  pass out all
  ```

- To create a configuration file for NAT rules, edit the `ipnat.conf` file.

  IP Filter uses the NAT rules that you put in to the `ipnat.conf` file. If you locate the rules file for NAT in the `/etc/ipf/ipnat.conf` file, this file is loaded when the system is booted. If you do not want the NAT rules loaded at boot time, put the `ipnat.conf` file in a location of your choice. You can then activate the NAT rules with the `ipnat` command.

See "Using IP Filter's NAT Feature" on page 516 for information about creating rules for NAT.

- To create a configuration file for address pools, edit the `ippool.conf` file.

  IP Filter uses the pool of addresses that you put in to the `ippool.conf` file. If you locate the rules file for the pool of addresses in the `/etc/ipf/ippool.conf` file, this file is loaded when the system is booted. If you do not want the pool of addresses loaded at boot time, put the `ippool.conf` file in a location of your choice. You can then activate the pool of addresses with the `ippool` command.

  See "Using IP Filter's Address Pools Feature" on page 518 for information about creating address pools.

IP Filter Configuration File Examples

The following examples provide an illustration of packet filtering rules used in filtering configurations.

EXAMPLE 25–24 IP Filter Host Configuration

This example shows a configuration on a host machine with an bge network interface.

```
# pass and log everything by default
pass in log on bge0 all
pass out log on bge0 all

# block, but don't log, incoming packets from other reserved addresses
block in quick on bge0 from 10.0.0.0/8 to any
block in quick on bge0 from 172.16.0.0/12 to any

# block and log untrusted internal IPs. 0/32 is notation that replaces
# address of the machine running Solaris IP Filter.
block in log quick from 192.168.1.15 to <thishost>
block in log quick from 192.168.1.43 to <thishost>

# block and log X11 (port 6000) and remote procedure call
# and portmapper (port 111) attempts
block in log quick on bge0 proto tcp from any to bge0/32 port = 6000 keep state
block in log quick on bge0 proto tcp/udp from any to bge0/32 port = 111 keep state
```

This rule set begins with two unrestricted rules that allow everything to pass into and out of the bge interface. The second set of rules blocks any incoming packets from the private address spaces `10.0.0.0` and `172.16.0.0` from entering the firewall. The next set of rules blocks specific internal addresses from the host machine. Finally, the last set of rules blocks packets coming in on port 6000 and port 111.

EXAMPLE 25–25 IP Filter Server Configuration

This example shows a configuration for a host machine acting as a web server. This machine has an e1000g network interface.

```
# web server with an e1000g interface
# block and log everything by default;
# then allow specific services
# group 100 - inbound rules
# group 200 - outbound rules
# (0/32) resolves to our IP address)
*** FTP proxy ***


# block short packets which are packets
# fragmented too short to be real.
block in log quick all with short


# block and log inbound and outbound by default,
# group by destination
block in log on e1000g0 from any to any head 100
block out log on e1000g0 from any to any head 200


# web rules that get hit most often
pass in quick on e1000g0 proto tcp from any \
to e1000g0/32 port = http flags S keep state group 100
pass in quick on e1000g0 proto tcp from any \
to e1000g0/32 port = https flags S keep state group 100


# inbound traffic - ssh, auth
pass in quick on e1000g0 proto tcp from any \
to e1000g0/32 port = 22 flags S keep state group 100
pass in log quick on e1000g0 proto tcp from any \
to e1000g0/32 port = 113 flags S keep state group 100
pass in log quick on e1000g0 proto tcp from any port = 113 \
to e1000g0/32 flags S keep state group 100


# outbound traffic - DNS, auth, NTP, ssh, WWW, smtp
pass out quick on e1000g0 proto tcp/udp from e1000g0/32 \
to any port = domain flags S keep state group 200
pass in quick on e1000g0 proto udp from any \
port = domain to e1000g0/32 group 100

pass out quick on e1000g0 proto tcp from e1000g0/32 \
to any port = 113 flags S keep state group 200
pass out quick on e1000g0 proto tcp from e1000g0/32 port = 113 \
to any flags S keep state group 200

pass out quick on e1000g0 proto udp from e1000g0/32 to any \
port = ntp group 200
pass in quick on e1000g0 proto udp from any \
port = ntp to e1000g0/32 port = ntp group 100
```

EXAMPLE 25–25 IP Filter Server Configuration *(Continued)*

```
pass out quick on e1000g0 proto tcp from e1000g0/32 \
to any port = ssh flags S keep state group 200

pass out quick on e1000g0 proto tcp from e1000g0/32 \
to any port = http flags S keep state group 200
pass out quick on e1000g0 proto tcp from e1000g0/32 \
to any port = https flags S keep state group 200

pass out quick on e1000g0 proto tcp from e1000g0/32 \
to any port = smtp flags S keep state group 200


# pass icmp packets in and out
pass in quick on e1000g0 proto icmp from any to e1000g0/32  keep state group 100
pass out quick on e1000g0 proto icmp from e1000g0/32 to any keep state group 200


# block and ignore NETBIOS packets
block in quick on e1000g0 proto tcp from any \
to any port = 135 flags S keep state group 100

block in quick on e1000g0 proto tcp from any port = 137 \
to any flags S keep state group 100
block in quick on e1000g0 proto udp from any to any port = 137 group 100
block in quick on e1000g0 proto udp from any port = 137 to any group 100

block in quick on e1000g0 proto tcp from any port = 138 \
to any flags S keep state group 100
block in quick on e1000g0 proto udp from any port = 138 to any group 100

block in quick on e1000g0 proto tcp from any port = 139 to any flags S keep state
group 100
block in quick on e1000g0 proto udp from any port = 139 to any group 100
```

EXAMPLE 25–26 IP Filter Router Configuration

This example shows a configuration for a router that has an internal interface, nge, and an external interface, ce1.

```
# internal interface is nge0 at 192.168.1.1
# external interface is nge1 IP obtained via DHCP
# block all packets and allow specific services
*** NAT ***
*** POOLS ***


# Short packets which are fragmented too short to be real.
block in log quick all with short


# By default, block and log everything.
block in log on nge0 all
block in log on nge1 all
block out log on nge0 all
```

EXAMPLE 25–26 IP Filter Router Configuration *(Continued)*

```
block out log on nge1 all


# Packets going in/out of network interfaces that aren't on the loopback
# interface should not exist.
block in log quick on nge0 from 127.0.0.0/8 to any
block in log quick on nge0 from any to 127.0.0.0/8
block in log quick on nge1 from 127.0.0.0/8 to any
block in log quick on nge1 from any to 127.0.0.0/8


# Deny reserved addresses.
block in quick on nge1 from 10.0.0.0/8 to any
block in quick on nge1 from 172.16.0.0/12 to any
block in log quick on nge1 from 192.168.1.0/24 to any
block in quick on nge1 from 192.168.0.0/16 to any


# Allow internal traffic
pass in quick on nge0 from 192.168.1.0/24 to 192.168.1.0/24
pass out quick on nge0 from 192.168.1.0/24 to 192.168.1.0/24


# Allow outgoing DNS requests from our servers on .1, .2, and .3
pass out quick on nge1 proto tcp/udp from nge1/32 to any port = domain keep state
pass in quick on nge0 proto tcp/udp from 192.168.1.2 to any port = domain keep state
pass in quick on nge0 proto tcp/udp from 192.168.1.3 to any port = domain keep state


# Allow NTP from any internal hosts to any external NTP server.
pass in quick on nge0 proto udp from 192.168.1.0/24 to any port = 123 keep state
pass out quick on nge1 proto udp from any to any port = 123 keep state


# Allow incoming mail
pass in quick on nge1 proto tcp from any to nge1/32 port = smtp keep state
pass in quick on nge1 proto tcp from any to nge1/32 port = smtp keep state
pass out quick on nge1 proto tcp from 192.168.1.0/24 to any port = smtp keep state


# Allow outgoing connections: SSH, WWW, NNTP, mail, whois
pass in quick on nge0 proto tcp from 192.168.1.0/24 to any port = 22 keep state
pass out quick on nge1 proto tcp from 192.168.1.0/24 to any port = 22 keep state

pass in quick on nge0 proto tcp from 192.168.1.0/24 to any port = 80 keep state
pass out quick on nge1 proto tcp from 192.168.1.0/24 to any port = 80 keep state
pass in quick on nge0 proto tcp from 192.168.1.0/24 to any port = 443 keep state
pass out quick on nge1 proto tcp from 192.168.1.0/24 to any port = 443 keep state

pass in quick on nge0 proto tcp from 192.168.1.0/24 to any port = nntp keep state
block in quick on nge1 proto tcp from any to any port = nntp keep state
pass out quick on nge1 proto tcp from 192.168.1.0/24 to any port = nntp keep state

pass in quick on nge0 proto tcp from 192.168.1.0/24 to any port = smtp keep state
```

EXAMPLE 25–26 IP Filter Router Configuration *(Continued)*

```
pass in quick on nge0 proto tcp from 192.168.1.0/24 to any port = whois keep state
pass out quick on nge1 proto tcp from any to any port = whois keep state


# Allow ssh from offsite
pass in quick on nge1 proto tcp from any to nge1/32 port = 22 keep state


# Allow ping out
pass in quick on nge0 proto icmp all keep state
pass out quick on nge1 proto icmp all keep state


# allow auth out
pass out quick on nge1 proto tcp from nge1/32 to any port = 113 keep state
pass out quick on nge1 proto tcp from nge1/32 port = 113 to any keep state


# return rst for incoming auth
block return-rst in quick on nge1 proto tcp from any to any port = 113 flags S/SA


# log and return reset for any TCP packets with S/SA
block return-rst in log on nge1 proto tcp from any to any flags S/SA


# return ICMP error packets for invalid UDP packets
block return-icmp(net-unr) in proto udp all
```

PART IV

Networking Performance

This part discusses networking performance features such as integrated load balancing and virtual router redundancy protocol.

◆ ◆ ◆ CHAPTER 26

Integrated Load Balancer Overview

Integrated Load Balancer (ILB) provides Layer 3 and Layer 4 load-balancing capabilities for the Oracle Solaris operating system (OS) installed on SPARC and x86 based systems. ILB intercepts incoming requests from clients, decides which back-end server should handle the request based on load-balancing rules, and then forwards the request to the selected server. ILB performs optional health checks and provides the data for the load-balancing algorithms to verify if the selected server can handle the incoming request.

This chapter discusses the following sections:

The key features of ILB include:

- Support for stateless Direct Server Return (DSR) and Network Address Translation (NAT) modes of operation for IPv4 and IPv6
- Allows ILB administration through a command-line interface (CLI)
- Provides server monitoring capabilities through health checks

ILB has three major components:

- `ilbadm` CLI – You can use this interface to configure load-balancing rules, perform optional health checks, and view statistics.
- `libilb` configuration library – `ilbadm` and other third-party applications can use the functionality implemented in `libilb` for ILB administration.
- `ilbd` daemon – This daemon performs the following tasks:
  - Manages persistent configuration

- Provides serial access to the ILB kernel module by processing the configuration information and sending it to the ILB kernel module for execution
- Performs health checks and notifies the results to the ILB kernel module so that the load distribution is properly adjusted

ILB Terminology

This section describes some terms that are useful to know when implementing ILB on your systems.

connection draining

A mechanism that provides the capability to prevent new connections to a server that is administratively disabled. This feature is useful for shutting down the servers without disrupting the active connections or sessions. The already existing connections to the server will work normally. After the server is ready to handle the requests, it can be administratively enabled again and the load balancer will forward the new connections to it. ILB provides this capability only for the servers with NAT-based virtual services.

Direct Server Return mode (DSR)

Refers to load-balancing incoming requests to the back-end servers and letting the return traffic from the servers bypass the load balancer by sending them directly to the client. ILB's current implementation of DSR does not provide TCP connection tracking (meaning that it is stateless).

Advantages:

- Better performance than NAT because only the destination MAC address of packets is changed and servers respond directly to clients.
- Full transparency: The servers see a connection directly from the client IP address and reply to the client through the default gateway.

Disadvantages:

- The back-end server must respond to both its own IP address (for health checks) and the virtual IP address (for load balanced traffic).
- Because the load balancer maintains no connection state (meaning that it is stateless), adding or removing servers will cause connection disruption.

load-balancing algorithm

The algorithm that ILB uses to select a back-end server from a server group for an incoming request.

load-balancing rule

In ILB, a virtual service is represented by a load-balancing rule and is defined by the following parameters:

- Virtual IP address
- Transport protocol: TCP or UDP
- Port number (or a port range)
- Load-balancing algorithm
- Type of load-balancing mode (DSR, full-NAT, or half-NAT)
- Server group consisting of a set of back-end servers
- Optional server health checks that can be executed for each server in the server group
- Optional port to use for health checks

Note – You can specify health checks on a particular port or on any port that the `ilbd` daemon randor from the port range for the server.

- Rule name to represent a virtual service

NAT-based load-balancing

Involves rewriting the IP header information, and handles both the request and the response traffic. There are two types of NAT: half-NAT and full-NAT. Both types rewrite the destination IP address. However, full-NAT also rewrites the source IP address, making it appear to the server that all connections are originating from the load balancer. NAT does provide TCP connection tracking (meaning that it is stateful).

Advantages:

- Works with all back-end servers by changing the default gateway to point to the load balancer.
- Because the load balancer maintains the connection state, adding or removing servers without connection disruption is possible.

Disadvantages:

- Slower performance than DSR because processing involves manipulation of the IP header and servers send responses to the load balancer.
- All the back-end servers must use the load balancer as a default gateway.

persistent configuration

In the context of ILB, a persistent configuration is a configuration (that is, a set of load-balancing rules) that persists across reboots and package updates.

proxy source

The range of IP addresses that can act as proxies. The range is limited to 10 IP addresses. The proxy source is required only when you have the full NAT implementation.

session

Consists of a number of packets that come from the same client during a time period, which might have some meaning as a whole.

session persistence Allows all packets from a client to be sent to the same back-end server. Also known as stickiness. You can setup simple session persistence (that is, source address persistence) for a virtual service by specifying the options `pmask=prefix length` and `persist-timeout=value in seconds`. After session persistence is established between a client and a server, all packets from the client to the virtual service are forwarded to the same back-end server as long as the persistence exists. The prefix length in CIDR notation is a value between 0–32 for IPv4 and 0–128 for IPv6.

server group Consists of zero or more back-end servers and must contain at least one server when it is used for a virtual service. For example, if you want to load balance HTTP requests, you must configure ILB with a server group consisting of one or more back-end servers. ILB will balance the HTTP traffic across the configured set of servers.

server ID A unique name for the IP address that is assigned by the system when the server is added to a server group.

virtual IP address (VIP) The IP address for a virtual service.

virtual service A service that the clients see as `VIP:port`. For example: `www.foo.com:80`. Although the service is being handled by a server group potentially consisting of more than one server, the server group appears to clients of the virtual service as a single `IP address:port`. A single server can be included in more than one server group and hence can serve multiple virtual services. Also, a single server group can service multiple virtual services.

Features of ILB

This section describes the key features of ILB.

ILB Operation Modes

ILB supports stateless DSR and NAT modes of operation for IPv4 and IPv6, in single-legged and dual-legged topologies.

- Stateless DSR mode – In the DSR mode, ILB balances the incoming requests to the back-end servers, but lets the return traffic from the servers to the clients bypass it. However, you can also set up ILB to be used as a router for the back-end server. In this case, the response from the back-end server to the client is routed through the machine that is running ILB. With stateless DSR, ILB does not save any state information of the processed packets, except for basic statistics. Since ILB does not save any state in this mode, the performance is comparable to the normal IP forwarding performance. This mode is best suited for connectionless protocols.
- NAT mode (full-NAT and half-NAT) – ILB uses NAT in stand-alone mode strictly for load-balancing functionality. In this mode, ILB rewrites the header information and handles the incoming as well as the outgoing traffic. NAT mode provides additional security and is best suited for HTTP (or SSL) traffic.

Note – The NAT code path that is implemented in ILB differs from the code path that is implemented in Oracle Solaris' IP Filter feature. Do *not* use both of these code paths simultaneously.

ILB Algorithms

ILB algorithms control traffic distributions and provide various characteristics for load distribution and server selection. ILB provides the following algorithms for the two modes of operation:

- Round-robin – In a round-robin algorithm, the load balancer assigns the requests to a list of the servers on a rotating basis. Once a server is assigned a request, the server is moved to the end of the list.
- *src IP* hash – In source IP hash method, the load balancer selects a server based on the hash value of the source IP address of the incoming request.
- *src-IP, port* hash – In source IP, port hash method, the load balancer selects a server based on the hash value of the source IP address, and the source port of the incoming request.
- *src-IP, VIP* hash – In source IP, VIP hash method, the load balancer selects a server based on the hash value of the source IP address, and the destination IP address of the incoming request.

ILB Command-Line Interface

The CLI is located in the `/usr/sbin/ilbadm` directory. It includes subcommands to configure load-balancing rules, server groups, and health checks. It also includes subcommands to display statistics as well as view configuration details. The subcommands can be divided into two categories:

- Configuration subcommands – These subcommands enable you to perform the following tasks:
  - Create and delete load-balancing rules
  - Enable and disable load-balancing rules
  - Create and delete server groups
  - Add and remove servers from a server group
  - Enable and disable back-end servers
  - Create and delete server health checks for a server group within a load-balancing rule

Note – To administer the configuration subcommands, you require privileges. The privileges are obtained through Role Based Access Control (RBAC). To create the appropriate role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

- View subcommands – These subcommands enable you to perform the following tasks:
  - View configured load-balancing rules, server groups, and health checks
  - View packet forwarding statistics
  - View the NAT connection table
  - View health check results
  - View the session persistence mapping table

Note – You do not need privileges to administer the view subcommands.

For a list of `ilbadm` subcommands, see "ILB Command and Subcommands" on page 564 . For more detailed information about `ilbadm` subcommands, refer to the `ilbadm(1M)` man page.

ILB Server Monitoring Feature

ILB offers an optional server monitoring feature that can provide server health checks with the following capabilities:

- Built-in ping probes
- Built-in TCP probes
- Built-in UDP probes
- User-supplied tests that can be run as server health checks

By default, ILB does not perform any health checks. You can specify health checks for each server group when creating a load-balancing rule. You can configure only one health check per load-balancing rule. As long as a virtual service is enabled, the health checks on the server group that is associated with the enabled virtual service start automatically and repeat periodically. The health checks stop as soon as the virtual service is disabled. The previous health check states are not preserved when the virtual service is re-enabled.

When you specify a TCP, UDP, or custom test probe for running a health check, ILB sends a ping probe, by default, to determine if the server is reachable before it sends the specified TCP, UDP, or custom test probe to the server. The ping probe is a method of monitoring server health. If the ping probe fails, the corresponding server is disabled with the health check status of `unreachable`. If the ping probe succeeds, but the TCP, UDP, or custom test probe fails, the server is disabled with the health check status of `dead`.

Note –

- You can disable the default ping probe.
- The default ping probe cannot be disabled for the UDP probe. Thus, for the UDP health checks, the ping probe is always the default probe.

You can configure the health check for the parameters shown in the following table.

TABLE 26–1 Configuring Health Check Parameters

Health Check Parameters	Description
`hc-test`	Specifies the type of health check to be performed.
`hc-timeout`	Initiates a timeout when a health check is not complete.
`hc-interval`	Specifies the Interval between consecutive health checks. **Note –** Intervals are randomized between the following values: `0.5*hc-interval` and `1.5*hc-interval`.
`hc-count`	Specifies the number of consecutive failed checks before a server is considered faulty.

Additional ILB features

This section describes the additional features of the ILB.

- **Enables clients to ping virtual IP (VIP) addresses** – ILB can respond to Internet Control Message Protocol (ICMP) echo requests to VIPs from clients. ILB provides this capability for DSR and NAT modes of operation.
- **Enables you to add and remove servers from a server group without interrupting service** – You can dynamically add and remove servers from a server group, without interrupting existing connections established with the back-end servers. ILB provides this capability for the NAT mode of operations.
- **Enables you to configure session persistence (stickiness)** – For many applications, it is important that a series of connections, packets or both from the same client are sent to the same back-end server. You can configure session persistence for a virtual service by specifying the netmask in the subcommand `create-rule{{-m persist=<netmask>]]`. After a persistent mapping is created, subsequent requests for connections packets or both to a virtual service with a matching source IP address of the client are forwarded to the same back-end server. The support for session persistence mechanism is available for both DSR and NAT modes of operation.

- **Enables you to perform connection draining** – ILB provides support for this capability only for servers of NAT-based virtual services. This capability prevents new connections from being sent to a server that is disabled. Existing connections to the server continue to function. After all the connections to that server terminate, the server can then be shut down for maintenance. After the server is ready to handle requests, enable the server so that the load balancer can forward new connections to it. This feature enables you to shut down servers for maintenance without disrupting active connections or sessions.
- **Enables load-balancing TCP and UDP ports** – ILB can load balance all ports on a given IP address across different sets of servers without requiring you to set up explicit rules for each port. ILB provides this capability for DSR and NAT modes of operation.
- **Enables you to specify independent ports for virtual services within the same server group** – With this feature, ILB enables you to specify different destination ports for different servers in the same server group for the NAT modes of operation.
- **Enables you to load balance simple port range** – ILB can load balance a range of ports on the VIP to a given server group. For convenience, you can conserve IP addresses by load-balancing different port ranges on the same VIP to different sets of back-end servers. Also, when session persistence is enabled for NAT mode, ILB sends requests from the same client IP address for different ports in the range to the same back-end server.
- **Enables port range shifting and collapsing** – Port range shifting and collapsing depend on the port range of a server in a load-balancing rule. So, if the port range of a server is different from the VIP port range, port shifting is automatically implemented. Port collapsing is implemented, if the server port range is a single port. These features are provided for the NAT modes of operation.

ILB Processes

This section describes the working of ILB processes like the client-to-server packet processing and server-to-client packet processing.

Client-to-server packet processing:

1. ILB receives an incoming request that is sent by the client to a VIP address and matches the request to a load-balancing rule.
2. If ILB finds a matching load-balancing rule, it uses a load-balancing algorithm to forward the request to the back-end server depending on the mode of operation.
   - In DSR mode, ILB replaces the MAC header of the incoming request with the MAC header of the selected back-end server.
   - In half-NAT mode, ILB replaces the destination IP address and the transport protocol port number of the incoming request with that of the selected back-end server.

- In full-NAT mode, ILB replaces the source IP address and the transport protocol port number of the incoming request with the load-balancing rule's NAT source address. ILB also replaces the destination IP address and the transport protocol port number of the incoming request with that of the selected back-end server.

3. ILB forwards the modified incoming request to the selected back-end server.

Server-to-client packet processing:

1. The back-end server sends a reply to ILB in response to the incoming request from the client.
2. ILB's action after receiving the response from the back-end server is based on the mode of operation, as follows:
   - In normal DSR mode, the response from the back-end server bypasses ILB and goes directly to the client. However, if ILB is also used as a router for the back-end server, then the response from the back-end server to the client is routed through the machine running ILB.
   - In half-NAT mode and full-NAT mode, ILB matches the response from the back-end server to the incoming request and replaces the changed IP address and the transport protocol port number with that of the original incoming request. ILB then forwards the response to the client.

Guidelines for Using ILB

The following guidelines describe how to use ILB:

- To administer ILB, you must be able to assume a role that includes the ILB Management rights profile, or become superuser. You can assign the ILB Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.
- To optionally enable auditing of ILB configuration commands, you must enable Oracle Solaris auditing service for the system-wide administration audit class. To do so, see "Configuring the Audit Service (Task Map)" in *System Administration Guide: Security Services*.
- ILB userland components are delivered as separate IPS package in the Oracle Solaris 11 Express repository with package names starting with `SUNWilb`. You must download these packages from the Oracle Solaris 11 Express repository using the `pkg install` command. For instructions on installing ILB, see "Installing the Integrated Load Balancer" on page 567.
- The ILB NAT implementation in stand-alone mode is limited to just the load-balancing functionality.
- ILB provides redundancy only for machine failures and does not handle switch failures. As of now, ILB does not provide synchronization between different machines running ILB.

ILB and the Service Management Facility

ILB is managed by the Service Management Facility (SMF) service `svc:/network/loadbalancer/ilb:default`. For an overview of SMF, see Chapter 11, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*. For step-by-step procedures that are associated with SMF, see Chapter 12, "Managing Services (Tasks)," in *System Administration Guide: Basic Administration*.

ILB Command and Subcommands

You can use `ilbadm` and its subcommands to manipulate the load-balancing rules. For more detailed information about `ilbadm` subcommands, refer to the `ilbadm(1M)` man page.

TABLE 26–2 ILB Commands and Subcommands Used to Manipulate the Load-balancing Rules

ILB Command	Description
`ilbadm create-rule`	Creates a `rule name` with the given characteristics.
`ilbadm show-rule`	Displays characteristics of specified rules or displays all the rules if no rules are specified.
`ilbadm delete-rule`	Removes all information pertaining to a `rule name`. If `name` does not exist, this subcommand fails.
`ilbadm enable-rule`	Enables a named rule, or all the rules if no names are specified.
`ilbadm disable-rule`	Disables a named rule or all the rules if no names are specified.
`ilbadm show-statistics`	Shows statistics. For example, `-t` with this subcommand includes a time stamp with every header.
`ilbadm show-hc-result`	Shows the health check results for the servers that are associated with the specified name of the rule `rule-name`. If `rule-name` is not specified, the health check results of servers for all the rules are displayed.
`ilbadm show-nat`	Displays NAT table information.
`ilbadm create-servergroup`	Creates a server group. Additional servers can be added by using `ilbadm add-server`.
`ilbadm delete-servergroup`	Deletes a server group.
`ilbadm show-servergroup`	Lists a server group or lists all the server groups if no server group is specified.
`ilbadm enable-server`	Enables a disabled server.
`ilbadm disable-server`	Disables the specified servers.
`ilbadm add-server`	Adds the specified servers to server groups.

TABLE 26–2 ILB Commands and Subcommands Used to Manipulate the Load-balancing Rules *(Continued)*

ILB Command	Description
`ilbadm show-server`	Displays servers associated with the named rules or displays all the servers if a rule name is not specified.
`ilbadm remove-server`	Removes servers from a server group.
`ilbadm create-healthcheck`	Sets up health check information that can be used to set up rules.
`ilbadm show-persist`	Displays the session persistence mapping table.
`ilbadm export-config` *filename*	Exports the existing configuration file in a format suitable for importing when needed by using `ilbadm import`. If *filename* is not specified, then `ilbadm export` writes to `stdout`.
`ilbadm import-config -p` *filename*	Imports a file and replaces the existing configuration with the contents of this imported file. If *filename* is not specified, then `ilbadm import` reads from `stdin`.

◆ ◆ ◆ CHAPTER 27

Configuration of Integrated Load Balancer Tasks

This chapter describes the installation and configuration of the Integrated Load Balancer (ILB) and contains the following sections:

Installing the Integrated Load Balancer

This section describes the installation of ILB.

ILB has two portions, the kernel and the userland. The kernel portion is automatically installed as a part of the Oracle Solaris 11 Express installation. But to get the userland portion of ILB, the user has to manually install the `ilb` present at `service/network/load-balancer/ilb` package.

Enabling and Disabling ILB

This section describes the procedures to enable and disable ILB.

▼ How to Enable ILB

Before You Begin Make sure that the system's Role Based Access Control (RBAC) attribute files have the following entries (if the entries are not there, add them manually):

- File name: `/etc/security/auth_attr`
  - `solaris.network.ilb.config:::Network ILB Configuration::help=NetworkILBconf.html`
  - `solaris.network.ilb.enable:::Network ILB Enable Configuration::help=NetworkILBenable.html`
  - `solaris.smf.manage.ilb:::Manage Integrated Load Balancer Service States::help=SmfILBStates.html`
- File name: `/etc/security/prof_attr`
  - `Network ILB:::Manage ILB configuration via ilbadm:auths=solaris.network.ilb.config,solaris.network.ilb.enable;help=RtNetILB.htm`
  - The Network Management entry in the file should include `solaris.smf.manage.ilb`.
- File name:`/etc/user_attr`
  - `daemon::::auths=solaris.smf.manage.ilb,solaris.smf.modify.application`

1 Assume a role that includes the ILB Management rights profile to a role that you create, or become superuser.

You can assign the ILB Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 Enable the appropriate forwarding service either IPv4 or IPv6 or both of them.

```
#svcadm enable svc:/network/ipv4-forwarding
# svcadm enable svc:/network/ipv6-forwarding
```

3 Enable the ILB service.

```
# svcadm enable ilb
```

4 Verify that the ILB service is enabled.

```
# svcs ilb
```

▼ How to Disable ILB

1. **Assume a role that includes the ILB Management rights profile to a role that you create, or become superuser.**

   You can assign the ILB Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring and Using RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Disable the ILB service.**

   ```
   # svcadm disable ilb
   ```

3. **Verify that the ILB service is disabled.**

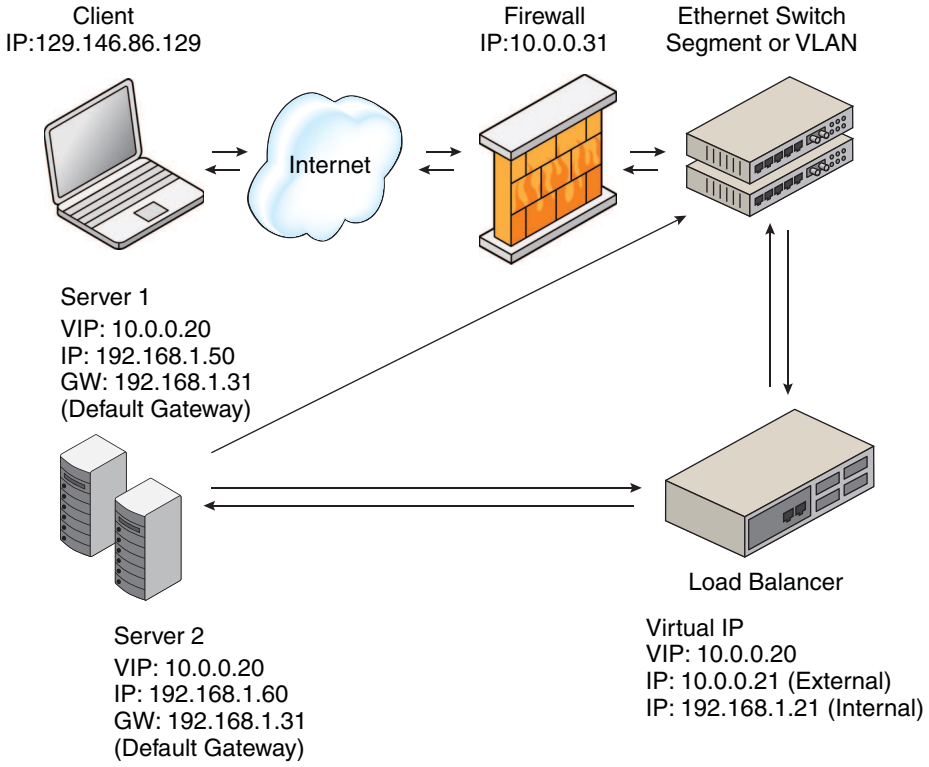
   ```
   # svcs ilb
   ```

Configuring ILB

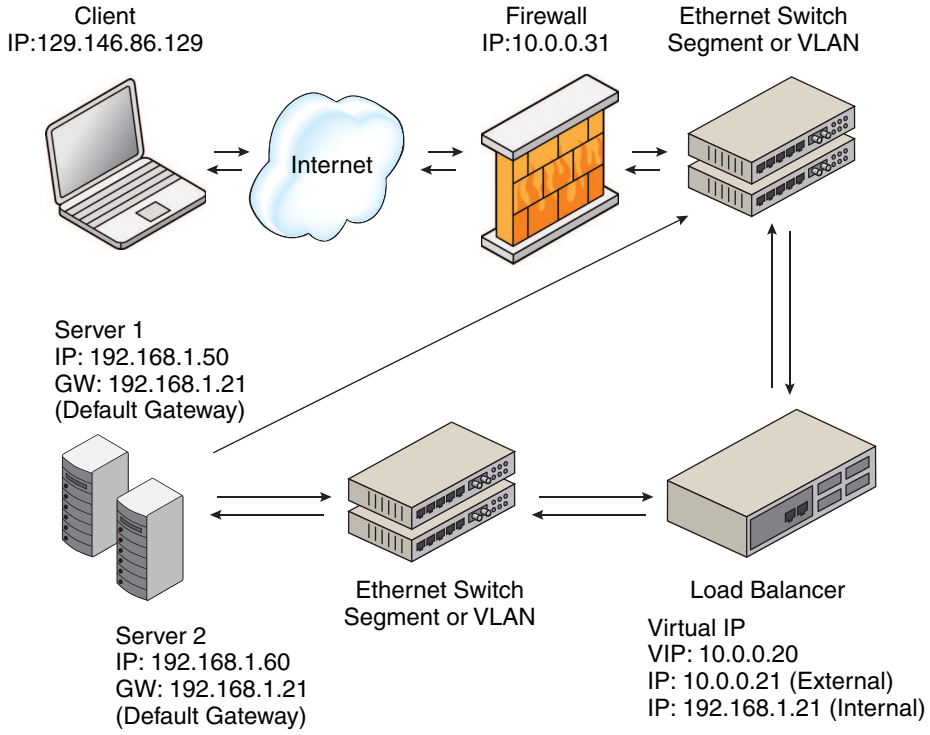
This section describes the implementation of ILB with DSR, half-NAT, and full-NAT topologies.

DSR, Full-NAT, and Half-NAT Topologies

The following figure shows the implementation of ILB using the DSR topology.



ILB operates in both the half-NAT and full-NAT modes. The general implementation of the NAT topology is as shown in the following figure.



Half-NAT Load-Balancing Topology

In the half-NAT mode of ILB operation, ILB rewrites only the destination IP address in the header of the packets. If you are using the half-NAT implementation, you cannot connect to a virtual IP (VIP) address of the service from the same subnet on which the server resides.

TABLE 27–1 Request Flow and Response Flow for the Half-NAT Implementation

	Request Flow	Source IP Address	Destination IP Address
1.	Client —> Load Balancer	Client	VIP of Load Balancer
2.	Load Balancer —> Server	Client	Server
	Response Flow		
3.	Server —> Load Balancer	Server	Client
4.	Load Balancer —> Client	VIP of Load Balancer	Client

If you connect the client PC to the same network as that of the servers, the intended server responds directly to the client. The fourth step does not occur and hence the source IP address

for the server response to the client is invalid. When the client sends a connection request to the load balancer, the response occurs from the intended server. Henceforth, the client's IP stack correctly drops all the responses.

In that case, the request flow and response flow proceed as shown in the following table.

TABLE 27–2 Request Flow and Response Flow for the Half-NAT Implementation

	Request Flow	Source IP Address	Destination IP Address
1.	Client —> Load Balancer	Client	VIP of Load Balancer
2.	Load Balancer —> Server	Client	Server
	Response Flow		
3.	Server —> Client	Server	Client

Full-NAT Load-Balancing Topology

In the full NAT implementation, the source and destination IP addresses are rewritten to ensure that the traffic goes through the load balancer in both directions. The full NAT topology makes it possible to connect to the VIP from the same subnet that the servers are on. The following table depicts the full-NAT topology for ILB. There is no default route required through the servers. The default route through the load balancer is the router address on subnet C. In this scenario, the load balancer behaves as a proxy.

TABLE 27–3 Request Flow and Response Flow for the Full-NAT Implementation

	Request Flow	Source IP Address	Destination IP Address
1.	Client —> Load Balancer	Client	VIP of Load Balancer
2.	Load Balancer —> Server	Interface address of the load balancer (subnet C)	Server
	Response Flow		
3.	Server —> Load Balancer	Server	Interface address of the load balancer (subnet C)
4.	Load Balancer —> Client	VIP of Load Balancer	Client

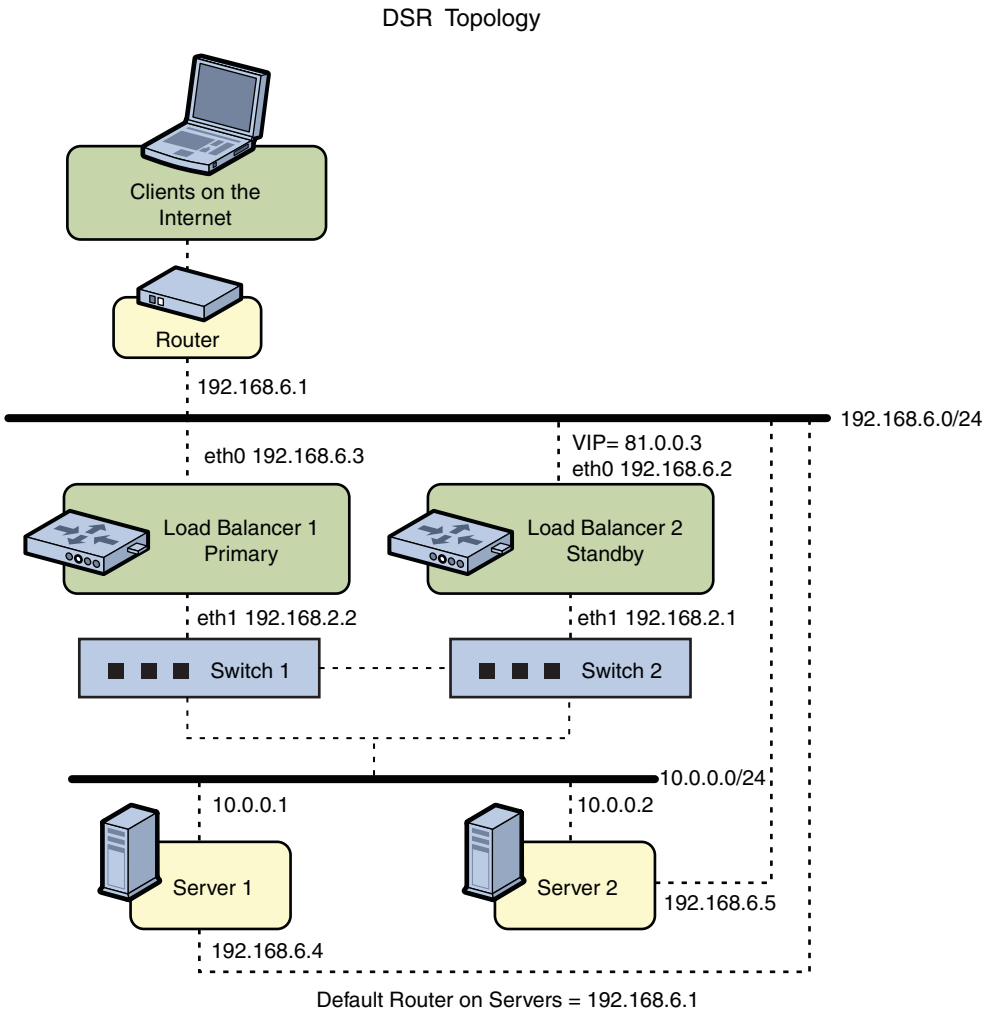
ILB High-Availability Configuration (Active-Passive Mode Only)

This section describes the high availability configuration of ILB using the DSR, half-NAT, and full-NAT topologies.

ILB HA Configuration Using the DSR Topology

This section describes how to set up the ILB connections to achieve high availability (HA) by using the DSR topology. You need to set up two load balancers, one as the primary load balancer and the other as the standby load balancer. If the primary load balancer fails, the standby load balancer assumes the role of the primary load balancer.

The following figure shows the DSR topology for configuring the ILB connections to achieve HA.



All VIPs on Load Balancers are configured on interfaces facing subnet 192.168.6.0/24.

How to Configure ILB to Achieve High-Availability by Using the DSR Topology

1 Configure both the primary and standby load balancers by using the following load balancer commands:

```
# ilbadm create-servergroup -s server=10.0.0.1,10.0.0.2 sg1
# ilbadm create-rule -i vip=81.0.0.3,port=9001 \
-m lbalg=hash-ip-port,type=DSR -o servergroup=sg1 rule1
```

2 Make sure that all server have VIP configured on their `lo0` interface.

```
Server1# ipadm create-addr -T static -d -a 81.0.0.3/24 lo0/server1
Server2# ipadm create-addr -T static -d -a 81.0.0.3/24 lo0/server2
```

3 Configure Load Balancer 1 to serve as the primary load balancer.

```
LB1# dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB1# vrrpadm create-router -V 1 -A inet -l eth0 -p 255 vrrp1
LB1# ipadm create-addr -T static -d -a 81.0.0.3/24 vnicl/lb1
```

4 Configure Load Balancer 2 to act as the stand by load balancer.

```
LB2# dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB2# vrrpadm create-router -V 1 -A inet -l eth0 -p 100 vrrp1
LB2# ipadm create-addr -T static -d -a 81.0.0.3/24 vincl/lb2
```

The preceding configuration provides protection against the following failure scenarios:

- If Load Balancer 1 fails, Load Balancer 2 becomes the primary, takes over address resolution for the VIP `81.0.0.3`, and handles all the packets from clients with the destination IP address `81.0.0.3`.

  When Load Balancer 1 recovers, Load Balancer 2 returns to the standby mode.

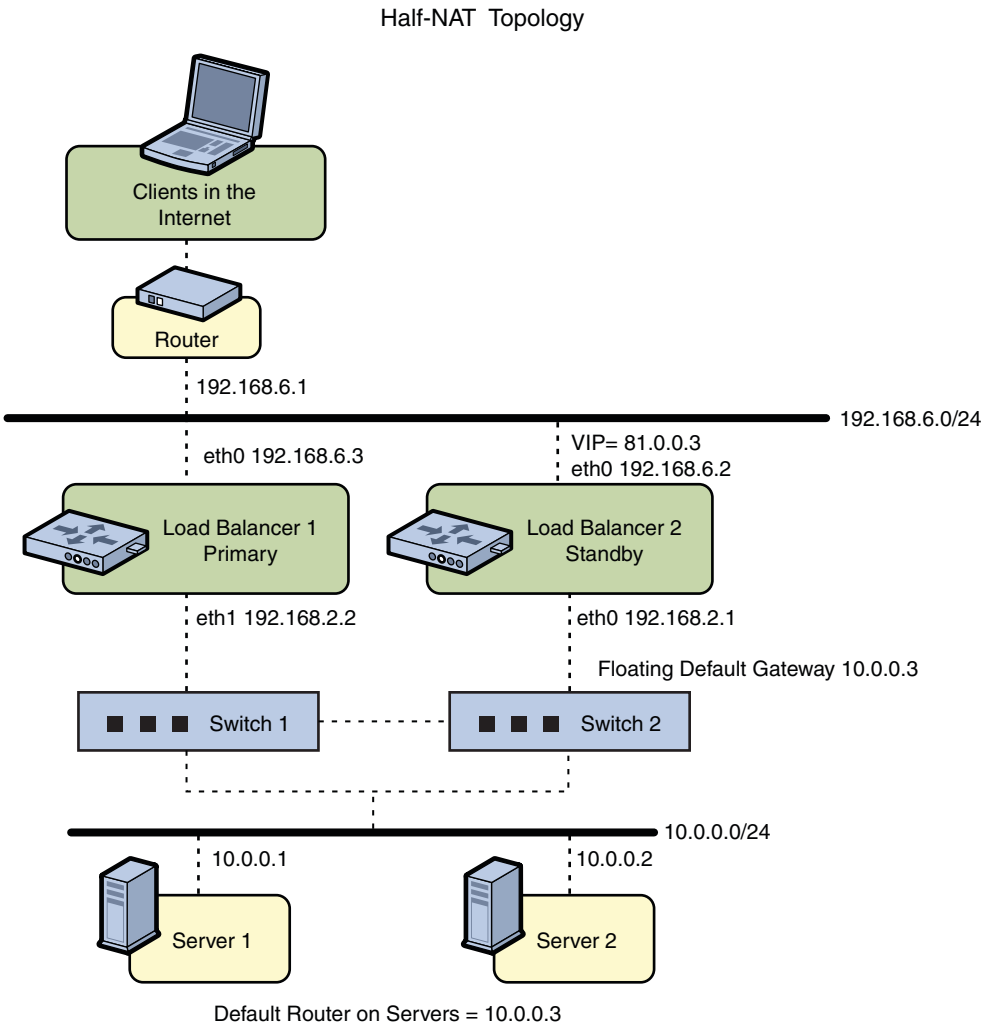
- If one or both of the Load Balancer 1's interfaces fails, Load Balancer 2 takes over as the primary. Thus, the Load Balancer 2 takes over address resolution for VIP `81.0.0.3` and handles all the packets from clients with the destination IP address `81.0.0.3`.

  When both of Load Balancer 1's interfaces are healthy, Load Balancer 2 returns to the standby mode.

ILB High-Availability Configuration by Using the Half-NAT Topology

This section describes how to set up the ILB connections to achieve HA by using the half-NAT topology. You need to set up two load balancers, one as the primary and the other as the stand by. If the primary load balancer fails, the standby load balancer assumes the role of the primary load balancer.

The following figure shows the half-NAT topology for configuring the ILB connections to achieve HA.



All VIPs on Load Balancers are configured on interfaces facing subnet 192.168.6.0/24.

▼ How to Configure ILB to Achieve High-Availability by Using the Half-NAT Topology

1 Configure both the primary and standby load balancers.

```
# ilbadm create servergroup -s server=10.0.0.1,10.0.0.2 sg1
# ilbadm create-rule -ep -i vip=81.0.0.3,port=9001-9006,protocol=udp \
-m lbalg=roundrobin,type=HALF-NAT,pmask=24 \
-h hc-name=hc1,hc-port=9006 \
-t conn-drain=70,nat-timeout=70,persist-timeout=70 -o servergroup=sg1 rule1
```

2 Configure Load Balancer 1 to serve as the primary load balancer.

```
LB1# dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB1# ipadm create-addr -T static -d -a 81.0.0.3/24 vnic1/lb1
LB1# vrrpadm create-router -V 1 -A inet -l eth0 -p 255 vrrp1
LB1# dladm create-vnic -m vrrp -V 2 -A inet -l eth1 vnic2
LB1# ipadm create-addr -T static -d -a 10.0.0.3/24 vnic2/lb1
LB1# vrrpadm create-router -V 2 -A inet -l eth1 -p 255 vrrp2
```

3 Configure the Load Balancer 2 to serve as the stand by load balancer.

```
LB2# dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB2# ipadm create-addr -T static -d -a 81.0.0.3/24 vnic1/lb2
LB2# vrrpadm create-router -V 1 -A inet -l eth0 -p 100 vrrp1
LB2# dladm create-vnic -m vrrp -V 2 -A inet -l eth1 vnic2
LB2# ipadm create-addr -T static -d -a 10.0.0.3/24 vnic2/lb2
LB2# vrrpadm create-router -V 2 -A inet -l eth1 -p 100 vrrp2
```

4 Add the IP address for the floating default gateway to both servers.

```
# route add net 192.168.6.0/24 10.0.0.3
```

The preceding configuration provides protection against the following failure scenarios:

- If Load Balancer 1 fails, Load Balancer 2 will become the primary and take over address resolution for the VIP `81.0.0.3` and handle all the packets from clients with the destination IP address `81.0.0.3`. It should also handle all the packets that are sent to the floating gateway address `10.0.0.3`.

  When Load Balancer 1 recovers, Load Balancer 2 will return to the standby mode.

- If one or both of Load Balancer 1's interfaces fails, Load Balancer 2 will take over as primary. Thus Load Balancer 2 takes over address resolution for VIP `81.0.0.3` and handles all packets from clients with the destination IP address `81.0.0.3`. It should also handle all the packets destined to the floating gateway address `10.0.0.3`.

  When both Load Balancer 1's interfaces are healthy, Load Balancer 2 returns to the standby mode.

Note – The current implementation of ILB does not synchronize primary and standby load balancers. When the primary load balancer fails and the standby load balancer takes over, the existing connections will fail. However, HA without synchronization is still valuable under circumstances when the primary load balancer fails.

Setting Up User Authorization for ILB Configuration Subcommands

You must have the `solaris.network.ilb.config` RBAC authorization to execute the following ILB configuration subcommands:

```
create-servergroup
delete-servergroup groupname
show-servergroup
add-server
remove-server
enable-server
disable-server
show-server
create-healthcheck
show-healthcheck
delete-healthcheck
show-rule
delete-rule
enable-rule
disable-rule
show-statistics
show-hc-result
show-nat
show-persist
export-config
import-config
```

To assign the authorization to an existing user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*

You can also provide the authorization when creating a new user account on the system. For example:

```
useradd -g 10 -u 1210 -A solaris.network.ilb.config ilbadmin
```

The `useradd` command adds a new user to the `/etc/passwd`, `/etc/shadow`, and `/etc/user_attr` files. The -A option assigns the authorization to the user.

Administering ILB Server Groups

You can use the `ilbadm` command to create, delete, and list ILB server groups. For the definition of a server group, see "ILB Terminology" on page 556.

How to Create a Server Group

1. **Select a name for the server group that you are about to create.**
2. **Select the servers that are to be included in the server group.**
   Servers can be specified by their host name or IP address and optional port.
3. **Create the server group.**
   ```
   # ilbadm create-servergroup -s servers=webserv1,webserv2,webserv3 webgroup
   ```

Example 27–1 Creating a Server Group

The following example creates a server group called `webgroup` consisting of three servers:

```
# ilbadm create-servergroup -s servers=webserv1,webserv2,webserv3 webgroup
```

How to Delete a Server Group

1. **Select the server group that you want to remove.**
   The server group must not be in use by an active rule. Otherwise, the deletion will fail.
2. **In the terminal window, delete the server group.**
   ```
   # ilbadm delete-servergroup webgroup
   ```

Example 27–2 Deleting a Server Group

The following example removes the server group called `webgroup`:

```
# ilbadm delete-servergroup webgroup
```

Displaying a Server Group

In a terminal window, type the `show-servergroup` subcommand to obtain information about a specific server group or all server groups.

The following example lists detailed information about all the server groups:

```
# ilbadm show-servergroup -o all
```

sgname	serverID	minport	maxport	IP_address
sg1	sg1.2	80	80	35.0.0.4
sg1	sg1.1	80	80	35.0.0.3
sg1	sg1.0	80	80	35.0.0.2
sg2	sg2.3	81	81	35.0.0.5
sg2	sg2.3	81	81	35.0.0.4
sg2	sg2.1	81	81	35.0.0.3
sg2	sg2.0	81	81	35.0.0.2

Administering Back-End Servers in ILB

You can use the `ilbadm` to add, remove, enable, and disable one or more back-end servers within server groups. For a list of definitions, see "ILB Terminology" on page 556.

▼ How to Add a Back-End Server to a Server Group

- **Add a back-end server to a server group.**

  Server specifications must include a host name or IP address and can also include an optional port or a range of ports. Server entries with the same IP address are disallowed within a server group.

  ```
  # ilbadm add-server -e -s server=192.168.89.1,192.168.89.2 ftpgroup
  # ilbadm add-server -e -s server=[2001:7::feed:6]:8080 sgrp
  ```

  The `-e` option enables the servers in addition to adding them to the group.

Note – IPv6 addresses must be enclosed in square brackets.

Example 27–3 Adding a Back-End Server to a Server Group

The following example adds servers to server groups `ftpgroup` and `sgrp`, and enables them.

```
# ilbadm add-server -e -s \
server=192.168.89.1,192.168.89.2 ftpgroup
# ilbadm add-server -e -s server=[2001:7::feed:6]:8080 sgrp
```

▼ How to Remove a Back-End Server From a Server Group

1. **To remove a server from a specific server group, follow these steps:**

   a. **Identify the server ID of the server that you want to remove from a server group. The server ID can be obtained from the output of `show-servergroup -o all` subcommand.**

   b. **Remove the server.**

      `# ilbadm remove-server -s serverID=`*10.1.1.2 websg*

2. **To remove a server from all server groups, follow these steps given below:**

   a. **Identify the IP address and the host name of the server you want to remove.**

   b. **Use the output of the `ilbadm show-servergroup-o all` command to identify the server groups that include the server.**

   c. **For each server group, run the following subcommand to remove the server from the server group.**

Example 27–4 Removing a Back-Server From a Server Group

The following example removes the server with server ID `10.1.1.2` from server group `websg`:

```
# ilbadm remove-server -s serverID=10.1.1.2 websg
```

Note the following:

- If the server is being used by a NAT or half-NAT rule, disable the server by using the `disable-server` subcommand before removal. When a server is disabled, it enters the connection-draining state. After all the connections are drained, the server can be removed by using the `remove-server` subcommand. After issuing the `disable-server` command, periodically check the NAT table (by using the `show-nat` command) to see if the server in question still has connections. After all of the connections are drained (the server does not get displayed in the `show-nat` command output), the server can then be removed by using the `remove-server` command.
- If the conn-drain timeout value is set, the connection-draining state will be completed upon conclusion of the timeout period. The default value of conn-drain timeout is 0, meaning it will keep waiting until a connection is gracefully shut down.

▼ How to Re-enable or Disable a Back-End Server

1 **Identify the IP address, host name, or server ID of the server you want to re-enable or disable. If a IP address or host name is specified, the server will be re-enabled or disabled for the all rules associated with it. If a server ID is specified, the server will be re-enabled or disabled for the specific rules that are associated to the server ID.**

Note – A server can have multiple server IDs, if it belongs to multiple server groups.

---

2 **Re-enable or disable a server.**

```
# ilbadm enable-server serverID=websg.1
# ilbadm disable-server s serverID=websg.1
```

Example 27–5 Re-enabling and Disabling a Back-End Server

In the following example a server with server ID `websg.1` is enabled and then disabled.

```
# ilbadm enable-server serverID=websg.1
# ilbadm disable-server s serverID=websg.1
```

Administering Health Checks in ILB

ILB provides the following optional types of server health checks for the user to select from:

- Built-in ping probes
- Built-in TCP probes
- Built-in UDP probes
- User-supplied tests that can run as health checks

By default, ILB does not perform any health checks. You can specify health checks for each server group when creating a load-balancing rule. You can configure only one health check per load-balancing rule. As long as a virtual service is enabled, the health checks on the server group that is associated with the enabled virtual service start automatically and repeat periodically. The health checks stop as soon as the virtual service is disabled. The previous health check states are not preserved when the virtual service is re-enabled.

When you specify a TCP, UDP, or custom test probe for running a health check, ILB sends a ping probe, by default, to determine if the server is reachable before it sends the specified TCP, UDP, or custom test probe to the server. The ping probe is a method of monitoring server health. If the ping probe fails, the corresponding server is disabled with the health check status of `unreachable`. If the ping probe succeeds, but the TCP, UDP, or custom test probe fails, the server is disabled with the health check status of `dead`.

You can use the `ilbadm` command to create, delete, and list the health checks. For a list of definitions, see "ILB Terminology" on page 556.

Creating a Health Check

In the following example, two health checks *objects,hc1* and *hc-myscript*, are created. The first health check uses the built-in TCP probe. The second health check uses a custom test, `/var/tmp/my-script`.

```
# ilbadm create-healthcheck \
-h hc-timeout=3,hc-count=2,hc-interval=8,hc-test=tcp hc1
# ilbadm create-healthcheck \
-h hc-timeout=3,hc-count=2,hc-interval=8,hc-test=/var/tmp/my-script hc-myscript
```

`hc-test` specifies the type of health check.

`hc-interval` specifies the interval between consecutive health checks. To avoid synchronization, the actual interval is randomized between `0.5 * hc-interval` and `1.5 * hc-interval`.

`hc-timeout` specifies the timeout when the health check considered to have failed if it does not complete.

`hc-count` specifies the number of attempts to run the `hc-test` health check.

Note – The port specification for `hc-test` is specified with the `hc-port` keyword in the `create-rule` subcommand. For details, refer to `ilbadm(1M)` man page.

User-Supplied Test Details

The following criteria must be met by the user-supplied test:

- The test can be a binary or a script.
- The test can reside anywhere on the system, and you must specify the absolute path when using the `create-healthcheck` subcommand.

  When you specify the test (for example, `/var/tmp/my-script`) as part of the health check specification in the `create-rule` subcommand, the `ilbd` daemon forks a process and executes the test, as follows:

  ```
  /var/tmp/my-script $1 $2 $3 $4 $5
  ```

  A description of the arguments is as follows:

  `$1` VIP (literal IPv4 or IPv6 address)

  `$2` Server IP (literal IPv4 or IPv6 address)

  `$3` Protocol (UDP, TCP as a string)

  `$4` Numeric port range (the user-specified value for `hc-port`)

`$5` maximum time (in seconds) that the test should wait before returning a failure. If the test runs beyond the specified time, it might be stopped, and the test would be considered failed. This value is user-defined and specified in`hc-timeout`.

The user-supplied test, *my-script* , might or might not use all the arguments, but it *must* return one of the following :

- Round Trip Time (RTT) in microseconds
- 0 if the test does not calculate RTT
- -1 for failure

By default, the health check test runs with the following privileges: `PRIV_PROC_FORK`, `RIV_PROC_EXEC`, `RIV_NET_ICMPACCESS`.

If a broader privilege set is required, you must implement `setuid` in the test. For more details on the privileges, refer to the `privileges(5)` man page.

Deleting a Health Check

The following example deletes a health check called *hc1*:

```
# ilbadm destroy-healthcheck hc1
```

Listing Health Checks

You can use the `list-healthcheck` subcommand to obtain detailed information about configured health checks. The following example lists two configured health checks:

```
# ilbadm list-healthcheck
```

NAME	TIMEOUT	COUNT	INTERVAL	DEF_PING	TEST
hc1	3	2	8	Y	tcp
hc2	3	2	8	N	/var/usr-script

Displaying Health Check Results

You can use the `list-hc-result` subcommand to obtain health check results. If a rule or a health check is not specified, the subcommand lists all the health checks.

The following example displays the health check results associated with a rule called `rule1`:

```
# ilbadm list-hc-result rule1
```

RULE	HC	SERVERID	TEST	STATUS	FAIL	LAST	NEXT
rule1	hc1	sg1:0	tcp	server-alive	3	11:23:30	11:23:40
rule1	hc1	sg1:1	tcp	server-dead	4	11:23:30	11:23:40

Administering ILB Rules

You can use `ilbadm` to create, delete, and list the load-balancing rules. For definition of a load-balancing rule and the parameters needed to create a rule, see "ILB Terminology" on page 556.

How to Create a Rule

1. **Create a server group that includes the appropriate back-end servers.**

   ```
   # ilbadm create-servergroup -s server=60.0.0.10:6000-6009,60.0.0.11:7000-7009 sg1
   ```

2. **If you want to associate server health checks with a rule, create a health check object.**

   ```
   # ilbadm create-healthcheck -h hc-test=tcp,hc-timeout=2,hc-count=3,hc-interval=10 hc1
   ```

3. **Identify the VIP, port, and optional protocol that are to be associated with the rule.**

4. **Select the operation you want to use (DSR, full-NAT or half-NAT). If NAT is selected, you must specify the IP address range that is to be used as the `proxy-src` address.**

5. **Select the load-balancing algorithm that is to be used.**

6. **Select other optional features (see the `ilbadm(1M)` man page for details).**

7. **Select a rule name.**

8. **Create and enable the rule.**

   ```
   # ilbadm create-rule -e -i vip=81.0.0.10,port=5000-5009,protocol=tcp \
   -m lbalg=rr,type=NAT,proxy-src=60.0.0.101-60.0.0.104,persist=/24
   -h hc-name=hc1 -o servergroup=sg1 rule1
   ```

Example 27–6 Creating a Full-NAT Rule With a Health Check Session Persistence

This example creates a health check called `hc1`, and a server group called `sg1` (consisting of two servers, each with a range of ports). The last command creates and enables a rule called `rule1` of full-NAT mode and associates the rule to the server group and the health check. Note that the creation of the server group and health check must precede the creation of the rule.

```
ilbadm create-healthcheck -h hc-test=tcp,hc-timeout=2,hc-count=3,hc-interval=10 hc1
ilbadm create-servergroup -s server=60.0.0.10:6000-6009,60.0.0.11:7000-7009 sg1
ilbadm create-rule -e -i vip=81.0.0.10,port=5000-5009,protocol=tcp \
-m lbalg=rr,type=NAT,proxy-src=60.0.0.101-60.0.0.104,persist=/24
-h hc-name=hc1 -o servergroup=sg1 rule1
```

When creating a NAT/half NAT rule, it is recommended to specify the value for connection-drain timeout. The default value of conn-drain timeout is 0, meaning it will keep waiting until a connection is gracefully shut down.

Deleting a Rule

To delete a rule, use the `delete-rule` subcommand. If you want to remove all rules, use the `-a` option. The following example deletes the rule called `rule1`:

```
# ilbadm delete-rule rule1
```

Listing Rules

To list the configuration details of a rule, use the `list-rule` subcommand. If no rule name is specified, information is provided for all rules.

```
# ilbadm list-rule
```

Rulename (+ = enabled)	LB-alg	Type	Proto	VIP/port
rule-http +	HIPP	H-NAT	TCP	10.0.0.1/http
rule-dns	HIP	DSR	UDP	10.0.0.1/53
rule-abc	RR	NAT	TCP	2003::1/1024
rule-xyz +	HIPV	NAT	TCP	2003::1/2048-2050

Displaying ILB Statistics

You can use the `ilbadm` command to obtain information such as printing statistics of a server or a rule, or displaying NAT table information and session persistence mapping table. For a list of definitions, see the "ILB Terminology" on page 556.

Obtaining Statistical Information Using the `show-statistics` Subcommand

Use the `show-statistics` subcommand to view load distribution details. The following example shows the usage of the `show-statistics` subcommand:

```
ilbadm show-statics
 PKT_P   BYTES_P   PKT_U   BYTES_U   PKT_D   BYTES_D
  9       636       0       0         0       0
```

where

- `PKT_P`: Packets processed
- `BYTES_P`: Bytes processed
- `PKT_U`: Unprocessed packets
- `BYTES_U`: Unprocessed bytes

Displaying the NAT Connection Table

Use the `show-nat` subcommand to view the NAT connection table. No assumptions should be made about the relative positions of elements in consecutive runs of this command. For example, executing `{{ ilbadm show-nat 10}}` twice is not guaranteed to show the same 10 items twice, especially on a busy system. If a count value is not specified, the entire NAT connection table is displayed.

The following example displays five entries from the NAT connection table:

EXAMPLE 27–7 NAT Connection Table Entries `ilbadm show-nat 5`

```
 UDP: 124.106.235.150.53688 > 85.0.0.1.1024 >>> 82.0.0.39.4127 > 82.0.0.56.1024
UDP: 71.159.95.31.61528 > 85.0.0.1.1024 >>> 82.0.0.39.4146 > 82.0.0.55.1024
UDP: 9.213.106.54.19787 > 85.0.0.1.1024 >>> 82.0.0.40.4114 > 82.0.0.55.1024
UDP: 118.148.25.17.26676 > 85.0.0.1.1024 >>> 82.0.0.40.4112 > 82.0.0.56.1024
UDP: 69.219.132.153.56132 > 85.0.0.1.1024 >>> 82.0.0.39.4134 > 82.0.0.55.1024
```

The format of entries is as follows:

```
 T: IP1 > IP2 >>> IP3 > IP4

T: The transport protocol used in this entry.
IP1: The client's IP address and port.
IP2: The VIP and port.
IP3: If half-NAT mode, the client's IP address and port.
If full-NAT mode, the client's IP address and port.
IP4: The back-end server's IP address and port.
```

Displaying the Session Persistence Mapping Table

Use the `show-persist` subcommand to view the session persistence mapping table.

EXAMPLE 27–8 ilbadm show-persist 5

The following example displays five entries from the table:

```
rule2: 124.106.235.150 --> 82.0.0.56
rule3: 71.159.95.31 --> 82.0.0.55
rule3: 9.213.106.54 --> 82.0.0.55
rule1: 118.148.25.17 --> 82.0.0.56
rule2: 69.219.132.153 --> 82.0.0.55
```

The format of entries is as follows:

```
 R: IP1 --> IP2

R: The rule that this persistence entry is tied to.
IP1: The client's IP address.
IP2: The back-end server's IP address.
```

Using Import and Export Subcommands

The export subcommand exports the current configuration to a user-specified file. This information can then be used as input for the import subcommand. The import subcommand deletes the existing configuration before importing unless specifically instructed to retain it. Omission of a file name instructs the command to read from standard input or write to standard output.

To export an ILB configuration, use the export-config command. The following example exports the current configuration into the file, /var/tmp/ilb_config, in a format suitable for importing by using the import subcommand:

```
# ilbadm export-config /var/tmp/ilb_config
```

To import an ILB configuration, use the import-config command. The following example reads configuration contents of the file, /var/tmp/ilb_config, and overrides the existing configuration:

```
# ilbadm import-config /var/tmp/ilb_config
```

◆ ◆ ◆ CHAPTER 28

Virtual Router Redundancy Protocol (Overview)

Virtual Router Redundancy Protocol (VRRP) is an Internet standard protocol specified in Virtual Router Redundancy Protocol Version 3 for IPv4 and IPv6 and is supported in the Oracle Solaris 11 Express operating system (OS) to provide high availability. The Oracle Solaris 11 Express OS provides an administrative tool that configures and manages the VRRP service.

When you set up a network such as a LAN, it is very important to provide a high availability service. One way to increase the reliability of the network is to provide backups of the critical components in the network. Adding components such as routers, switches, and links to the network ensures the continuity of the service across failures. Providing redundancy at the endpoints of a network is a crucial task and it can be done easily with VRRP. Virtual routers can be introduced in the LAN by using VRRP to provide failure recovery for a router.

To know more about the terms used in VRRP, see "VRRP Terminology" on page 590.

This chapter includes the following sections:

VRRP is an election protocol that dynamically assigns the responsibilities of a virtual router to one of the VRRP routers within the LAN. VRRP provides one or more backup routers for a statically configured router on the LAN.

A VRRP router called the master router controls the IPv4 or IPv6 address or addresses that are associated with the virtual router. The virtual router forwards the packets that are sent to the IP address of the master router.

The election process provides dynamic failover while forwarding packets sent to these IP addresses. VRRP eliminates the single point of failure that is inherent in the static default routed environment.

By using the VRRP feature in the Oracle Solaris 11 Express OS, you can have a more highly available default path for the routing process without having to configure the dynamic routing or router discovery protocols on every end-host.

VRRP Terminology

This section describes some terms that are useful to know when you implement VRRP on your systems.

backup router A VRRP instance for a VRID that is active but not in the master state. Any number of backups can exist for a VRID. A backup router is ready to assume the role of the master router if the current master router fails.

master router A VRRP instance that performs the routing function for the virtual router at a given time. Only one master router is active at a time for a given VRID.

virtual IP address An IP address associated with a VRID from which other hosts can use to obtain network service. The VRIP is managed by the VRRP instances belonging to a VRID.

virtual MAC address A predefined MAC address used by VRRP instances while executing in a media, such as Ethernet that uses MAC addressing. A virtual MAC addresses isolates the operation of the virtual router from the real router providing the routing function and is used instead of the real MAC address. A virtual MAC address is derived from the VRID.

virtual router ID (VRID) A unique number used to identify a virtual router. VRIDs must be unique on a given network segment.

VNIC A pseudo network interface that is configured on top of a system's physical network adapter, also called a network interface (NIC) card. A physical interface can have more than one VNIC. VNICs are essential components of network virtualization. For more information, see Part IV, " Network Virtualization and Resource Management," in *System Administration Guide: Network Interfaces and Network Virtualization*.

VRRP instance A program running on a router by using the VRRP implementation. A single VRRP instance can provide VRRP capability for more than one virtual router.

VRRP router A single router image created by the operation of one or more routers that use VRRP.

VRRP Architectural Overview

VRRP Router

VRRP runs on each VRRP router and manages the state of the router. A host can have multiple VRRP routers configured, where each VRRP router belongs to a different virtual router.

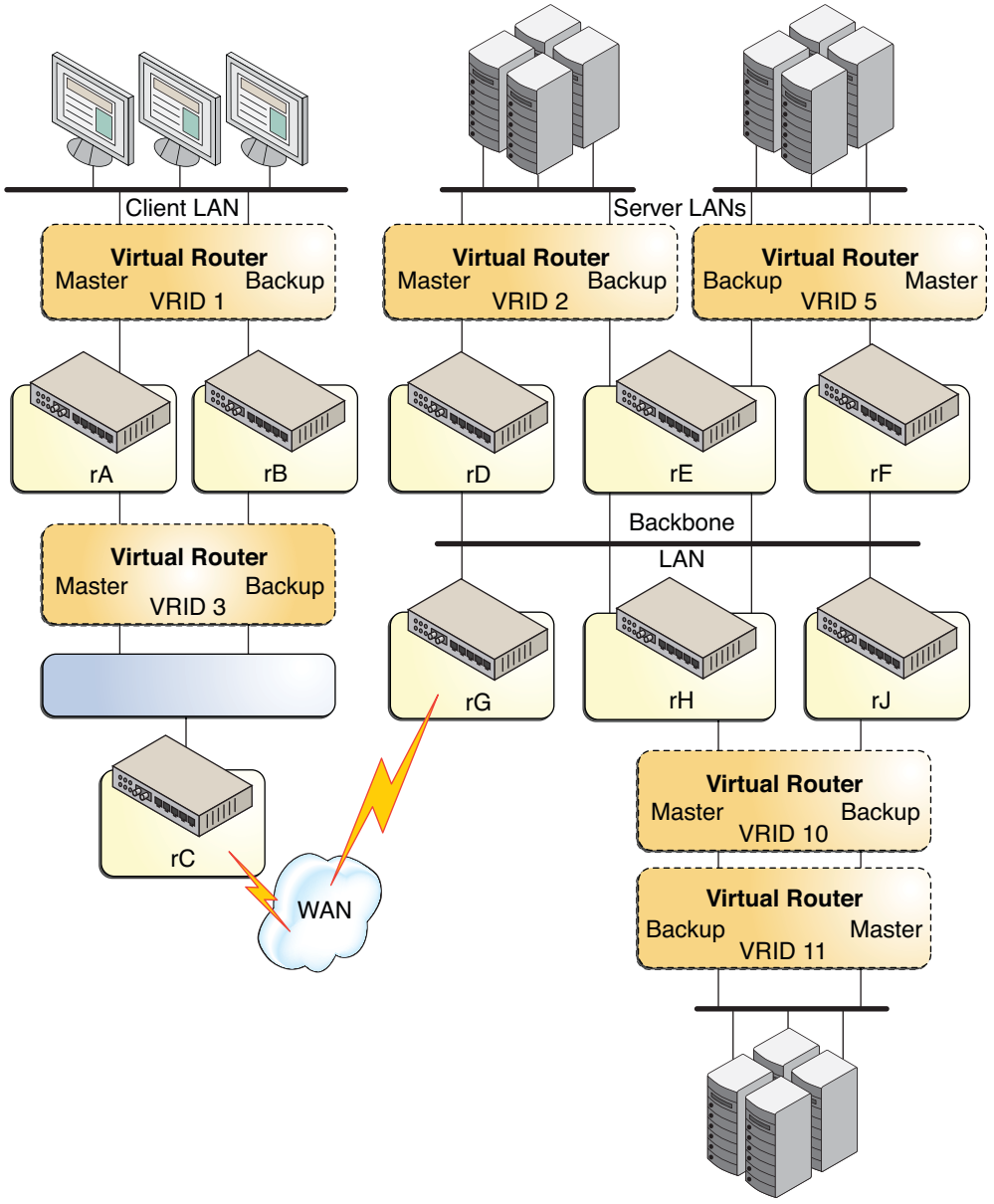
A VRRP router has the following attributes:

- Router name – A system-wide unique identifier

- VRID – Identifies the virtual router within a LAN
- Primary IP address – Used as the source IP address of the VRRP advertisement
- Virtual IP addresses
- VRRP parameters – Includes priority, advertise interval, preempt mode, and accept mode
- VRRP state information and statistics

VRRP Processes

The following figure shows how VRRP works.



As shown in the preceding figure, VRRP works using the following components:

- Router `rA` is the master router for virtual router VRID 1 and the backup router for VRID 3. Router `rA` handles the routing of packets that are addressed to the VIP for VRID 1 and is ready to assume the routing role for VRID 3.

- Router `rB` is the master router for virtual router VRID 3 and the backup router for VRID 1. Router `rB` handles the routing of packets that are addressed to the VIP for VRID 3 and is ready to assume the routing role for VRID 1.
- Router `rC` does not have VRRP functions, but uses the VIP for VRID 3 to reach the client LAN subnet.
- Router `rD` is the master router for VRID 2. Router `rF` is the master router for VRID 5. Router `rE` is the backup router for both of these VRIDs. If `rD` or `rF` fails, `rE` becomes the master router for that VRID. Both `rD` and `rF` could fail at the same time. The fact that a VRRP router is a master router for one VRID does not preclude it from being a master router for another VRID.
- Router `rG` is the WAN gateway for the Backbone LAN. All of the routers attached to the backbone are sharing routing information with the routers on the WAN by using a dynamic routing protocol such as Open Shortest Path First (OSPF). VRRP is not involved in this, although router `rC` advertises that the path to the client LAN subnet is through the VIP of VRID 3.
- Router `rH` is the master router for VRID 10, and the backup router for VRID 11. Likewise, router `rJ` is them master router for VRID 11 and the backup router for VRID 10. This VRRP load-sharing configuration illustrates that multiple VRIDs can exist on a single router interface.

VRRP can be used as a part of a network design that provides almost total routing redundancy for all systems on the network.

VRRP Limitations

Exclusive-IP Zone Support

In each exclusive-IP zone, the VRRP service `svc:/network/vrrp/default` is enabled automatically when any VRRP router is created in the particular zone. The VRRP service manages the VRRP router for that specific zone.

However, the support for an exclusive-IP zone is limited because of the following reasons:

- VNIC cannot be created inside a non-global zone. Therefore, create the VRRP VNIC in the global-zone first, and then assign the VNIC to the non-global zone where the VRRP router resides. The VRRP router can then be created and started in the non-global zone by using the `vrrpadm` command.
- On a single Oracle Solaris 11 Express system, it is not possible to create two VRRP routers in different zones to participate with the same virtual router. The reason is that Oracle Solaris 11 Express does not allow you to create two VNICs with the same MAC address.

Inter-operations With Other Network Features

The VRRP service cannot work on an IP Network Multipathing (IPMP) interface. The reason is because VRRP requires specific VRRP MAC addresses while IPMP works completely in the IP layer.

Further, the VRRP virtual IP addresses can only be statically configured and cannot be auto-configured by the two existing auto-configuration tools for IP addresses: `in.ndpd` for IPv6 auto-configuration and `dhcpagent` for DHCP configuration. Because the master and the backup VRRP routers (VNICs) share the same MAC address, `in.ndpd` and `dhcpagent` can become confused. Eventually unexpected results can occur. Therefore, IPv6 auto-configuration and DHCP configurations are not supported over VRRP VNICs. If you configure either IPv6 auto-configuration or DHCP over a VRRP VNIC, the attempt to bring up the auto-configured IP address fails, as will the auto-configuration operation.

◆ ◆ ◆ CHAPTER 29

VRRP Configuration (Tasks)

A VRRP router executes VRRP and works with other VRRP routers participating with the same virtual router. VRRP has a set of virtual IP addresses.

This chapter describes the following sections:

Within a LAN, each virtual router is uniquely identified by the VRID, address family and is associated with a set of protected virtual IP addresses.

Each participating VRRP router has additional parameters such as priority, advertisement interval, and accept mode. At one time, only one VRRP router (the Master) will assume the responsibility of the virtual router and forward the packets sent to the virtual IP addresses.

Whenever the master fails, the other participating VRRP routers will detect its absence and another VRRP router will be elected as the master and assume the responsibility.

All the VRRP routers with the same virtual router share the same VRRP virtual MAC address. The virtual MAC address is calculated based on the address family and the VRID of the virtual router (in hexadecimal format in Internet standard bit-order). For example:

```
IPv4: 00-00-5E-00-01-{VRID}

IPv6: 00-00-5E-00-02-{VRID}
```

Therefore, a special VRRP VNIC with the virtual MAC address must first be created in order for the VRRP router to work properly. All the IP addresses residing on this VNIC are regarded as virtual IP addresses protected by the VRRP router. Those virtual IP addresses reside in the backup router and are brought up when the router becomes the master router, thus providing high availability for these virtual IP addresses.

VRRP VNIC Creation

The existing `dladm create-vnic` subcommand has been extended to enable you create the VRRP VNIC. The syntax is as follows:

```
# dladm create-vnic [-t] [-R root-dir] [-l link] [-m vrrp -V VRID -A
{inet | inet6}] [-v vlan-id] [-p prop=value[,...]] vnic-link
```

A new VNIC address type, `vrrp` has been introduced. You must specify the VRID and address family with this new VNIC address type.

As a result, a VNIC with a well-known virtual router MAC address will be created.

`vrrpadm` Configuration

The following sections summarize the `vrrpadm` subcommands. See the `vrrpadm(1M)` man page for details. All the subcommands are persistent except for the `vrrpadm show-router` subcommand. For example, the VRRP router created by `vrrpadm create-router` will persist across reboot.

`vrrpadm create-router` subcommand

The `vrrpadm create-router` subcommand creates a VRRP router of the specified VRID and address family with the given parameters. Each VRRP router requires a special VRRP VNIC to be created, and the VNIC can be created by using the `dladm create-vnic` command. For more information, see `vrrpadm(1M)` man page. The syntax is as follows:

```
# vrrpadm create-router -V vrid -l link -A {inet | inet6} [-p \
priority] [-i adv-interval] [-o flags]router-name
```

The `-o` option is used to configure the preempt and accept modes of the VRRP router. Values can be: `preempt`, `un_preempt`, `accept`, `no_accept`. By default , both modes are set to `true`.

The *router-name* is used as the unique identifier this VRRP router and is used in the other `vrrpadm` subcommands. The permitted characters in a router name are: alphanumeric (a-z, A-Z, 0-9) and underscore ('_') . The maximum length of a router name is 31 characters.

`vrrpadm modify-router` subcommand

The `vrrpadm modify-router` subcommand changes the configuration of a specified VRRP router. The syntax is as follows:

```
# vrrpadm modify-router [-p priority] [-i adv-interval] [-o flags] \
router-name
```

vrrpadm delete-router subcommand

The vrrpadm delete-router subcommand deletes a specified VRRP router. The syntax is as follows:

```
# vrrpadm delete-router router-name
```

vrrpadm disable-router subcommand

A VRRP router does not function until it is enabled. By default, a VRRP router is enabled when it is first created. However at times, it is useful to temporarily disable a VRRP router so that you can make configuration changes and then re-enable the router again. The syntax is as follows:

```
# vrrpadm disable-router router-name
```

vrrpadm enable-router subcommand

A disabled VRRP router can be re-enabled by using the enable-router subcommand. The underlying datalink that the VRRP router is created over (specified with the -l option when the router is created with vrrpadm create-router) and the router's VRRP VNIC must exist when the router is enabled. Otherwise, the enable operation fails. The syntax is as follows:

```
# vrrpadm enable-router router-name
```

vrrpadm show-router subcommand

The vrrpadm show-router subcommand shows the configuration and status of a specified VRRP router. See more details in the vrrpadm(1M) man page. The syntax is as follows:

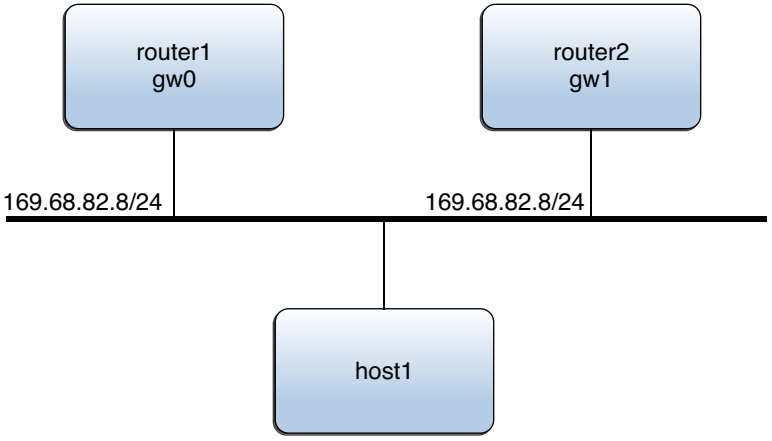
```
# vrrpadm show-router [-P | -x] [-p] [-o field[,...]] [router-name]
```

The following are examples of the vrrpadm show-router output:

```
# vrrpadm show-router vrrp1
NAME VRID LINK AF PRIO ADV_INTV MODE STATE VNIC
vrrp1 1 bge1 IPv4 100 1000 e-pa- BACK vnic1

# vrrpadm show-router -x vrrp1
NAME STATE PRV_STAT STAT_LAST VNIC PRIMARY_IP VIRTUAL_IPS
vrrp1 BACK MAST 1m17s vnic1 10.0.0.100 10.0.0.1

# vrrpadm show-router -P vrrp1
NAME PEER P_PRIO P_INTV P_ADV_LAST M_DOWN_INTV
vrrp1 10.0.0.123 120 1000 0.313s 3609
```

EXAMPLE 29–1 VRRP Configuration Example

The following figure shows a typical VRRP configuration.



In this example, the IP address `169.68.82.8` is configured as the default gateway for `host1`. This IP address is the virtual IP address that is protected by the virtual router that consists of two VRRP routers: `router1` and `router2`. At one time, only one of the two routers serves as the master router and assumes the responsibilities of the virtual router and forwards packets that come from `host1`.

Assume that the VRID of the virtual router is `12`, the following shows the steps that are used to configure the preceding VRRP configuration on `router1` and `router2`. `router1` is the owner of the virtual IP address `169.68.82.8` and its priority is the default value (`255`). `router2` is the backup whose priority is `100`.

```
router1:
# dladm create-vnic -m vrrp -V 12 -A inet -l gw0 vnic1
# vrrpadm create-router -V 12 -A inet -l gw0 vrrp1
# ipadm create-addr -T static -d -a 169.68.82.8/24 vnic1/router1
# ipadm create-addr -T static -d -a 169.68.82.100/24 gw0/router1
# vrrpadm show-router -x vrrp1
NAME STATE PRV_STAT STAT_LAST VNIC PRIMARY_IP VIRTUAL_IPS
vrrp1 MAST BACK 1m17s vnic1 169.68.82.100 169.68.82.8
router2:
# dladm create-vnic -m vrrp -V 12 -A inet -l gw1 vnic1
# vrrpadm create-router -V 12 -A inet -l gw1 -p 100 vrrp1
# ipadm create-addr -T static -d -a 169.68.82.8/24 vnic1/router2
# ipadm create-addr -T static -d -a 169.68.82.101/24 gw0/router2
# vrrpadm show-router -x vrrp1
NAME STATE PRV_STAT STAT_LAST VNIC PRIMARY_IP VIRTUAL_IPS
vrrp1 BACK INIT 2m32s vnic1 169.68.82.101 169.68.82.8
```

EXAMPLE 29–1 VRRP Configuration Example *(Continued)*

Using the configuration of `router1` as an example, you must configure at least one IP address over `gw0`. In the following example, this IP address of router1 is the primary IP address, which is used to send the VRRP advertisement packets:

```
# vrrpadm show-router -x vrrp1
NAME STATE PRV_STAT STAT_LAST VNIC PRIMARY_IP VIRTUAL_IPS
vrrp1 MAST BACK 1m17s vnic1 169.68.82.100 169.68.82.8
```

Security Considerations

A new `solaris.network.vrrp` authorization has been introduced and is required to configure the VRRP service. Note that the read-only operation - `vrrpadm showrouter` does not require this authorization.

The `solaris.network.vrrp` authorization has been added to the Network Management profile.

◆ ◆ ◆ CHAPTER 30

Implementing Congestion Control

This chapter discusses how congestion control is implemented in Oracle Solaris. Controls are set to prevent congestion of TCP and SCTP traffic.

Network Congestion and Congestion Control

Network congestion occurs typically in the form of router buffer overflows, when nodes send more packets than the network can accommodate. Various algorithms have been developed that prevent of traffic congestion by setting controls on the sending systems. These algorithms are supported in Oracle Solaris and can be directly plugged in to the operating system.

The following table lists the supported algorithms with brief descriptions of each.

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

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

- `cong_enabled` - contains a list of algorithms, separated by commas, that are currently operational in the system. You can add or remove algorithms to enable only those algorithms you want to use.
- `cong_default` - the algorithm that is used by default when applications do not specify the algorithms explicitly in socket options. Currently, the setting of the `cong_default` property applies to both global and non-global zones.

To set these properties, you use the `ipadm set-prop` command. You use either the `+=` modifier to add algorithms or the `-=` modifier to remove algorithms.

How to Implement TCP and SCTP Network Congestion Control

1 Become an administrator.

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

2 Display the current settings of the protocol's congestion-control properties.

```
# ipadm show-prop -p cong_enabled,cong_default tcp
```

If you do not specify the properties, then all of the protocol properties will be displayed.

The command displays both the current settings as well as the possible algorithms that you can assign to the properties.

3 Set the congestion control properties of the protocol.

```
# ipadm set-prop -p cong-ctrl-property+=algorithm tcp
```

where

cong-ctrl-property refers to either the `cong_enabled` property or the `cong_default` property.

algorithm specifies the algorithm that you are setting for the property. You can specify any algorithm that is listed under the `POSSIBLE` field heading in the output of the `ipadm show-prop` command.

4 (Optional) Remove an algorithm that is currently enabled.

```
# ipadm set-prop -p cong-ctrl-property-=algorithm tcp
```

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

5 (Optional) Display the new settings of the congestion control properties.

```
# ipadm show-prop -p cong_enabled,cong_default tcp
```

Example 30–1 Setting Algorithms for Congestion Control

This example changes the default algorithm of the TCP protocol from `newreno` to `cubic`. It also removes `vegas` from the list of enabled algorithms.

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

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

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

PART V

IP Quality of Service (IPQoS)

This part contains tasks and information about IP Quality of Service (IPQoS), Oracle Solaris's implementation of differentiated services.

◆ ◆ ◆ CHAPTER 31

Introducing IPQoS (Overview)

IP Quality of Service (IPQoS) enables you to prioritize, control, and gather accounting statistics. Using IPQoS, you can provide consistent levels of service to users of your network. You can also manage traffic to avoid network congestion.

The following is a list of topics in this chapter:

IPQoS Basics

IPQoS enables the Differentiated Services (Diffserv) architecture that is defined by the Differentiated Services Working Group of the Internet Engineering Task Force (IETF). In Oracle Solaris, IPQoS is implemented at the IP level of the TCP/IP protocol stack.

What Are Differentiated Services?

By enabling IPQoS, you can provide different levels of network service for selected customers and selected applications. The different levels of service are collectively referred to as *differentiated services*. The differentiated services that you provide to customers can be based on a structure of service levels that your company offers to its customers. You can also provide differentiated services based on the priorities that are set for applications or users on your network.

Providing quality of service involves the following activities:

- Delegating levels of service to different groups, such as customers or departments in an enterprise

- Prioritizing network services that are given to particular groups or applications
- Discovering and eliminating areas of network bottlenecks and other forms of congestion
- Monitoring network performance and providing performance statistics
- Regulating bandwidth to and from network resources

IPQoS Features

IPQoS has the following features:

- `ipqosconf` Command-line tool for configuring the QoS policy
- Classifier that selects actions, which are based on filters that configure the QoS policy of your organization
- Metering module that measures network traffic, in compliance with the Diffserv model
- Service differentiation that is based on the ability to mark a packet's IP header with forwarding information
- Flow-accounting module that gathers statistics for traffic flows
- Statistics gathering for traffic classes, through the UNIX® `kstat` command
- Support for SPARC® and x86 architecture
- Support for IPv4 and IPv6 addressing
- Interoperability with IP Security Architecture (IPsec)
- Support for 802.1D user-priority markings for virtual local area networks (VLANs)

Where to Get More Information About Quality-of-Service Theory and Practice

You can find information on differentiated services and quality of service from print and online sources.

Books About Quality of Service

For more information on quality-of-service theory and practice, refer to the following books:

- Ferguson, Paul and Geoff Huston. *Quality of Service*. John Wiley & Sons, Inc., 1998.
- Kilkki, Kalevi. *Differentiated Services for the Internet*. Macmillan Technical Publishing, 1999.

Requests for Comments (RFCs) About Quality of Service

IPQoS conforms to the specifications that are described in the following RFCs and the following Internet drafts:

- RFC 2474, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (`http://www.ietf.org/rfc/rfc2474.txt?number=2474`) – Describes an enhancement to the type of service (ToS) field or DS fields of the IPv4 and IPv6 packet headers to support differentiated services
- RFC 2475, An Architecture for Differentiated Services (`http://www.ietf.org/rfc/rfc2475.txt?number=2475`) – Provides a detailed description of the organization and modules of the Diffserv architecture
- RFC 2597, Assured Forwarding PHB Group (`http://www.ietf.org/rfc/rfc2597.txt?number=2597`) – Describes how the assured forwarding (AF) per-hop behavior works.
- RFC 2598, An Expedited Forwarding PHB (`http://www.ietf.org/rfc/rfc2598.txt?number=2598`) – Describes how the expedited forwarding (EF) per-hop behavior works
- Internet-Draft, *An Informal Management Model for Diffserv Routers* – Presents a model for implementing the Diffserv architecture on routers.

Web Sites With Quality-of-Service Information

The Differentiated Services Working Group of the IETF maintains a web site with links to Diffserv Internet drafts at `http://www.ietf.org/html.charters/diffserv-charter.html`.

Router manufacturers such as Cisco Systems and Juniper Networks provide information on their corporate web sites that describes how Differentiated Services are implemented in their products.

IPQoS Man Pages

IPQoS documentation includes the following man pages:

- `ipqosconf(1M)` - Describes the command for setting up the IPQoS configuration file
- `ipqos(7ipp)` – Describes the IPQoS implementation of the Diffserv architectural model
- `ipgpc(7ipp)` – Describes the IPQoS implementation of a Diffserv classifier
- `tokenmt(7ipp)` – Describes the IPQoS `tokenmt` meter
- `tswtclmt(7ipp)` – Describes the IPQoS `tswtclmt` meter
- `dscpmk(7ipp)` – Describes the DSCP marker module
- `dlcosmk(7ipp)` – Describes the IPQoS 802.1D user-priority marker module
- `flowacct(7ipp)`– Describes the IPQoS flow-accounting module
- `acctadm(1M)` – Describes the command that configures the Oracle Solaris extended accounting facilities. The `acctadm` command includes IPQoS extensions.

Providing Quality of Service With IPQoS

IPQoS features enable Internet service providers (ISPs) and application service providers (ASPs) to offer different levels of network service to customers. These features enable individual companies and educational institutions to prioritize services for internal organizations or for major applications.

Implementing Service-Level Agreements

If your organization is an ISP or ASP, you can base your IPQoS configuration on the *service-level agreement* (SLA) that your company offers to its customers. In an SLA, a service provider guarantees to a customer a certain level of network service that is based on a price structure. For example, a premium-priced SLA might ensure that the customer receives highest priority for all types of network traffic 24 hours per day. Conversely, a medium-priced SLA might guarantee that the customer receives high priority for email only during business hours. All other traffic would receive medium priority 24 hours a day.

Assuring Quality of Service for an Individual Organization

If your organization is an enterprise or an institution, you can also provide quality-of-service features for your network. You can guarantee that traffic from a particular group or from a certain application is assured a higher or lower degree of service.

Introducing the Quality-of-Service Policy

You implement quality of service by defining a *quality-of-service (QoS) policy*. The QoS policy defines various network attributes, such as customers' or applications' priorities, and actions for handling different categories of traffic. You implement your organization's QoS policy in an IPQoS configuration file. This file configures the IPQoS modules that reside in the Oracle Solaris kernel. A host with an applied IPQoS policy is considered an *IPQoS-enabled system*.

Your QoS policy typically defines the following:

- Discrete groups of network traffic that are called *classes of service*.
- Metrics for regulating the amount of network traffic for each class. These metrics govern the traffic-measuring process that is called *metering*.
- An action that an IPQoS system and a Diffserv router must apply to a packet flow. This type of action is called a *per-hop behavior* (PHB).
- Any statistics gathering that your organization requires for a class of service. An example is traffic that is generated by a customer or particular application.

When packets pass to your network, the IPQoS-enabled system evaluates the packet headers. The action that the IPQoS system takes is determined by your QoS policy.

Tasks for designing the QoS policy are described in "Planning the Quality-of-Service Policy" on page 625.

Improving Network Efficiency With IPQoS

IPQoS contains features that can help you make network performance more efficient as you implement quality of service. When computer networks expand, the need also increases for managing network traffic that is generated by increasing numbers of users and more powerful processors. Some symptoms of an overused network include lost data and traffic congestion. Both symptoms result in slow response times.

In the past, system administrators handled network traffic problems by adding more bandwidth. Often, the level of traffic on the links varied widely. With IPQoS, you can manage traffic on the existing network and help assess where, and whether, expansion is necessary.

For example, for an enterprise or institution, you must maintain an efficient network to avoid traffic bottlenecks. You must also ensure that a group or application does not consume more than its allotted bandwidth. For an ISP or ASP, you must manage network performance to ensure that customers receive their paid-for level of network service.

How Bandwidth Affects Network Traffic

You can use IPQoS to regulate network *bandwidth*, the maximum amount of data that a fully used network link or device can transfer. Your QoS policy should prioritize the use of bandwidth to provide quality of service to customers or users. The IPQoS metering modules enable you to measure and control bandwidth allocation among the various traffic classes on an IPQoS-enabled host.

Before you can effectively manage traffic on your network, you must answer these questions about bandwidth usage:

- What are the traffic problem areas for your local network?
- What must you do to achieve optimum use of available bandwidth?
- What are your site's critical applications, which must be given highest priority?
- Which applications are sensitive to congestion?
- What are your less critical applications, which can be given a lower priority?

Using Classes of Service to Prioritize Traffic

To implement quality of service, you analyze network traffic to determine any broad groupings into which the traffic can be divided. Then, you organize the various groupings into classes of service with individual characteristics and individual priorities. These classes form the basic categories on which you base the QoS policy for your organization. The classes of service represent the traffic groups that you want to control.

For example, a provider might offer platinum, gold, silver, and bronze levels of service, available at a sliding price structure. A platinum SLA might guarantee top priority to incoming traffic that is destined for a web site that the ISP hosts for the customer. Thus, incoming traffic to the customer's web site could be one traffic class.

For an enterprise, you could create classes of service that are based on department requirements. Or, you could create classes that are based on the preponderance of a particular application in the network traffic. Here are a few examples of traffic classes for an enterprise:

- Popular applications such as email and outgoing FTP to a particular server, either of which could constitute a class. Because employees constantly use these applications, your QoS policy might guarantee email and outgoing FTP a small amount of bandwidth and a lower priority.
- An order-entry database that needs to run 24 hours a day. Depending on the importance of the database application to the enterprise, you might give the database a large amount of bandwidth and a high priority.
- A department that performs critical work or sensitive work, such as the payroll department. The importance of the department to the organization would determine the priority and amount of bandwidth you would give to such a department.
- Incoming calls to a company's external web site. You might give this class a moderate amount of bandwidth that runs at low priority.

Differentiated Services Model

IPQoS includes the following modules, which are part of the *Differentiated Services (Diffserv)* architecture that is defined in RFC 2475:

- Classifier
- Meter
- Marker

IPQoS adds the following enhancements to the Diffserv model:

- Flow-accounting module
- 802.1D datagram marker

This section introduces the Diffserv modules as they are used by IPQoS. You need to know about these modules, their names, and their uses to set up the QoS policy. For detailed information about each module, refer to "IPQoS Architecture and the Diffserv Model" on page 681.

Classifier (`ipgpc`) Overview

In the Diffserv model, the *classifier* selects packets from a network traffic flow. A *traffic flow* consists of a group of packets with identical information in the following IP header fields:

- Source address
- Destination address
- Source port
- Destination port
- Protocol number

In IPQoS, these fields are referred to as the *5-tuple*.

The IPQoS classifier module is named `ipgpc`. The `ipgpc` classifier arranges traffic flows into classes that are based on characteristics you configure in the IPQoS configuration file.

For detailed information about `ipgpc`, refer to "Classifier Module" on page 681.

IPQoS Classes

A *class* is a group of network flows that share similar characteristics. For example, an ISP might define classes to represent the different service levels that are offered to customers. An ASP might define SLAs that give different levels of service to various applications. For an ASP's QoS policy, a class might include outgoing FTP traffic that is bound for a particular destination IP address. Outgoing traffic from a company's external web site might also be defined as a class.

Grouping traffic into classes is a major part of planning your QoS policy. When you create classes by using the `ipqosconf` utility, you are actually configuring the `ipgpc` classifier.

For information on how to define classes, see "How to Define the Classes for Your QoS Policy" on page 627.

IPQoS Filters

Filters are sets of rules that contain parameters called *selectors*. Each filter must point to a class. IPQoS matches packets against the selectors of each filter to determine if the packet belongs to the filter's class. You can filter on a packet by using a variety of selectors, for example, the IPQoS 5-tuple and other common parameters:

- Source address and destination addresses
- Source port and destination port

- Protocol numbers
- User IDs
- Project IDs
- Differentiated Services Codepoint (DSCP)
- Interface index

For example, a simple filter might include the destination port with the value of 80. The `ipgpc` classifier then selects all packets that are bound for destination port 80 (HTTP) and handles the packets as directed in the QoS policy.

For information on creating filters, see "How to Define Filters in the QoS Policy" on page 630.

Meter (`tokenmt` and `tswtclmt`) Overview

In the Diffserv model, the *meter* tracks the transmission rate of traffic flows on a per-class basis. The meter evaluates how much the actual rate of the flow conforms to the configured rates to determine the appropriate outcome. Based on the traffic flow's outcome, the meter selects a subsequent action. Subsequent actions might include sending the packet to another action or returning the packet to the network without further processing.

The IPQoS meters determine whether a network flow conforms to the transmission rate that is defined for its class in the QoS policy. IPQoS includes two metering modules:

- `tokenmt` – Uses a two-token bucket metering scheme
- `tswtclmt` – Uses a time-sliding window metering scheme

Both metering modules recognize three outcomes: red, yellow, and green. You define the actions to be taken for each outcome in the parameters `red_action_name`, `yellow_action_name`, and `green_action_name`.

In addition, you can configure `tokenmt` to be color aware. A color-aware metering instance uses the packet's size, DSCP, traffic rate, and configured parameters to determine the outcome. The meter uses the DSCP to map the packet's outcome to a green, yellow, or red.

For information on defining parameters for the IPQoS meters, refer to "How to Plan Flow Control" on page 631.

Marker (`dscpmk` and `dlcosmk`) Overview

In the Diffserv model, the *marker* marks a packet with a value that reflects a forwarding behavior. *Marking* is the process of placing a value in the packet's header to indicate how to forward the packet to the network. IPQoS contains two marker modules:

- `dscpmk` – Marks the DS field in an IP packet header with a numeric value that is called the *Differentiated Services codepoint*, or *DSCP*. A Diffserv-aware router can then use the DS codepoint to apply the appropriate forwarding behavior to the packet.

- `dlcosmk` – Marks the virtual local area network (VLAN) tag of an Ethernet frame header with a numeric value that is called the *user priority*. The user priority indicates the *class of service (CoS)*, which defines the appropriate forwarding behavior to be applied to the datagram.

  `dlcosmk` is an IPQoS addition that is not part of the Diffserv model, as designed by the IETF.

For information on implementing a marker strategy for the QoS policy, see "How to Plan Forwarding Behavior" on page 634.

Flow Accounting (`flowacct`) Overview

IPQoS adds the `flowacct` accounting module to the Diffserv model. You can use `flowacct` to gather statistics on traffic flows, and bill customers in agreement with their SLAs. Flow accounting is also useful for capacity planning and system monitoring.

The `flowacct` module works with the `acctadm` command to create an accounting log file. A basic log includes the IPQoS 5-tuple and two additional attributes, as shown in the following list:

- Source address
- Source port
- Destination address
- Destination port
- Protocol number
- Number of packets
- Number of bytes

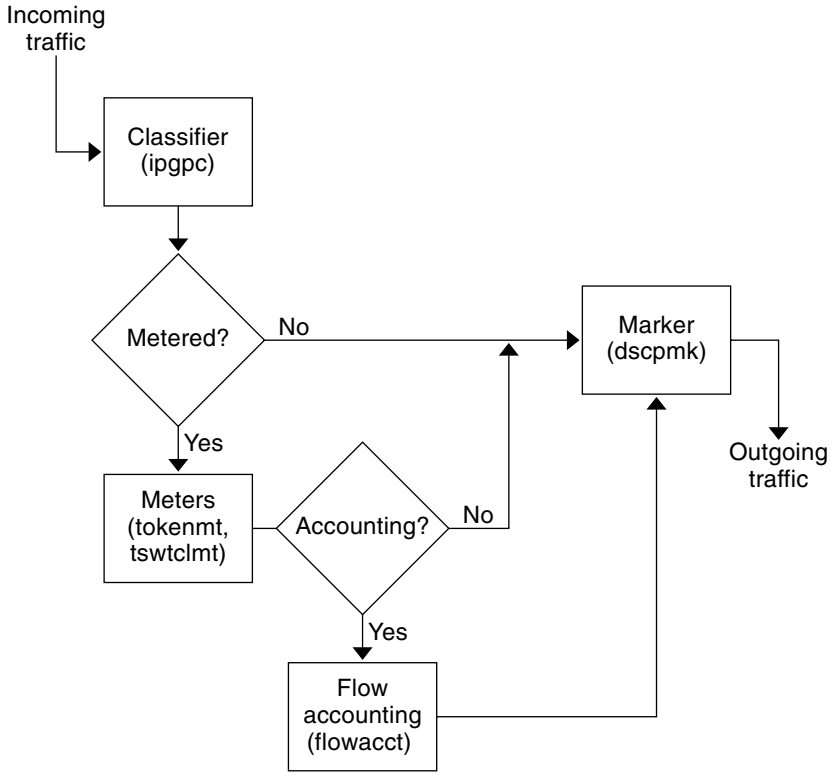
You can also gather statistics on other attributes, as described in "Recording Information About Traffic Flows" on page 675, and in the `flowacct`(7ipp) and `acctadm`(1M) man pages.

For information on planning a flow-accounting strategy, see "How to Plan for Flow Accounting" on page 636.

How Traffic Flows Through the IPQoS Modules

The next figure shows a path that incoming traffic might take through some of the IPQoS modules.

FIGURE 31–1 Traffic Flow Through the IPQoS Implementation of the Diffserv Model



This figure illustrates a common traffic flow sequence on an IPQoS-enabled machine:

1. The classifier selects from the packet stream all packets that match the filtering criteria in the system's QoS policy.
2. The selected packets are then evaluated for the next action to be taken.
3. The classifier sends to the marker any traffic that does not require flow control.
4. Traffic to be flow-controlled is sent to the meter.
5. The meter enforces the configured rate. Then, the meter assigns a traffic conformance value to the flow-controlled packets.
6. The flow-controlled packets are then evaluated to determine if any packets require accounting.
7. The meter sends to the marker any traffic that does not require flow accounting.
8. The flow-accounting module gathers statistics on received packets. The module then sends the packets to the marker.

9. The marker assigns a DS codepoint to the packet header. This DSCP indicates the per-hop behavior that a Diffserv-aware system must apply to the packet.

Traffic Forwarding on an IPQoS-Enabled Network

This section introduces the elements that are involved in forwarding packets on an IPQoS-enabled network. An IPQoS-enabled system handles any packets on the network stream with the system's IP address as the destination. The IPQoS system then applies its QoS policy to the packet to establish differentiated services.

DS Codepoint

The DS codepoint (DSCP) defines in the packet header the action that any Diffserv-aware system should take on a marked packet. The diffserv architecture defines a set of DS codepoints for the IPQoS-enabled system and diffserv router to use. The Diffserv architecture also defines a set of actions that are called *forwarding behaviors*, which correspond to the DSCPs. The IPQoS-enabled system marks the precedence bits of the DS field in the packet header with the DSCP. When a router receives a packet with a DSCP value, the router applies the forwarding behavior that is associated with that DSCP. The packet is then released onto the network.

Note – The `dlcosmk` marker does not use the DSCP. Rather, `dlcosmk` marks Ethernet frame headers with a CoS value. If you plan to configure IPQoS on a network that uses VLAN devices, refer to "Marker Module" on page 686.

Per-Hop Behaviors

In Diffserv terminology, the forwarding behavior that is assigned to a DSCP is called the *per-hop behavior (PHB)*. The PHB defines the forwarding precedence that a marked packet receives in relation to other traffic on the Diffserv-aware system. This precedence ultimately determines whether the IPQoS-enabled system or Diffserv router forwards or drops the marked packet. For a forwarded packet, each Diffserv router that the packet encounters en route to its destination applies the same PHB. The exception is if another Diffserv system changes the DSCP. For more information on PHBs, refer to "Using the `dscpmk` Marker for Forwarding Packets" on page 687.

The goal of a PHB is to provide a specified amount of network resources to a class of traffic on the contiguous network. You can achieve this goal in the QoS policy. Define DSCPs that indicate the precedence levels for traffic classes when the traffic flows leave the IPQoS-enabled system. Precedences can range from high-precedence/low-drop probability to low-precedence/high-drop probability.

For example, your QoS policy can assign to one class of traffic a DSCP that guarantees a low-drop PHB. This traffic class then receives a low-drop precedence PHB from any Diffserv-aware router, which guarantees bandwidth to packets of this class. You can add to the QoS policy other DSCPs that assign varying levels of precedence to other traffic classes. The lower-precedence packets are given bandwidth by Diffserv systems in agreement with the priorities that are indicated in the packets' DSCPs.

IPQoS supports two types of forwarding behaviors, which are defined in the Diffserv architecture, expedited forwarding and assured forwarding.

Expedited Forwarding

The *expedited forwarding (EF)* per-hop behavior assures that any traffic class with EFs related DSCP is given highest priority. Traffic with an EF DSCP is not queued. EF provides low loss, latency, and jitter. The recommended DSCP for EF is 101110. A packet that is marked with 101110 receives guaranteed low-drop precedence as the packet traverses Diffserv-aware networks en route to its destination. Use the EF DSCP when assigning priority to customers or applications with a premium SLA.

Assured Forwarding

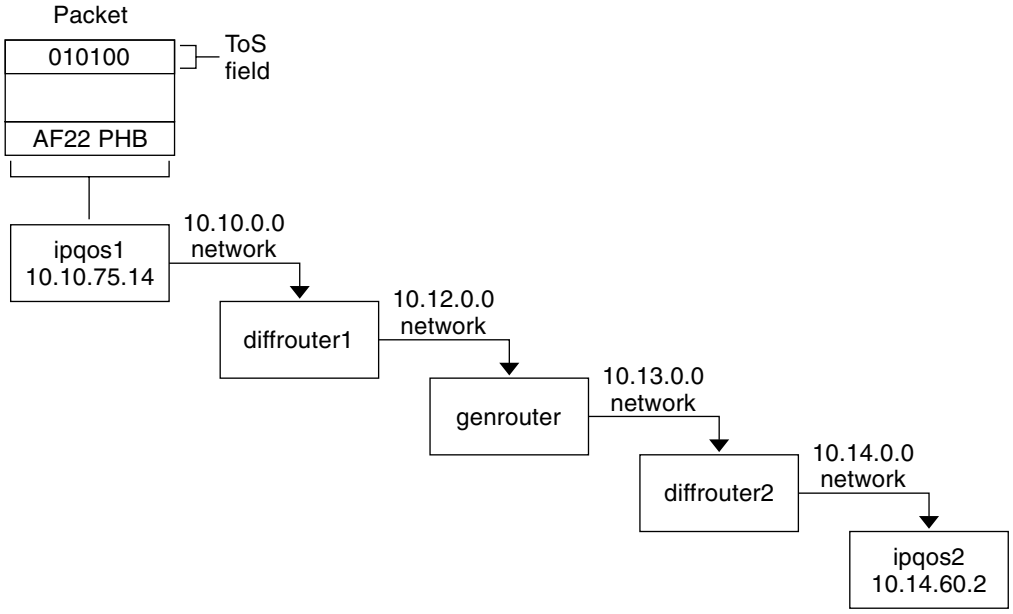
The *assured forwarding (AF)* per-hop behavior provides four different forwarding classes that you can assign to a packet. Every forwarding class provides three drop precedences, as shown in Table 36–2.

The various AF codepoints provide the ability to assign different levels of service to customers and applications. In the QoS policy, you can prioritize traffic and services on your network when you plan the QoS policy. You can then assign different AF levels to the prioritized traffic.

Packet Forwarding in a Diffserv Environment

The following figure shows part of an intranet at a company with a partially Diffserv-enabled environment. In this scenario, all hosts on networks `10.10.0.0` and `10.14.0.0` are IPQoS enabled, and the local routers on both networks are Diffserv aware. However, the interim networks are not configured for Diffserv.

FIGURE 31–2 Packet Forwarding Across Diffserv-Aware Network Hops



The next steps trace the flow of the packet that is shown in this figure. The steps begin with the progress of a packet that originates at host `ipqos1`. The steps then continue through several hops to host `ipqos2`.

1. The user on `ipqos1` runs the `ftp` command to access host `ipqos2`, which is three hops away.
2. `ipqos1` applies its QoS policy to the resulting packet flow. `ipqos1` then successfully classifies the `ftp` traffic.

   The system administrator has created a class for all outgoing `ftp` traffic that originates on the local network 10.10.0.0. Traffic for the `ftp` class is assigned the AF22 per-hop behavior: class two, medium-drop precedence. A traffic flow rate of 2Mb/sec is configured for the `ftp` class.
3. `ipqos-1` meters the `ftp` flow to determine if the flow exceeds the committed rate of 2 Mbit/sec.
4. The marker on `ipqos1` marks the DS fields in the outgoing `ftp` packets with the 010100 DSCP, corresponding to the AF22 PHB.
5. The router `diffrouter1` receives the `ftp` packets. `diffrouter1` then checks the DSCP. If `diffrouter1` is congested, packets that are marked with AF22 are dropped.
6. `ftp` traffic is forwarded to the next hop in agreement with the per-hop behavior that is configured for AF22 in `diffrouter1`'s files.
7. The `ftp` traffic traverses network `10.12.0.0` to `genrouter`, which is not Diffserv aware. As a result, the traffic receives "best-effort" forwarding behavior.

8. `genrouter` passes the `ftp` traffic to network `10.13.0.0`, where the traffic is received by `diffrouter2`.
9. `diffrouter2` is Diffserv aware. Therefore, the router forwards the `ftp` packets to the network in agreement with the PHB that is defined in the router policy for AF22 packets.
10. `ipqos2` receives the `ftp` traffic. `ipqos2` then prompts the user on `ipqos1` for a user name and password.

◆ ◆ ◆ CHAPTER 32

Planning for an IPQoS-Enabled Network (Tasks)

You can configure IPQoS on any system that runs Oracle Solaris. The IPQoS system then works with Diffserv-aware routers to provide differentiated services and traffic management on an intranet.

This chapter contains planning tasks for adding IPQoS-enabled systems onto a Diffserv-aware network. The following topics are covered.

General IPQoS Configuration Planning (Task Map)

Implementing differentiated services, including IPQoS, on a network requires extensive planning. You must consider not only the position and function of each IPQoS-enabled system, but also each system's relationship to the router on the local network. The following task map lists the major planning tasks for implementing IPQoS on your network and links to procedures to complete the tasks.

Task	Description	For Instructions
1. Plan a Diffserv network topology that incorporates IPQoS-enabled systems.	Learn about the various Diffserv network topologies to determine the best solution for your site.	"Planning the Diffserv Network Topology" on page 622.
2. Plan the different types of services to be offered by the IPQoS systems.	Organize the types of services that the network provides into service-level agreements (SLAs).	"Planning the Quality-of-Service Policy" on page 625.

Task	Description	For Instructions
3. Plan the QoS policy for each IPQoS system.	Decide on the classes, metering, and accounting features that are needed to implement each SLA.	"Planning the Quality-of-Service Policy" on page 625.
4. If applicable, plan the policy for the Diffserv router.	Decide any scheduling and queuing policies for the Diffserv router that is used with the IPQoS systems.	Refer to router documentation for queuing and scheduling policies.

Planning the Diffserv Network Topology

To provide differentiated services for your network, you need at least one IPQoS-enabled system and a Diffserv-aware router. You can expand this basic scenario in a variety of ways, as explained in this section.

Hardware Strategies for the Diffserv Network

Typically, customers run IPQoS on servers and server consolidations, such as the Sun Enterprise™ 0000 server. Conversely, you can also run IPQoS on desktop systems such as UltraSPARC® systems, depending on the needs of your network. The following list describes possible systems for an IPQoS configuration:

- Oracle Solaris systems that offer various services, such as web servers and database servers
- Application servers that offer email, FTP, or other popular network applications
- Web cache servers or proxy servers
- Network of IPQoS-enabled server farms that are managed by Diffserv-aware load balancers
- Firewalls that manage traffic for a single heterogeneous network
- IPQoS systems that are part of a virtual local area network (LAN)

You might introduce IPQoS systems into a network topology with already functioning Diffserv-aware routers. If your router does not currently offer Diffserv, consider the Diffserv solutions that are offered by Cisco Systems, Juniper Networks, and other router manufacturers. If the local router does not implement Diffserv, then the router passes marked packets on to the next hop without evaluating the marks.

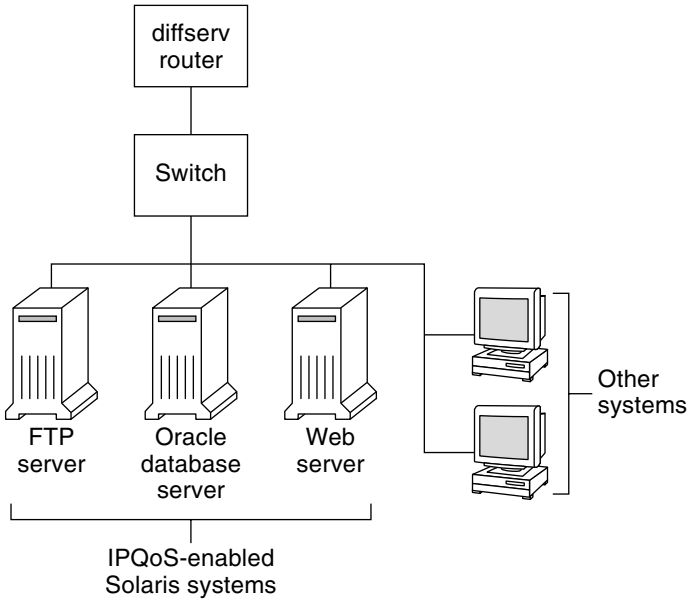
IPQoS Network Topologies

This section illustrates IPQoS strategies for various network needs.

IPQoS on Individual Hosts

The following figure shows a single network of IPQoS-enabled systems.

FIGURE 32–1 IPQoS Systems on a Network Segment



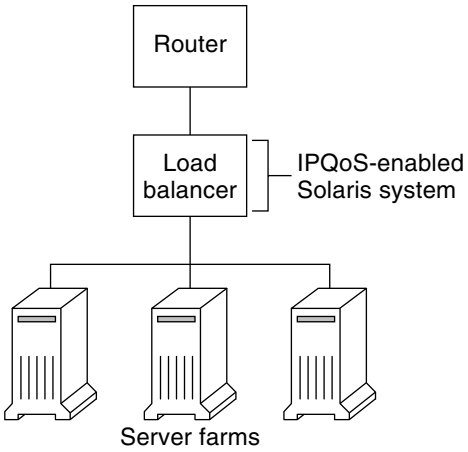
This network is but one segment of a corporate intranet. By enabling IPQoS on the application servers and web servers, you can control the rate at which each IPQoS system releases outgoing traffic. If you make the router Diffserv aware, you can further control incoming and outgoing traffic.

The examples in this guide use the "IPQoS on an individual host" scenario. For the example topology that is used throughout the guide, see Figure 32–4.

IPQoS on a Network of Server Farms

The following figure shows a network with several heterogeneous server farms.

FIGURE 32–2 Network of IPQoS-Enabled Server Farms



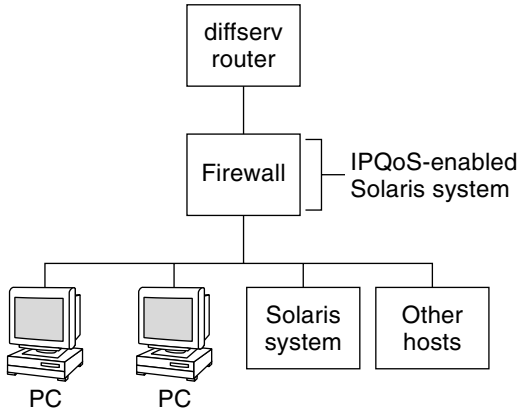
In such a topology, the router is Diffserv aware, and therefore able to queue and rate both incoming and outgoing traffic. The load balancer is also Diffserv-aware, and the server farms are IPQoS enabled. The load balancer can provide additional filtering beyond the router by using selectors such as user ID and project ID. These selectors are included in the application data.

This scenario provides flow control and traffic forwarding to manage congestion on the local network. This scenario also prevents outgoing traffic from the server farms from overloading other portions of the intranet.

IPQoS on a Firewall

The following figure shows a segment of a corporate network that is secured from other segments by a firewall.

FIGURE 32–3 Network Protected by an IPQoS-Enabled Firewall



In this scenario, traffic flows into a Diffserv-aware router where the packets are filtered and queued. All incoming traffic that is forwarded by the router then travels into the IPQoS-enabled firewall. To use IPQoS, the firewall must not bypass the IP forwarding stack.

The firewall's security policy determines whether incoming traffic is permitted to enter or depart the internal network. The QoS policy controls the service levels for incoming traffic that has passed the firewall. Depending on the QoS policy, outgoing traffic can also be marked with a forwarding behavior.

Planning the Quality-of-Service Policy

When you plan the quality-of-service (QoS) policy, you must review, classify, and then prioritize the services that your network provides. You must also assess the amount of available bandwidth to determine the rate at which each traffic class is released onto the network.

QoS Policy Planning Aids

Gather information for planning the QoS policy in a format that includes the information needed for the IPQoS configuration file. For example, you can use the following template to list the major categories of information to be used in the IPQoS configuration file.

TABLE 32–1 QoS Planning Template

Class	Priority	Filter	Selector	Rate	Forwarding?	Accounting?
Class 1	1	Filter 1 Filter 3	Selector 1 Selector 2	Meter rates, depending on meter type	Marker drop precedence	Requires flow-accounting statistics

TABLE 32–1 QoS Planning Template *(Continued)*

Class	Priority	Filter	Selector	Rate	Forwarding?	Accounting?
Class 1	1	Filter 2	Selector 1 Selector 2	N/A	N/A	N/A
Class 2	2	Filter 1	Selector 1 Selector 2	Meter rates, depending on meter type	Marker drop precedence	Requires flow-accounting statistics
Class 2	2	Filter 2	Selector 1 Selector 2	N/A	N/A	N/A

You can divide each major category to further define the QoS policy. Subsequent sections explain how to obtain information for the categories that are shown in the template.

QoS Policy Planning (Task Map)

This task map lists the major tasks for planning a QoS policy and links to the instructions to perform each task.

Task	Description	For Instructions
1. Design your network topology to support IPQoS.	Identify the hosts and routers on your network to provide differentiated services.	"How to Prepare a Network for IPQoS" on page 627
2. Define the classes into which services on your network must be divided.	Examine the types of services and SLAs that are offered by your site, and determine the discrete traffic classes into which these services fall.	"How to Define the Classes for Your QoS Policy" on page 627
3. Define filters for the classes.	Determine the best ways of separating traffic of a particular class from the network traffic flow.	"How to Define Filters in the QoS Policy" on page 630
4. Define flow-control rates for measuring traffic as packets leave the IPQoS system.	Determine acceptable flow rates for each class of traffic.	"How to Plan Flow Control" on page 631
5. Define DSCPs or user-priority values to be used in the QoS policy.	Plan a scheme to determine the forwarding behavior that is assigned to a traffic flow when the flow is handled by the router or switch.	"How to Plan Forwarding Behavior" on page 634
6. If applicable, set up a statistics-monitoring plan for traffic flows on the network.	Evaluate the traffic classes to determine which traffic flows must be monitored for accounting or statistical purposes.	"How to Plan for Flow Accounting" on page 636

Note – The rest of this section explains how to plan the QoS policy of an IPQoS-enabled system. To plan the QoS policy for the Diffserv router, refer to the router documentation and the router manufacturer's web site.

How to Prepare a Network for IPQoS

The following procedure lists general planning tasks to do before you create the QoS policy.

1. **Review your network topology. Then, plan a strategy that uses IPQoS systems and Diffserv routers.**

   For topology examples, see "Planning the Diffserv Network Topology" on page 622.

2. **Identify the hosts in the topology that require IPQoS or that might become good candidates for IPQoS service.**

3. **Determine which IPQoS-enabled systems could use the same QoS policy.**

   For example, if you plan to enable IPQoS on all hosts on the network, identify any hosts that could use the same QoS policy. Each IPQoS-enabled system must have a local QoS policy, which is implemented in its IPQoS configuration file. However, you can create one IPQoS configuration file to be used by a range of systems. You can then copy the configuration file to every system with the same QoS policy requirements.

4. **Review and perform any planning tasks that are required by the Diffserv router on your network.**

   Refer to the router documentation and the router manufacturer's web site for details.

How to Define the Classes for Your QoS Policy

The first step in defining the QoS policy is organizing traffic flows into classes. You do not need to create classes for every type of traffic on a Diffserv network. Moreover, depending on your network topology, you might have to create a different QoS policy for each IPQoS-enabled system.

Note – For an overview of classes, see "IPQoS Classes" on page 613.

The next procedure assumes that you have determined which systems on your network are to be IPQoS-enabled, as identified in "How to Prepare a Network for IPQoS" on page 627.

1. **Create a QoS planning table for organizing the QoS policy information.**

   For suggestions, refer to Table 32–1.

2 Perform the remaining steps for every QoS policy that is on your network.

3 Define the classes to be used in the QoS policy.

The following questions are a guideline for analyzing network traffic for possible class definitions.

- **Does your company offer service-level agreements to customers?**

  If yes, then evaluate the relative priority levels of the SLAs that your company offers to customers. The same applications might be offered to customers who are guaranteed different priority levels.

  For example, your company might offer web site hosting to each customer, which indicates that you need to define a class for each customer web site. One SLA might provide a premium web site as one service level. Another SLA might offer a "best-effort" personal web site to discount customers. This factor indicates not only different web site classes but also potentially different per-hop behaviors that are assigned to the web site classes.

- **Does the IPQoS system offer popular applications that might need flow control?**

  You can improve network performance by enabling IPQoS on servers offering popular applications that generate excessive traffic. Common examples are electronic mail, network news, and FTP. Consider creating separate classes for incoming and outgoing traffic for each service type, where applicable. For example, you might create a `mail-in` class and a `mail-out` class for the QoS policy for a mail server.

- **Does your network run certain applications that require highest-priority forwarding behaviors?**

  Any critical applications that require highest-priority forwarding behaviors must receive highest priority in the router's queue. Typical examples are streaming video and streaming audio.

  Define incoming classes and outgoing classes for these high-priority applications. Then, add the classes to the QoS policies of both the IPQoS-enabled system that serves the applications and the Diffserv router.

- **Does your network experience traffic flows that must be controlled because the flows consume large amounts of bandwidth?**

  Use `netstat`, `snoop`, and other network monitoring utilities to discover the types of traffic that are causing problems on the network. Review the classes that you have created thus far, and then create new classes for any undefined problem traffic category. If you have already defined classes for a category of problem traffic, then define rates for the meter to control the problem traffic.

  Create classes for the problem traffic on every IPQoS-enabled system on the network. Each IPQoS system can then handle any problem traffic by limiting the rate at which the traffic flow is released onto the network. Be sure also to define these problem classes in the QoS policy on the Diffserv router. The router can then queue and schedule the problem flows as configured in its QoS policy.

- **Do you need to obtain statistics on certain types of traffic?**

  A quick review of an SLA can indicate which types of customer traffic require accounting. If your site does offer SLAs, you probably have already created classes for traffic that requires accounting. You might also define classes to enable statistics gathering on traffic flows that you are monitoring. You could also create classes for traffic to which you restrict access for security reasons.

4 **List the classes that you have defined in the QoS planning table you created in Step 1.**

5 **Assign a priority level to each class.**

For example, have priority level 1 represent the highest-priority class, and assign descending-level priorities to the remaining classes. The priority level that you assign is for organizational purposes only. Priority levels that you set in the QoS policy template are not actually used by IPQoS. Moreover, you can assign the same priority to more than one class, if appropriate for your QoS policy.

6 **When you finish defining classes, you next define filters for each class, as explained in "How to Define Filters in the QoS Policy" on page 630.**

More Information

Prioritizing the Classes

As you create classes, you quickly realize which classes have highest priority, medium priority, and best-effort priority. A good scheme for prioritizing classes becomes particularly important when you assign per-hop behaviors to outgoing traffic, as explained in "How to Plan Forwarding Behavior" on page 634.

In addition to assigning a PHB to a class, you can also define a priority selector in a filter for the class. The priority selector is active on the IPQoS-enabled host only. Suppose several classes with equal rates and identical DSCPs sometimes compete for bandwidth as they leave the IPQoS system. The priority selector in each class can further order the level of service that is given to the otherwise identically valued classes.

Defining Filters

You create filters to identify packet flows as members of a particular class. Each filter contains selectors, which define the criteria for evaluating a packet flow. The IPQoS-enabled system then uses the criteria in the selectors to extract packets from a traffic flow. The IPQoS system then associates the packets with a class. For an introduction to filters, see "IPQoS Filters" on page 613.

The following table lists the most commonly used selectors. The first five selectors represent the IPQoS 5-tuple, which the IPQoS system uses to identify packets as members of a flow. For a complete list of selectors, see Table 36–1.

TABLE 32–2 Common IPQoS Selectors

Name	Definition
`saddr`	Source address.
`daddr`	Destination address.
`sport`	Source port number. You can use a well-known port number, as defined in `/etc/services`, or a user-defined port number.
`dport`	Destination port number.
`protocol`	IP protocol number or protocol name that is assigned to the traffic flow type in `/etc/protocols`.
`ip_version`	Addressing style to use. Use either IPv4 or IPv6. IPv4 is the default.
`dsfield`	Contents of the DS field, that is, the DSCP. Use this selector for extracting incoming packets that are already marked with a particular DSCP.
`priority`	Priority level that is assigned to the class. For more information, see "How to Define the Classes for Your QoS Policy" on page 627.
`user`	Either the UNIX user ID or user name that is used when the upper-level application is executed.
`projid`	Project ID that is used when the upper-level application is executed.
`direction`	Direction of traffic flow. Value is either `LOCAL_IN`, `LOCAL_OUT`, `FWD_IN`, or `FWD_OUT`.

Note – Be judicious in your choice of selectors. Use only as many selectors as you need to extract packets for a class. The more selectors that you define, the greater the impact on IPQoS performance.

▼ How to Define Filters in the QoS Policy

Before You Begin Before you can perform the next steps, you should have completed the procedure "How to Define the Classes for Your QoS Policy" on page 627.

1 Create at least one filter for each class in the QoS planning table that you created in "How to Define the Classes for Your QoS Policy" on page 627.

Consider creating separate filters for incoming and outgoing traffic for each class, where applicable. For example, add an `ftp-in` filter and an `ftp-out` filter to the QoS policy of an IPQoS-enabled FTP server. You then can define an appropriate `direction` selector in addition to the basic selectors.

2 **Define at least one selector for each filter in a class.**

Use the QoS planning table that was introduced in Table 32–1 to fill in filters for the classes you defined.

Example 32–1 Defining Filters for FTP Traffic

The next table is an example that shows how you would define a filter for outgoing FTP traffic.

Class	Priority	Filters	Selectors
`ftp-traffic`	4	`ftp-out`	`saddr 10.190.17.44`
			`daddr 10.100.10.53`
			`sport 21`
			`direction LOCAL_OUT`

See Also

- To define a flow-control scheme, refer to "How to Plan Flow Control" on page 631.
- To define forwarding behaviors for flows as the flows return to the network stream, refer to "How to Plan Forwarding Behavior" on page 634.
- To plan for flow accounting of certain types of traffic, refer to "How to Plan for Flow Accounting" on page 636.
- To add more classes to the QoS policy, refer to "How to Define the Classes for Your QoS Policy" on page 627.
- To add more filters to the QoS policy, refer to "How to Define Filters in the QoS Policy" on page 630.

▼ How to Plan Flow Control

Flow control involves measuring traffic flow for a class and then releasing packets onto the network at a defined rate. When you plan flow control, you define parameters to be used by the IPQoS metering modules. The meters determine the rate at which traffic is released onto the network. For an introduction to the metering modules, see "Meter (`tokenmt` and `tswtclmt`) Overview" on page 614.

The next procedure assumes that you have defined filters and selectors, as described in "How to Define Filters in the QoS Policy" on page 630.

1 **Determine the maximum bandwidth for your network.**

2 **Review any SLAs that are supported on your network. Identify customers and the type of service that is guaranteed to each customer.**

To guarantee a certain level of service, you might need to meter certain traffic classes that are generated by the customer.

3 **Review the list of classes that you created in "How to Define the Classes for Your QoS Policy" on page 627.**

Determine if any classes other than those classes that are associated with SLAs need to be metered.

Suppose the IPQoS system runs an application that generates a high level of traffic. After you classify the application's traffic, meter the flows to control the rate at which the packets of the flow return to the network.

Note – Not all classes need to be metered. Remember this guideline as you review your list of classes.

---

4 **Determine which filters in each class select traffic that needs flow control. Then, refine your list of classes that require metering.**

Classes that have more than one filter might require metering for only one filter. Suppose that you define filters for incoming and outgoing traffic of a certain class. You might conclude that only traffic in one direction requires flow control.

5 **Choose a meter module for each class to be flow controlled.**

Add the module name to the meter column in your QoS planning table.

6 **Add the rates for each class to be metered to the organizational table.**

If you use the `tokenmt` module, you need to define the following rates in bits per second:

- Committed rate
- Peak rate

If these rates are sufficient to meter a particular class, you can define only the committed rate and the committed burst for `tokenmt`.

If needed, you can also define the following rates:

- Committed burst
- Peak burst

For a complete definition of `tokenmt` rates, refer to "Configuring `tokenmt` as a Two-Rate Meter" on page 685. You can also find more detailed information in the `tokenmt`(7ipp) man page.

If you use the `tswtclmt` module, you need to define the following rates in bits per second.

- Committed rate
- Peak rate

You can also define the window size in milliseconds. These rates are defined in "tswtclmt Metering Module" on page 686 and in the `twstclmt`(7ipp) man page.

7 Add traffic conformance outcomes for the metered traffic.

The outcomes for both metering modules are green, red, and yellow. Add to your QoS organizational table the traffic conformance outcomes that apply to the rates you define. Outcomes for the meters are fully explained in "Meter Module" on page 684.

You need to determine what action should be taken on traffic that conforms, or does not conform, to the committed rate. Often, but not always, this action is to mark the packet header with a per-hop behavior. One acceptable action for green-level traffic could be to continue processing while traffic flows do not exceed the committed rate. Another action could be to drop packets of the class if flows exceed peak rate.

Example 32–2 Defining Meters

The next table is an example that shows meter entries for a class of email traffic. The network on which the IPQoS system is located has a total bandwidth of 100 Mbits/sec, or 10000000 bits per second. The QoS policy assigns a low priority to the email class. This class also receives best-effort forwarding behavior.

Class	Priority	Filter	Selector	Rate
`email`	8	`mail_in`	`daddr10.50.50.5`	
			`dport imap`	
			`direction LOCAL_IN`	
`email`	8	`mail_out`	`saddr10.50.50.5`	meter=`tokenmt`
			`sport imap`	committed rate=5000000
			`direction LOCAL_OUT`	committed burst =5000000
				peak rate =10000000
				peak burst=1000000
				green precedence=continue processing
				yellow precedence=mark yellow PHB
				red precedence=drop

See Also

- To define forwarding behaviors for flows as the packets return to the network stream, refer to "How to Plan Forwarding Behavior" on page 634.

- To plan for flow accounting of certain types of traffic, refer to "How to Plan for Flow Accounting" on page 636.
- To add more classes to the QoS policy, refer to "How to Define the Classes for Your QoS Policy" on page 627.
- To add more filters to the QoS policy, refer to "How to Define Filters in the QoS Policy" on page 630.
- To define another flow-control scheme, refer to "How to Plan Flow Control" on page 631.
- To create an IPQoS configuration file, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645.

How to Plan Forwarding Behavior

Forwarding behavior determines the priority and drop precedence of traffic flows that are about to be forwarded to the network. You can choose two major forwarding behaviors: prioritize the flows of a class in relationship to other traffic classes or drop the flows entirely.

The Diffserv model uses the marker to assign the chosen forwarding behavior to traffic flows. IPQoS offers the following marker modules.

- `dscpmk` – Used to mark the DS field of an IP packet with a DSCP
- `dlcosmk` – Used to mark the VLAN tag of a datagram with a class-of-service (CoS) value

---

Note – The suggestions in this section refer specifically to IP packets. If your IPQoS system includes a VLAN device, you can use the `dlcosmk` marker to mark forwarding behaviors for datagrams. For more information, refer to "Using the `dlcosmk` Marker With VLAN Devices" on page 689.

To prioritize IP traffic, you need to assign a DSCP to each packet. The `dscpmk` marker marks the DS field of the packet with the DSCP. You choose the DSCP for a class from a group of well-known codepoints that are associated with the forwarding behavior type. These well-known codepoints are 46 (101110) for the EF PHB and a range of codepoints for the AF PHB. For overview information on DSCP and forwarding, refer to "Traffic Forwarding on an IPQoS-Enabled Network" on page 617.

Before You Begin The next steps assume that you have defined classes and filters for the QoS policy. Though you often use the meter with the marker to control traffic, you can use the marker alone to define a forwarding behavior.

1 **Review the classes that you have created thus far and the priorities that you have assigned to each class.**

Not all traffic classes need to be marked.

2. **Assign the EF per-hop behavior to the class with the highest priority.**

   The EF PHB guarantees that packets with the EF DSCP 46 (101110) are released onto the network before packets with any AF PHBs. Use the EF PHB for your highest-priority traffic. For more information about EF, refer to "Expedited Forwarding (EF) PHB" on page 687.

3. **Assign forwarding behaviors to classes that have traffic to be metered.**

4. **Assign DS codepoints to the remaining classes in agreement with the priorities that you have assigned to the classes.**

Example 32–3 QoS Policy for a Games Application

Traffic is generally metered for the following reasons:

- An SLA guarantees packets of this class greater service or lesser service when the network is heavily used.
- A class with a lower priority might have a tendency to flood the network.

You use the marker with the meter to provide differentiated services and bandwidth management to these classes. For example, the following table shows a portion of a QoS policy. This policy defines a class for a popular games application that generates a high level of traffic.

Class	Priority	Filter	Selector	Rate	Forwarding?
`games_app`	9	`games_in`	`sport 6080`	N/A	N/A
`games_app`	9	`games_out`	`dport 6081`	meter=`tokenmt` committed rate=5000000 committed burst =5000000 peak rate =10000000 peak burst=15000000 green precedence=continue processing yellow precedence=mark yellow PHB red precedence=drop	green =AF31 yellow=AF42 red=drop

The forwarding behaviors assign low-priority DSCPs to `games_app` traffic that conforms to its committed rate or is under the peak rate. When `games_app` traffic exceeds peak rate, the QoS policy indicates that packets from `games_app` are to be dropped. All AF codepoints are listed in Table 36–2.

See Also
- To plan for flow accounting of certain types of traffic, refer to "How to Plan for Flow Accounting" on page 636.
- To add more classes to the QoS policy, refer to "How to Define the Classes for Your QoS Policy" on page 627.
- To add more filters to the QoS policy, refer to "How to Define Filters in the QoS Policy" on page 630.
- To define a flow-control scheme, refer to "How to Plan Flow Control" on page 631.
- To define additional forwarding behaviors for flows as the packets return to the network stream, refer to "How to Plan Forwarding Behavior" on page 634.
- To create an IPQoS configuration file, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645.

▼ How to Plan for Flow Accounting

You use the IPQoS `flowacct` module to track traffic flows for billing or network management purposes. Use the following procedure to determine if your QoS policy should include flow accounting.

1 **Does your company offer SLAs to customers?**

If the answer is yes, then you should use flow accounting. Review the SLAs to determine what types of network traffic your company wants to bill customers for. Then, review your QoS policy to determine which classes select traffic to be billed.

2 **Are there applications that might need monitoring or testing to avoid network problems?**

If the answer is yes, consider using flow accounting to observe the behavior of these applications. Review your QoS policy to determine the classes that you have assigned to traffic that requires monitoring.

3 **Mark Y in the flow-accounting column for each class that requires flow accounting in your QoS planning table.**

See Also
- To add more classes to the QoS policy, refer to "How to Define the Classes for Your QoS Policy" on page 627.
- To add more filters to the QoS policy, refer to "How to Define Filters in the QoS Policy" on page 630.

- To define a flow-control scheme, refer to "How to Plan Flow Control" on page 631.
- To define forwarding behaviors for flows as the packets return to the network stream, refer to "How to Plan Forwarding Behavior" on page 634.
- To plan for additional flow accounting of certain types of traffic, refer to "How to Plan for Flow Accounting" on page 636.
- To create the IPQoS configuration file, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645.

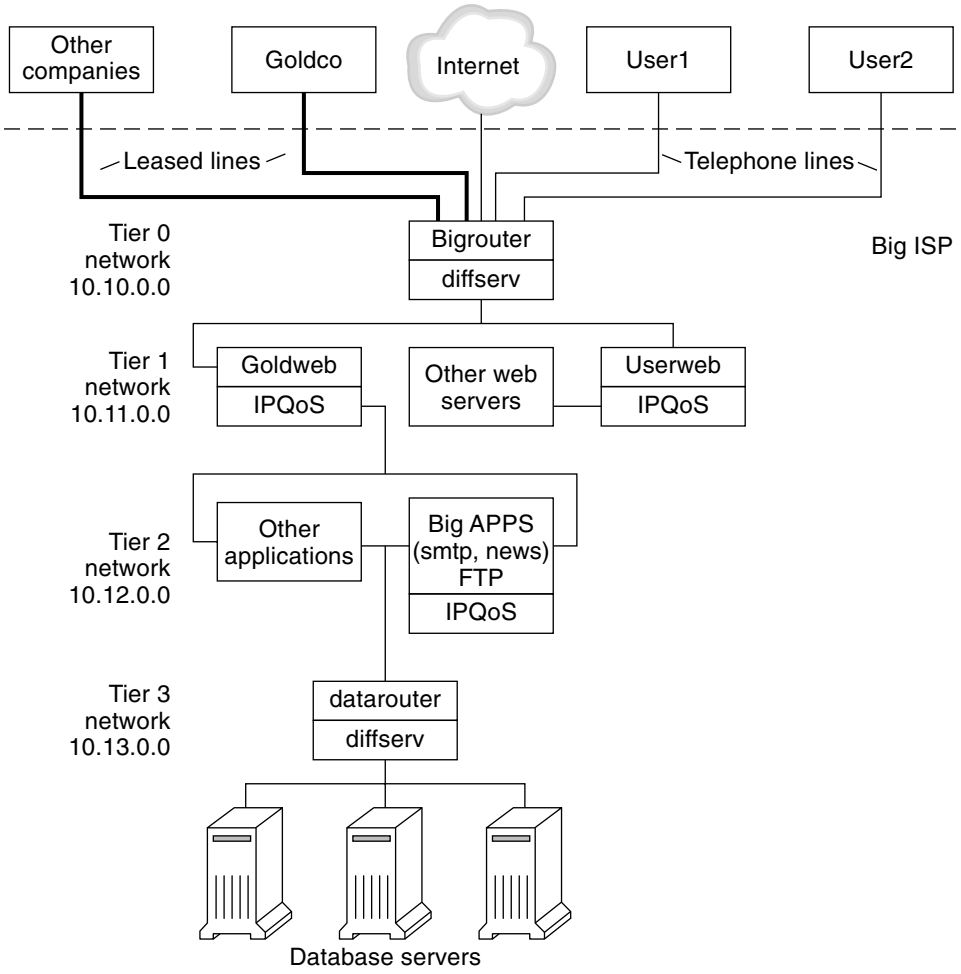
Introducing the IPQoS Configuration Example

Tasks in the remaining chapters of the guide use the example IPQoS configuration that is introduced in this section. The example shows the differentiated services solution on the public intranet of BigISP, a fictitious service provider. BigISP offers services to large companies that reach BigISP through leased lines. Individuals who dial in from modems can also buy services from BigISP.

IPQoS Topology

The following figure shows the network topology that is used for BigISP's public intranet.

FIGURE 32–4 IPQoS Example Topology



BigISP has implemented these four tiers in its public intranet:

- **Tier 0** – Network `10.10.0.0` includes a large Diffserv router that is called `Bigrouter`, which has both external and internal interfaces. Several companies, including a large organization that is called Goldco, have rented leased-line services that terminate at `Bigrouter`. Tier 0 also handles individual customers who call over telephone lines or ISDN.
- **Tier 1** – Network `10.11.0.0` provides web services. The `Goldweb` server hosts the web site which was purchased by Goldco as part of the premium service that Goldco has purchased from BigISP. The server `Userweb` hosts small web sites that were purchased by individual customers. Both `Goldweb` and `Userweb` are IPQoS enabled.

- **Tier 2** – Network `10.12.0.0` provides applications for all customers to use. `BigAPPS`, one of the application servers, is IPQoS-enabled. `BigAPPS` provides SMTP, News, and FTP services.
- **Tier 3** – Network `10.13.0.0` houses large database servers. Access to Tier 3 is controlled by `datarouter`, a Diffserv router.

◆ ◆ ◆ CHAPTER 33

Creating the IPQoS Configuration File (Tasks)

This chapter shows how to create IPQoS configuration files. Topics that are covered in the chapter include the following.

This chapter assumes that you have defined a complete QoS policy, and you are ready to use this policy as the basis for the IPQoS configuration file. For instructions on QoS policy planning, refer to "Planning the Quality-of-Service Policy" on page 625.

Defining a QoS Policy in the IPQoS Configuration File (Task Map)

This task map lists the general tasks for creating an IPQoS configuration file and the links to the sections that describe the steps to perform the tasks.

Task	Description	For Instructions
1. Plan your IPQoS-enabled network configuration.	Decide which systems on the local network should become IPQoS enabled.	"How to Prepare a Network for IPQoS" on page 627
2. Plan the QoS policy for IPQoS systems on your network.	Identify traffic flows as distinct classes of service. Then, determine which flows require traffic management.	"Planning the Quality-of-Service Policy" on page 625

Task	Description	For Instructions
3. Create the IPQoS configuration file and define its first action.	Create the IPQoS file, invoke the IP classifier, and define a class for processing.	"How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645
4. Create filters for a class.	Add the filters that govern which traffic is selected and organized into a class.	"How to Define Filters in the IPQoS Configuration File" on page 647
5. Add more classes and filters to the IPQoS configuration file.	Create more classes and filters to be processed by the IP classifier.	"How to Create an IPQoS Configuration File for a Best-Effort Web Server" on page 653
6. Add an `action` statement with parameters that configure the metering modules.	If the QoS policy calls for flow control, assign flow-control rates and conformance levels to the meter.	"How to Configure Flow Control in the IPQoS Configuration File" on page 662
7. Add an `action` statement with parameters that configure the marker.	If the QoS policy calls for differentiated forwarding behaviors, define how traffic classes are to be forwarded.	"How to Define Traffic Forwarding in the IPQoS Configuration File" on page 649
8. Add an `action` statement with parameters that configure the flow-accounting module.	If the QoS policy calls for statistics gathering on traffic flows, define how accounting statistics are to be gathered.	"How to Enable Accounting for a Class in the IPQoS Configuration File" on page 652
9. Apply the IPQoS configuration file.	Add the content of a specified IPQoS configuration file into the appropriate kernel modules.	"How to Apply a New Configuration to the IPQoS Kernel Modules" on page 668
10. Configure forwarding behaviors in the router files.	If any IPQoS configuration files on the network define forwarding behaviors, add the resulting DSCPs to the appropriate scheduling files on the router.	"How to Configure a Router on an IPQoS-Enabled Network" on page 665

Tools for Creating a QoS Policy

The QoS policy for your network resides in the IPQoS configuration file. You create this configuration file with a text editor. Then, you provide the file as an argument to `ipqosconf`, the IPQoS configuration utility. When you instruct `ipqosconf` to apply the policy that is defined in your configuration file, the policy is written into the kernel IPQoS system. For detailed information about the `ipqosconf` command, refer to the `ipqosconf`(1M) man page. For instructions on the use of `ipqosconf`, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 668.

Basic IPQoS Configuration File

An IPQoS configuration file consists of a tree of `action` statements that implement the QoS policy that you defined in "Planning the Quality-of-Service Policy" on page 625. The IPQoS configuration file configures the IPQoS modules. Each action statement contains a set of *classes*, *filters*, or *parameters* to be processed by the module that is called in the action statement.

For the complete syntax of the IPQoS configuration file, refer to Example 36–3 and the `ipqosconf`(1M) man page.

Configuring the IPQoS Example Topology

The tasks in this chapter explain how to create IPQoS configuration files for three IPQoS-enabled systems. These systems are part of the network topology of the company BigISP, which was introduced in Figure 32–4.

- `Goldweb` – A web server that hosts web sites for customers who have purchased premium-level SLAs
- `Userweb` – A less-powerful web server that hosts personal web sites for home users who have purchased "best-effort" SLAs
- `BigAPPS` – An application server that serves mail, network news, and FTP to both gold-level and best-effort customers

These three configuration files illustrate the most common IPQoS configurations. You might use the sample files that are shown in the next section as templates for your own IPQoS implementation.

Creating IPQoS Configuration Files for Web Servers

This section introduces the IPQoS configuration file by showing how to create a configuration for a premium web server. The section then shows how to configure a completely different level of service in another configuration file for a server that hosts personal web sites. Both servers are part of the network example that is shown in Figure 32–4.

The following configuration file defines IPQoS activities for the `Goldweb` server. This server hosts the web site for Goldco, the company that has purchased a premium SLA.

EXAMPLE 33–1 Sample IPQoS Configuration File for a Premium Web Server

```
fmt_version 1.0

action {
    module ipgpc
    name ipgpc.classify
    params {
        global_stats TRUE
```

EXAMPLE 33–1 Sample IPQoS Configuration File for a Premium Web Server *(Continued)*

```
    }
    class {
        name goldweb
        next_action markAF11
        enable_stats FALSE
    }
    class {
        name video
        next_action markEF
        enable_stats FALSE
    }
    filter {
        name webout
        sport 80
        direction LOCAL_OUT
        class goldweb
    }
    filter {
        name videoout
        sport videosrv
        direction LOCAL_OUT
        class video
    }
}
action {
    module dscpmk
    name markAF11
    params {
        global_stats FALSE
        dscp_map{0-63:10}
        next_action continue
    }
}
action {
    module dscpmk
    name markEF
    params {
        global_stats TRUE
        dscp_map{0-63:46}
        next_action acct
    }
}
action {
    module flowacct
    name acct
    params {
        enable_stats TRUE
        timer 10000
        timeout 10000
        max_limit 2048
    }
}
```

The following configuration file defines IPQoS activities on `Userweb`. This server hosts web sites for individuals with low-priced, or *best-effort*, SLAs. This level of service guarantees the best service that can be delivered to best-effort customers after the IPQoS system handles traffic from customers with more expensive SLAs.

EXAMPLE 33–2 Sample Configuration for a Best-Effort Web Server

```
fmt_version 1.0

action {
    module ipgpc
    name ipgpc.classify
    params {
        global_stats TRUE
    }
    class {
        name Userweb
        next_action markAF12
        enable_stats FALSE
    }
    filter {
        name webout
        sport 80
        direction LOCAL_OUT
        class Userweb
   }
}
action {
    module dscpmk
    name markAF12
    params {
        global_stats FALSE
        dscp_map{0-63:12}
        next_action continue
    }
}
```

How to Create the IPQoS Configuration File and Define Traffic Classes

You can create your first IPQoS configuration file in whatever directory is easiest for you to maintain. The tasks in this chapter use the directory `/var/ipqos` as the location for IPQoS configuration files. The next procedure builds the initial segment of the IPQoS configuration file that is introduced in Example 33–1.

Note – As you create the IPQoS configuration file, be very careful to start and end each `action` statement and clause with curly braces ({ }). For an example of the use of braces, see Example 33–1.

---

1 **Log in to the premium web server, and create a new IPQoS configuration file with a `.qos` extension.**

Every IPQoS configuration file must start with the version number `fmt_version 1.0` as its first uncommented line.

2 **Follow the opening parameter with the initial `action` statement, which configures the generic IP classifier `ipgpc`.**

This initial action begins the tree of `action` statements that compose the IPQoS configuration file. For example, the `/var/ipqos/Goldweb.qos` file begins with the initial `action` statement to call the `ipgpc` classifier.

```
fmt_version 1.0

action {
    module ipgpc
    name ipgpc.classify
```

`fmt_version 1.0`	Begins the IPQoS configuration file.
`action {`	Begins the action statement.
`module ipgpc`	Configures the `ipgpc` classifier as the first action in the configuration file.
`name ipgpc.classify`	Defines the name of the classifier `action` statement, which must always be `ipgpc.classify`.

For detailed syntactical information about `action` statements, refer to "`action` Statement" on page 694 and the `ipqosconf`(1M) man page.

3 **Add a `params` clause with the statistics parameter `global_stats`.**

```
params {
         global_stats TRUE
   }
```

The parameter `global_stats TRUE` in the`ipgpc.classify` statement enables statistics gathering for that action. `global_stats TRUE` also enables per-class statistics gathering wherever a class clause definition specifies `enable_stats TRUE`.

Turning on statistics impacts performance. You might want to gather statistics on a new IPQoS configuration file to verify that IPQoS works properly. Later, you can turn off statistics collection by changing the argument to `global_stats` to `FALSE`.

Global statistics are but one type of parameter you can define in a `params` clause. For syntactical and other details about `params` clauses, refer to "`params` Clause" on page 696 and the `ipqosconf`(1M) man page.

4 **Define a class that identifies traffic that is bound for the premium server.**

```
class {
        name goldweb
```

```
        next_action markAF11
        enable_stats FALSE
    }
```

This statement is called a *class clause*. A `class` clause has the following contents.

name goldweb	Creates the class `goldweb` to identify traffic that is bound for the `Goldweb` server.
next_action markAF11	Instructs the `ipgpc` module to pass packets of the `goldweb` class to the `markAF11` action statement. The `markAF11` action statement calls the `dscpmk` marker.
enable_stats FALSE	Enables statistics taking for the `goldweb` class. However, because the value of `enable_stats` is `FALSE`, statistics for this class are not turned on.

For detailed information about the syntax of the `class` clause, see "`class` Clause" on page 695 and the `ipqosconf`(1M) man page.

5 Define a class that identifies an application that must have highest-priority forwarding.

```
class {
        name video
        next_action markEF
        enable_stats FALSE
    }
```

`name video`	Creates the class `video` to identify streaming video traffic that is outgoing from the `Goldweb` server.
`next_action markEF`	Instructs the `ipgpc` module to pass packets of the `video` class to the `markEF` statement after `ipgpc` completes processing. The `markEF` statement calls the `dscpmk` marker.
`enable_stats FALSE`	Enables statistics collection for the `video` class. However, because the value of `enable_stats` is `FALSE`, statistics collection for this class is not turned on.

See Also

- To define filters for the class you just created, refer to "How to Define Filters in the IPQoS Configuration File" on page 647.
- To create another class clause for the configuration file, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645.

▼ How to Define Filters in the IPQoS Configuration File

The next procedure shows how to define filters for a class in the IPQoS configuration file.

Before You Begin The procedure assumes that you have already started file creation and have defined classes. The steps continue building the `/var/ipqos/Goldweb.qos` file that is created in "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645.

Note – As you create the IPQoS configuration file, be very careful to start and end each `class` clause and each `filter` clause with curly braces ({ }). For an example of the use of braces, use Example 33–1.

---

1 Open the IPQoS configuration file, and locate the end of the last class that you defined.

For example, on the IPQoS-enabled server `Goldweb`, you would start after the following `class` clause in `/var/ipqos/Goldweb.qos`:

```
class {
        name video
        next_action markEF
        enable_stats FALSE
    }
```

2 Define a `filter` clause to select outgoing traffic from the IPQoS system.

```
filter {
    name webout
    sport 80
    direction LOCAL_OUT
    class goldweb
}
```

`name webout`	Gives the name `webout` to the filter.
`sport 80`	Selects traffic with a source port of 80, the well-known port for HTTP (web) traffic.
`direction LOCAL_OUT`	Further selects traffic that is outgoing from the local system.
`class goldweb`	Identifies the class to which the filter belongs, in this instance, class `goldweb`.

For syntactical and detailed information about the `filter` clause in the IPQoS configuration file, refer to "`filter` Clause" on page 696.

3 Define a `filter` clause to select streaming video traffic on the IPQoS system.

```
filter {
    name videoout
    sport videosrv
    direction LOCAL_OUT
    class video
}
```

name videoout	Gives the name `videoout` to the filter.

`sport videosrv`	Selects traffic with a source port of `videosrv`, a previously defined port for the streaming video application on this system.
`direction LOCAL_OUT`	Further selects traffic that is outgoing from the local system.
`class video`	Identifies the class to which the filter belongs, in this instance, class `video`.

See Also

- To define forwarding behaviors for the marker modules, refer to "How to Define Traffic Forwarding in the IPQoS Configuration File" on page 649.
- To define flow-control parameters for the metering modules, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 662.
- To activate the IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 668.
- To define additional filters, refer to "How to Define Filters in the IPQoS Configuration File" on page 647.
- To create classes for traffic flows from applications, refer to "How to Configure the IPQoS Configuration File for an Application Server" on page 658.

▼ How to Define Traffic Forwarding in the IPQoS Configuration File

The next procedure shows how to define traffic forwarding by adding per-hop behaviors for a class into the IPQoS configuration file.

Before You Begin The procedure assumes that you have an existing IPQoS configuration file with already defined classes and already defined filters. The steps continue building the `/var/ipqos/Goldweb.qos` file from Example 33–1.

Note – The procedure shows how to configure traffic forwarding by using the `dscpmk` marker module. For information about traffic forwarding on VLAN systems by using the `dlclosmk` marker, refer to "Using the `dlcosmk` Marker With VLAN Devices" on page 689.

---

1 Open the IPQoS configuration file, and locate the end of the last filter you defined.

For example, on the IPQoS-enabled server `Goldweb`, you would start after the following `filter` clause in `/var/ipqos/Goldweb.qos`:

```
filter {
        name videoout
        sport videosrv
        direction LOCAL_OUT
        class video
```

```
    }
}
```

Note that this `filter` clause is at the end of the `ipgpc` classifier `action` statement. Therefore, you need a closing brace to terminate the filter and a second closing brace to terminate the `action` statement.

2 Invoke the marker with the following `action` statement.

```
action {
    module dscpmk
    name markAF11
```

module dscpmk — Calls the marker module `dscpmk`.

name markAF11 — Gives the name `markAF11` to the `action` statement.

The previously defined class `goldweb` includes a `next_action` `markAF11` statement. This statement sends traffic flows to the `markAF11` action statement after the classifier concludes processing.

3 Define actions for the marker to take on the traffic flow.

```
    params {
        global_stats FALSE
        dscp_map{0-63:10}
        next_action continue
    }
}
```

global_stats FALSE — Enables statistics collection for the `markAF11` marker `action` statement. However, because the value of `enable_stats` is `FALSE`, statistics are not collected.

dscp_map{0–63:10} — Assigns a DSCP of `10` to the packet headers of the traffic class `goldweb`, which is currently being processed by the marker.

next_action continue — Indicates that no further processing is required on packets of the traffic class `goldweb`, and that these packets can return to the network stream.

The DSCP of `10` instructs the marker to set all entries in the `dscp` map to the decimal value 10 (binary 001010). This codepoint indicates that packets of the `goldweb` traffic class are subject to the AF11 per-hop behavior. AF11 guarantees that all packets with the DSCP of `10` receive a low-drop, high-priority service. Thus, outgoing traffic for premium customers on `Goldweb` is given the highest priority that is available for the Assured Forwarding (AF) PHB. For a table of possible DSCPs for AF, refer to Table 36–2.

4 Start another marker `action` statement.

```
action {
    module dscpmk
    name markEF
```

module dscpmk	Calls the marker module `dscpmk`.
`name markEF`	Gives the name `markEF` to the `action` statement.

5 Define actions for the marker to take on the traffic flow.

```
    params {
        global_stats TRUE
        dscp_map{0-63:46}
        next_action acct
    }
}
```

`global_stats TRUE`	Enables statistics collection on class `video`, which selects streaming video packets.
`dscp_map{0–63:46}`	Assigns a DSCP of 46 to the packet headers of the traffic class `video`, which is currently being processed by the marker.
`next_action acct`	Instructs the `dscpmk` module to pass packets of the class `video` to the `acct action` statement after `dscpmk` completes processing. The `acct action` statement invokes the `flowacct` module.

The DSCP of 46 instructs the `dscpmk` module to set all entries in the `dscp` map to the decimal value 46 (binary 101110) in the DS field. This codepoint indicates that packets of the `video` traffic class are subject to the Expedited Forwarding (EF) per-hop behavior.

Note – The recommended codepoint for EF is 46 (binary 101110). Other DSCPs assign AF PHBs to a packet.

The EF PHB guarantees that packets with the DSCP of 46 are given the highest precedence by IPQoS and Diffserv-aware systems. Streaming applications require highest-priority service, which is the rationale behind assigning to streaming applications the EF PHBs in the QoS policy. For more details about the expedited forwarding PHB, refer to "Expedited Forwarding (EF) PHB" on page 687.

6 Add the DSCPs that you have just created to the appropriate files on the Diffserv router.

For more information, refer to "How to Configure a Router on an IPQoS-Enabled Network" on page 665.

See Also

- To start gathering flow-accounting statistics on traffic flows, refer to "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 652.
- To define forwarding behaviors for the marker modules, refer to "How to Define Traffic Forwarding in the IPQoS Configuration File" on page 649.
- To define flow-control parameters for the metering modules, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 662.

- To activate the IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 668.
- To define additional filters, refer to "How to Define Filters in the IPQoS Configuration File" on page 647.
- To create classes for traffic flows from applications, refer to "How to Configure the IPQoS Configuration File for an Application Server" on page 658.

▼ How to Enable Accounting for a Class in the IPQoS Configuration File

The next procedure shows how to enable accounting on a traffic class in the IPQoS configuration file. The procedure shows how to define flow accounting for the `video` class, which is introduced in "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645. This class selects streaming video traffic, which must be billed as part of a premium customer's SLA.

Before You Begin The procedure assumes that you have an existing IPQoS configuration file with already defined classes, filters, metering actions, if appropriate, and marking actions, if appropriate. The steps continue building the `/var/ipqos/Goldweb.qos` file from Example 33–1.

1 Open the IPQoS configuration file, and locate the end of the last `action` statement you defined.

For example, on the IPQoS-enabled server `Goldweb`, you would start after the following `markEF` `action` statement in `/var/ipqos/Goldweb.qos`.

```
action {
    module dscpmk
    name markEF
    params {
        global_stats TRUE
        dscp_map{0-63:46}
        next_action acct
    }
}
```

2 Begin an `action` statement that calls flow accounting.

```
action {
    module flowacct
    name acct
```

`module flowacct`	Invokes the flow-accounting module `flowacct`.
`name acct`	Gives the name `acct` to the `action` statement

3 Define a `params` clause to control accounting on the traffic class.

```
params {
        global_stats TRUE
```

```
        timer 10000
        timeout 10000
        max_limit 2048
        next_action continue
    }
}
```

`global_stats TRUE`	Enables statistics collection on the class `video`, which selects streaming video packets.
`timer 10000`	Specifies the duration of the interval, in milliseconds, when the flow table is scanned for timed-out flows. In this parameter, that interval is 10000 milliseconds.
`timeout 10000`	Specifies the minimum interval time out value. A flow "times out" when packets for the flow are not seen during a time out interval. In this parameter, packets time out after 10000 milliseconds.
`max_limit 2048`	Sets the maximum number of active flow records in the flow table for this action instance.
`next_action continue`	Indicates that no further processing is required on packets of the traffic class `video`, and that these packets can return to the network stream.

The `flowacct` module gathers statistical information on packet flows of a particular class until a specified `timeout` value is reached.

See Also

- To configure per-hop behaviors on a router, refer to "How to Configure a Router on an IPQoS-Enabled Network" on page 665.
- To activate the IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 668.
- To create classes for traffic flows from applications, refer to "How to Configure the IPQoS Configuration File for an Application Server" on page 658.

▼ How to Create an IPQoS Configuration File for a Best-Effort Web Server

The IPQoS configuration file for a best-effort web server differs slightly from an IPQoS configuration file for a premium web server. As an example, the procedure uses the configuration file from Example 33–2.

1 Log in to the best-effort web server.

2 Create a new IPQoS configuration file with a `.qos` extension.

```
fmt_vesion 1.0
action {
    module ipgpc
    name ipgpc.classify
    params {
        global_stats TRUE
   }
```

The `/var/ipqos/userweb.qos` file must begin with the partial `action` statement to invoke the `ipgpc` classifier. In addition, the `action` statement also has a params clause to turn on statistics collection. For an explanation of this `action` statement, see "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645.

3 Define a class that identifies traffic that is bound for the best-effort web server.

```
class {
        name userweb
        next_action markAF12
        enable_stats FALSE
    }
```

`name userweb`	Creates a class that is called `userweb` for forwarding web traffic from users.
`next_action markAF1`	Instructs the `ipgpc` module to pass packets of the `userweb` class to the `markAF12` `action` statement after `ipgpc` completes processing. The `markAF12` `action` statement invokes the `dscpmk` marker.
`enable_stats FALSE`	Enables statistics collection for the `userweb` class. However, because the value of `enable_stats` is `FALSE`, statistics collection for this class does not occur.

For an explanation of the `class` clause task, see "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645.

4 Define a `filter` clause to select traffic flows for the `userweb` class.

```
   filter {
       name webout
       sport 80
       direction LOCAL_OUT
       class userweb
   }
}
```

`name webout`	Gives the name `webout` to the filter.
`sport 80`	Selects traffic with a source port of 80, the well-known port for HTTP (web) traffic.
`direction LOCAL_OUT`	Further selects traffic that is outgoing from the local system.

class userweb	Identifies the class to which the filter belongs, in this instance, class `userweb`.

For an explanation of the `filter` clause task, see "How to Define Filters in the IPQoS Configuration File" on page 647.

5 Begin the `action` statement to invoke the `dscpmk` marker.

```
action {
    module dscpmk
    name markAF12
```

`module dscpmk`	Invokes the marker module `dscpmk`.
`name markAF12`	Gives the name `markAF12` to the `action` statement.

The previously defined class `userweb` includes a `next_action markAF12` statement. This statement sends traffic flows to the `markAF12 action` statement after the classifier concludes processing.

6 Define parameters for the marker to use for processing the traffic flow.

```
    params {
        global_stats FALSE
        dscp_map{0-63:12}
        next_action continue
    }
}
```

`global_stats FALSE`	Enables statistics collection for the `markAF12` marker `action` statement. However, because the value of `enable_stats` is `FALSE`, statistics collection does not occur.
`dscp_map{0–63:12}`	Assigns a DSCP of `12` to the packet headers of the traffic class `userweb`, which is currently being processed by the marker.
next_action continue	Indicates that no further processing is required on packets of the traffic class `userweb`, and that these packets can return to the network stream.

The DSCP of `12` instructs the marker to set all entries in the `dscp` map to the decimal value 12 (binary 001100). This codepoint indicates that packets of the `userweb` traffic class are subject to the AF12 per-hop behavior. AF12 guarantees that all packets with the DSCP of `12` in the DS field receive a medium-drop, high-priority service.

7 When you complete the IPQoS configuration file, apply the configuration.

See Also

- To add classes and other configuration for traffic flows from applications, refer to "How to Configure the IPQoS Configuration File for an Application Server" on page 658.

- To configure per-hop behaviors on a router, refer to "How to Configure a Router on an IPQoS-Enabled Network" on page 665.
- To activate your IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 668.

Creating an IPQoS Configuration File for an Application Server

This section explains how to create a configuration file for an application server that provides major applications to customers. The procedure uses as its example the `BigAPPS` server from Figure 32–4.

The following configuration file defines IPQoS activities for the `BigAPPS` server. This server hosts FTP, electronic mail (SMTP), and network news (NNTP) for customers.

EXAMPLE 33–3 Sample IPQoS Configuration File for an Application Server

```
fmt_version 1.0

action {
    module ipgpc
    name ipgpc.classify
    params {
        global_stats TRUE
    }
    class {
        name smtp
        enable_stats FALSE
        next_action markAF13
    }
    class {
        name news
        next_action markAF21
    }
    class {
        name ftp
        next_action meterftp
    }
    filter {
        name smtpout
        sport smtp
        class smtp
    }
    filter {
        name newsout
        sport nntp
        class news
    }
    filter {
        name ftpout
        sport ftp
        class ftp
    }
```

EXAMPLE 33–3 Sample IPQoS Configuration File for an Application Server *(Continued)*

```
   filter {
        name ftpdata
        sport ftp-data
        class ftp
    }
}
action {
    module dscpmk
    name markAF13
    params {
        global_stats FALSE
        dscp_map{0-63:14}
        next_action continue
    }
}
action {
    module dscpmk
    name markAF21
    params {
        global_stats FALSE
        dscp_map{0-63:18}
        next_action continue
    }
}
action {
    module tokenmt
    name meterftp
    params {
        committed_rate 50000000
        committed_burst 50000000
        red_action_name AF31
        green_action_name markAF22
        global_stats TRUE
    }
}
action {
    module dscpmk
    name markAF31
    params {
        global_stats TRUE
        dscp_map{0-63:26}
        next_action continue
    }
}
action {
    module dscpmk
    name markAF22
    params {
        global_stats TRUE
        dscp_map{0-63:20}
        next_action continue
    }
}
```

▼ How to Configure the IPQoS Configuration File for an Application Server

1 Log in to the IPQoS-enabled application server, and create a new IPQoS configuration file with a `.qos` extension.

For example, you would create the `/var/ipqos/BigAPPS.qos` file for the application server. Begin with the following required phrases to start the `action` statement that invokes the `ipgpc` classifier:

```
fmt_version 1.0

action {
    module ipgpc
    name ipgpc.classify
    params {
        global_stats TRUE
    }
```

For an explanation of the opening `action` statement, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645.

2 Create classes to select traffic from three applications on the BigAPPS server.

Add the class definitions after the opening `action` statement.

```
    class {
        name smtp
        enable_stats FALSE
        next_action markAF13
    }
    class {
        name news
        next_action markAF21
    }
    class {
        name ftp
        enable_stats TRUE
        next_action meterftp
    }
```

`name smtp`	Creates a class that is called `smtp`, which includes email traffic flows to be handled by the SMTP application
`enable_stats FALSE`	Enables statistics collection for the `smtp` class. However, because the value of `enable_stats` is `FALSE`, statistics for this class are not taken.
`next_action markAF13`	Instructs the `ipgpc` module to pass packets of the `smtp` class to the `markAF13` `action` statement after `ipgpc` completes processing.
`name news`	Creates a class that is called `news`, which includes network news traffic flows to be handled by the NNTP application.

`next_action markAF21`	Instructs the `ipgpc` module to pass packets of the `news` class to the `markAF21` action statement after `ipgpc` completes processing.
`name ftp`	Creates a class that is called `ftp`, which handles outgoing traffic that is handled by the FTP application.
`enable_stats TRUE`	Enables statistics collection for the `ftp` class.
`next_action meterftp`	Instructs the `ipgpc` module to pass packets of the `ftp` class to the `meterftp action` statement after `ipgpc` completes processing.

For more information about defining classes, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645.

3 Define `filter` clauses to select traffic of the classes defined in Step 2.

```
    filter {
        name smtpout
        sport smtp
        class smtp
    }
    filter {
        name newsout
        sport nntp
        class news
    }
        filter {
        name ftpout
        sport ftp
        class ftp
    }
        filter {
        name ftpdata
        sport ftp-data
        class ftp
    }
}
```

`name smtpout`	Gives the name `smtpout` to the filter.
`sport smtp`	Selects traffic with a source port of 25, the well-known port for the `sendmail` (SMTP) application.
`class smtp`	Identifies the class to which the filter belongs, in this instance, class `smtp`.
`name newsout`	Gives the name `newsout` to the filter.
`sport nntp`	Selects traffic with a source port name of `nntp`, the well-known port name for the network news (NNTP) application.
`class news`	Identifies the class to which the filter belongs, in this instance, class `news`.
`name ftpout`	Gives the name `ftpout` to the filter.

`sport ftp`	Selects control data with a source port of 21, the well-known port number for FTP traffic.
`name ftpdata`	Gives the name `ftpdata` to the filter.
`sport ftp-data`	Selects traffic with a source port of 20, the well-known port number for FTP data traffic.
`class ftp`	Identifies the class to which the `ftpout` and `ftpdata` filters belong, in this instance `ftp`.

See Also

- To define filters, refer to "How to Define Filters in the IPQoS Configuration File" on page 647.
- To define forwarding behaviors for application traffic, refer to "How to Configure Forwarding for Application Traffic in the IPQoS Configuration File" on page 660.
- To configure flow control by using the metering modules, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 662.
- To configure flow accounting, refer to "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 652.

▼ How to Configure Forwarding for Application Traffic in the IPQoS Configuration File

The next procedure shows how to configure forwarding for application traffic. In the procedure, you define per-hop behaviors for application traffic classes that might have lower precedence than other traffic on a network. The steps continue building the `/var/ipqos/BigAPPS.qos` file in Example 33–3.

Before You Begin The procedure assumes that you have an existing IPQoS configuration file with already-defined classes and already-defined filters for the applications to be marked.

1 Open the IPQoS configuration file that you have created for the application server, and locate the end of the last `filter` clause.

In the `/var/ipqos/BigAPPS.qos` file, the last filter is the following:

```
 filter {
        name ftpdata
        sport ftp-data
        class ftp
    }
}
```

2 Invoke the marker as follows:

```
action {
    module dscpmk
    name markAF13
```

`module dscpmk` Invokes the marker module `dscpmk`.

`name markAF13` Gives the name `markAF13` to the `action` statement.

3 Define the per-hop behavior to be marked on electronic mail traffic flows.

```
    params {
        global_stats FALSE
        dscp_map{0-63:14}
        next_action continue
    }
}
```

`global_stats FALSE` Enables statistics collection for the `markAF13` marker `action` statement. However, because the value of `enable_stats` is `FALSE`, statistics are not collected.

`dscp_map{0–63:14}` Assigns a DSCP of `14` to the packet headers of the traffic class `smtp`, which is currently being processed by the marker.

`next_action continue` Indicates that no further processing is required on packets of the traffic class `smtp`. These packets can then return to the network stream.

The DSCP of `14` tells the marker to set all entries in the `dscp` map to the decimal value 14 (binary 001110). The DSCP of `14` sets the AF13 per-hop behavior. The marker marks packets of the `smtp` traffic class with the DSCP of `14` in the DS field.

AF13 assigns all packets with a DSCP of `14` to a high-drop precedence. However, because AF13 also assures a Class 1 priority, the router still guarantees outgoing email traffic a high priority in its queue. For a table of possible AF codepoints, refer to Table 36–2.

4 Add a marker `action` statement to define a per-hop behavior for network news traffic:

```
action {
    module dscpmk
    name markAF21
    params {
        global_stats FALSE
        dscp_map{0-63:18}
        next_action continue
    }
}
```

`name markAF21` Gives the name `markAF21` to the `action` statement.

`dscp_map{0–63:18}` Assigns a DSCP of `18` to the packet headers of the traffic class `nntp`, which is currently being processed by the marker.

The DSCP of 18 tells the marker to set all entries in the `dscp` map to the decimal value 18 (binary 010010). The DSCP of 18 sets the AF21 per-hop behavior. The marker marks packets of the `news` traffic class with the DSCP of 18 in the DS field.

AF21 assures that all packets with a DSCP of 18 receive a low-drop precedence, but with only Class 2 priority. Thus, the possibility of network news traffic being dropped is low.

See Also

- To add configuration information for web servers, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645.
- To configure flow control by using the metering modules, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 662.
- To configure flow accounting, refer to "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 652.
- To configure forwarding behaviors on a router, refer to "How to Configure a Router on an IPQoS-Enabled Network" on page 665.
- To activate the IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 668.

▼ How to Configure Flow Control in the IPQoS Configuration File

To control the rate at which a particular traffic flow is released onto the network, you must define parameters for the meter. You can use either of the two meter modules, `tokenmt` or `tswtclmt`, in the IPQoS configuration file.

The next procedure continues to build the IPQoS configuration file for the application server in Example 33–3. In the procedure, you configure not only the meter but also two marker actions that are called within the meter `action` statement.

Before You Begin The steps assume that you have already defined a class and a filter for the application to be flow-controlled.

1 Open the IPQoS configuration file that you have created for the applications server.

In the `/var/ipqos/BigAPPS.qos` file, you begin after the following marker action:

```
action {
    module dscpmk
    name markAF21
    params {
        global_stats FALSE
        dscp_map{0-63:18}
        next_action continue
    }
}
```

2 Create a meter `action` statement to flow-control traffic of the `ftp` class.

```
action {
    module tokenmt
    name meterftp
```

`module tokenmt` Invokes the `tokenmt` meter.

`name meterftp` Gives the name `meterftp` to the `action` statement.

3 Add parameters to configure the meter's rate.

```
params {
       committed_rate 50000000
       committed_burst 50000000
```

`committed_rate 50000000` Assigns a transmission rate of 50,000,000 bps to traffic of the `ftp` class.

`committed_burst 50000000` Commits a burst size of 50,000,000 bits to traffic of the `ftp` class.

For an explanation of `tokenmt` parameters, refer to "Configuring `tokenmt` as a Two-Rate Meter" on page 685.

4 Add parameters to configure traffic conformance precedences:

```
    red_action markAF31
    green_action_name markAF22
    global_stats TRUE
    }
}
```

`red_action_name markAF31` Indicates that when the traffic flow of the `ftp` class exceeds the committed rate, packets are sent to the `markAF31` marker `action` statement.

`green_action_name markAF22` Indicates that when traffic flows of class `ftp` conform to the committed rate, packets are sent to the `markAF22` action statement.

`global_stats TRUE` Enables metering statistics for the `ftp` class.

For more information about traffic conformance, see "Meter Module" on page 684.

5 Add a marker `action` statement to assign a per-hop behavior to nonconformant traffic flows of class `ftp`.

```
action {
    module dscpmk
    name markAF31
    params {
        global_stats TRUE
        dscp_map{0-63:26}
```

```
        next_action continue
    }
}
```

`module dscpmk`	Invokes the marker module `dscpmk`.
`name markAF31`	Gives the name `markAF31` to the `action` statement.
`global_stats TRUE`	Enables statistics for the `ftp` class.
`dscp_map{0–63:26}`	Assigns a DSCP of `26` to the packet headers of the traffic class `ftp` whenever this traffic exceeds the committed rate.
`next_action continue`	Indicates that no further processing is required on packets of the traffic class `ftp`. Then these packets can return to the network stream.

The DSCP of `26` instructs the marker to set all entries in the `dscp` map to the decimal value 26 (binary 011010). The DSCP of `26` sets the AF31 per-hop behavior. The marker marks packets of the `ftp` traffic class with the DSCP of `26` in the DS field.

AF31 assures that all packets with a DSCP of `26` receive a low-drop precedence, but with only Class 3 priority. Therefore, the possibility of nonconformant FTP traffic being dropped is low. For a table of possible AF codepoints, refer to Table 36–2.

6 Add a marker `action` statement to assign a per-hop behavior to `ftp` traffic flows that conform to the committed rate.

```
action {
    module dscpmk
    name markAF22
    params {
        global_stats TRUE
        dscp_map{0-63:20}
        next_action continue
    }
}
```

`name markAF22`	Gives the name `markAF22` to the `marker` action.
`dscp_map{0–63:20}`	Assigns a DSCP of `20` to the packet headers of the traffic class `ftp` whenever `ftp` traffic conforms to its configured rate.

The DSCP of 20 tells the marker to set all entries in the `dscp` map to the decimal value 20 (binary 010100). The DSCP of `20` sets the AF22 per-hop behavior. The marker marks packets of the `ftp` traffic class with the DSCP of `20` in the DS field.

AF22 assures that all packets with a DSCP of `20` receive a medium-drop precedence with Class 2 priority. Therefore, conformant FTP traffic is assured a medium-drop precedence among flows that are simultaneously released by the IPQoS system. However, the router gives a higher forwarding priority to traffic classes with a Class 1 medium-drop precedence mark or higher. For a table of possible AF codepoints, refer to Table 36–2.

7 Add the DSCPs that you have created for the application server to the appropriate files on the Diffserv router.

See Also
- To activate the IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 668.
- To add configuration information for web servers, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645.
- To configure flow accounting, refer to "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 652.
- To configure forwarding behaviors on a router, refer to "How to Configure a Router on an IPQoS-Enabled Network" on page 665.

Providing Differentiated Services on a Router

To provide true differentiated services, you must include a Diffserv-aware router in your network topology, as described in "Hardware Strategies for the Diffserv Network" on page 622. The actual steps for configuring Diffserv on a router and updating that router's files are outside the scope of this guide.

This section gives general steps for coordinating the forwarding information among various IPQoS-enabled systems on the network and the Diffserv router.

▼ How to Configure a Router on an IPQoS-Enabled Network

The next procedure uses as its example the topology in Figure 32–4.

Before You Begin The next procedure assumes that you have already configured the IPQoS systems on your network by performing the previous tasks in this chapter.

1 Review the configuration files for all IPQoS-enabled systems on your network.

2 Identify each codepoint that is used in the QoS various policies.

List the codepoints, and the systems and classes, to which the codepoints apply. The next table can illustrate areas where you might have used the same codepoint. This practice is acceptable. However, you should provide other criteria in the IPQoS configuration file, such as a `precedence` selector, to determine the precedence of identically marked classes.

For example, for the sample network that is used in the procedures throughout this chapter, you might construct the following codepoint table.

System	Class	PHB	DS Codepoint
Goldweb	video	EF	46 (101110)
Goldweb	goldweb	AF11	10 (001010)
Userweb	webout	AF12	12 (001100)
BigAPPS	smtp	AF13	14 (001110)
BigAPPS	news	AF18	18 (010010)
BigAPPS	ftp conformant traffic	AF22	20 (010100)
BigAPPS	ftp nonconformant traffic	AF31	26 (011010)

3 Add the codepoints from your network's IPQoS configuration files to the appropriate files on the Diffserv router.

The codepoints that you supply should help to configure the router's Diffserv scheduling mechanism. Refer to the router manufacturer's documentation and web sites for instructions.

◆ ◆ ◆ CHAPTER 34

Starting and Maintaining IPQoS (Tasks)

This chapter contains tasks for activating an IPQoS configuration file and for logging IPQoS-related events. The following topics are covered:

Administering IPQoS (Task Map)

This section lists the set of tasks for starting and maintaining IPQoS on an Oracle Solaris system. Before you use the tasks, you must have a completed IPQoS configuration file, as described in "Defining a QoS Policy in the IPQoS Configuration File (Task Map)" on page 641.

The following table itemizes and describes those tasks and contains links to the sections that detail how to complete these tasks.

Task	Description	For Instructions
1. Configure IPQoS on a system.	Use the `ipqosconf` command to activate the IPQoS configuration file on a system.	"How to Apply a New Configuration to the IPQoS Kernel Modules" on page 668
2. Make the Oracle Solaris startup scripts apply the debugged IPQoS configuration file after each system boot.	Ensure that the IPQoS configuration is applied each time the system reboots.	"How to Ensure That the IPQoS Configuration Is Applied After Each Reboot" on page 669.
3. Enable `syslog` logging for IPQoS.	Add an entry to enable `syslog` logging of IPQoS messages.	"How to Enable Logging of IPQoS Messages During Booting" on page 669.
4. Fix any IPQoS problems that arise.	Troubleshoot IPQoS problems by using error messages.	Refer to the error messages in Table 34–1.

Applying an IPQoS Configuration

You activate and otherwise manipulate the IPQoS configuration by using the `ipqosconf` command.

▼ How to Apply a New Configuration to the IPQoS Kernel Modules

You use the `ipqosconf` command to read the IPQoS configuration file and to configure the IPQoS modules in the UNIX kernel. The next procedure uses as an example the file `/var/ipqos/Goldweb.qos`, which is created in "Creating IPQoS Configuration Files for Web Servers" on page 643. For detailed information, refer to the `ipqosconf`(1M) man page.

1 Apply the new configuration.

```
# /usr/sbin/ipqosconf -a/var/ipqos/Goldweb.qos
```

`ipqosconf` writes the information in the specified IPQoS configuration file into the IPQoS modules in the Oracle Solaris kernel. In this example, the contents of `/var/ipqos/Goldweb.qos` are applied to the current Oracle Solaris kernel.

Note – When you apply an IPQoS configuration file with the `-a` option, the actions in the file are active for the current session only.

---

2 Test and debug the new IPQoS configuration.

Use UNIX utilities to track IPQoS behavior and to gather statistics on your IPQoS implementation. This information can help you determine if the configuration operates as expected.

See Also

- To view statistics on how IPQoS modules are working, refer to "Gathering Statistical Information" on page 678.
- To log `ipqosconf` messages, refer to "Enabling `syslog` Logging for IPQoS Messages" on page 669.
- To ensure that the current IPQoS configuration is applied after each boot, refer to "How to Ensure That the IPQoS Configuration Is Applied After Each Reboot" on page 669.

▼ How to Ensure That the IPQoS Configuration Is Applied After Each Reboot

You must explicitly make an IPQoS configuration persistent across reboots. Otherwise, the current configuration applies only until the system reboots. When IPQoS works correctly on a system, do the following to make the configuration persistent across reboots.

1. **Test for the existence of an IPQoS configuration in the kernel modules.**

   ```
   # ipqosconf -l
   ```

   If a configuration already exists, `ipqosconf` displays the configuration on the screen. If you do not receive output, apply the configuration, as explained in "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 668.

2. **Ensure that the existing IPQoS configuration is applied every time the IPQoS system reboots.**

   ```
   # /usr/sbin/ipqosconf -c
   ```

   The `-c` option causes the current IPQoS configuration to be represented in the boot-time configuration file `/etc/inet/ipqosinit.conf`.

Enabling `syslog` Logging for IPQoS Messages

To record IPQoS boot-time messages, you need to modify the `/etc/syslog.conf` file as shown in the next procedure.

▼ How to Enable Logging of IPQoS Messages During Booting

1. **Open the `/etc/syslog.conf` file.**

2. **Add the following text as the final entry in the file.**

   ```
   user.info                       /var/adm/messages
   ```

   Use tabs rather than spaces between the columns.

   The entry logs all boot-time messages that are generated by IPQoS into the `/var/adm/messages` file.

3. **Reboot the system to apply the messages.**

Example 34–1 IPQoS Output From `/var/adm/messages`

When you view `/var/adm/messages` after system reboot, your output might contain IPQoS logging messages that are similar to the following.

```
May 14 10:44:33 ipqos-14 ipqosconf: [ID 815575 user.info]
 New configuration applied.
May 14 10:44:46 ipqos-14 ipqosconf: [ID 469457 user.info]
Current configuration saved to init file.
May 14 10:44:55 ipqos-14 ipqosconf: [ID 435810 user.info]
Configuration flushed.
```

You might also see IPQoS error messages that are similar to the following in your IPQoS system's `/var/adm/messages` file.

```
May 14 10:56:47 ipqos-14 ipqosconf: [ID 123217 user.error]
 Missing/Invalid config file fmt_version.
May 14 10:58:19 ipqos-14 ipqosconf: [ID 671991 user.error]
No ipgpc action defined.
```

For a description of these error messages, see Table 34–1.

Troubleshooting with IPQoS Error Messages

This section contains a table of error messages that are generated by IPQoS and their possible solutions.

TABLE 34–1 IPQoS Error Messages

Error Message	Description	Solution
`Undefined action in parameter` *parameter-name's* `action` *action-name*	In the IPQoS configuration file, the action name that you specified in *parameter-name* does not exist in the configuration file.	Create the action. Or, refer to a different, existing action in the parameter.
`action` *action-name* `involved in cycle`	In the IPQoS configuration file, *action-name* is part of a cycle of actions, which is not allowed by IPQoS.	Determine the action cycle. Then remove one of the cyclical references from the IPQoS configuration file.
`Action` *action-name* `isn't referenced by any other actions`	A non-`ipgpc` action definition is not referenced by any other defined actions in the IPQoS configuration, which is not allowed by IPQoS.	Remove the unreferenced action. Alternatively, make another action reference the currently unreferenced action.
`Missing/Invalid config file fmt_version`	The format of the configuration file is not specified as the first entry of the file, which is required by IPQoS.	Add the format version, as explained in "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645.

TABLE 34–1 IPQoS Error Messages *(Continued)*

Error Message	Description	Solution
`Unsupported config file format version`	The format version that is specified in the configuration file is not supported by IPQoS.	Change the format version to `fmt_version 1.0`, which is required to run the Solaris 9 9/02 and later versions of IPQoS.
`No ipgpc action defined.`	You did not define an action for the `ipgpc` classifier in the configuration file, which is an IPQoS requirement.	Define an action for `ipgpc`, as shown in "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 645.
`Can't commit a null configuration`	When you ran `ipqosconf -c` to commit a configuration, that configuration was empty, which IPQoS does not allow.	Be sure to apply a configuration file before you attempt to commit a configuration. For instructions, see "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 668.
`Invalid CIDR mask on line` *line-number*	In the configuration file, you used a CIDR mask as part of the IP address that is out of the valid range for IP addresses.	Change the mask value to be in the range of 1–32 for IPv4 and 1–128 for IPv6.
`Address masks aren't allowed for host names line` *line-number*	In the configuration file, you defined a CIDR mask for a host name, which is not allowed in IPQoS.	Remove the mask or change the host name to an IP address.
`Invalid module name line` *line-number*	In the configuration file, the module name that you specified in an action statement is invalid.	Check the spelling of the module name. For a list of IPQoS modules, refer to Table 36–5.
`ipgpc action has incorrect name line` *line-number*	The name that you gave to the `ipgpc` action in the configuration file is not the required `ipgpc.classify`.	Rename the action `ipgpc.classify`.
`Second parameter clause not supported line` *line-number*	In the configuration file, you specified two parameter clauses for a single action, which IPQoS does not allow.	Combine all parameters for the action into a single parameters clause.
`Duplicate named action`	In the configuration file, you gave the same name to two actions.	Rename or remove one of the actions.
`Duplicate named filter/class in action` *action-name*	You gave the same name to two filters or two classes in the same action, which is not allowed in the IPQoS configuration file.	Rename or remove one of the filters or classes.
`Undefined class in filter` *filter-name* `in action` *action-name*	In the configuration file, the filter references a class that is not defined in the action.	Create the class, or change the filter reference to an already existing class.
`Undefined action in class` *class-name* `action` *action-name*	The class refers to an action that is not defined in the configuration file.	Create the action, or change the reference to an already existing action.

TABLE 34–1 IPQoS Error Messages *(Continued)*

Error Message	Description	Solution
`Invalid parameters for action` *action-name*	In the configuration file, one of the parameters is invalid.	For the module that is called by the named action, refer to the module entry in "IPQoS Architecture and the Diffserv Model" on page 681. Alternatively, you can refer to the `ipqosconf`(1M) man page.
`Mandatory parameter missing for action` *action-name*	You have not defined a required parameter for an action in the configuration file.	For the module that is called by the named action, refer to the module entry in "IPQoS Architecture and the Diffserv Model" on page 681. Alternatively, you can refer to the `ipqosconf`(1M) man page.
`Max number of classes reached in ipgpc`	You specified more classes than are allowed in the `ipgpc` action of the IPQoS configuration file. The maximum number is 10007.	Review the configuration file, and remove unneeded classes. Alternatively, you can raise the maximum number of classes by adding to the `/etc/system` file the entry `ipgpc_max_classes`*class-number*.
`Max number of filters reached in action ipgpc`	You specified more filters than are allowed in the `ipgpc` action of the IPQoS configuration file. The maximum number is 10007.	Review the configuration file, and remove unneeded filters. Alternatively, you can raise the maximum number of filters by adding to the `/etc/system` file the entry `ipgpc_max_filters`*filter-number*.
`Invalid/missing parameters for filter` *filter-name* `in action ipgpc`	In the configuration file, filter *filter-name* has an invalid or missing parameter.	Refer to the `ipqosconf`(1M) man page for the list of valid parameters.
`Name not allowed to start with '!', line` *line-number*	You began an action, filter, or class name with an exclamation mark (!), which is not allowed in the IPQoS file.	Remove the exclamation mark, or rename the action, class, or filter.
`Name exceeds the maximum name length line` *line-number*	You defined a name for an action, class, or filter in the configuration file that exceeds the maximum length of 23 characters.	Give a shorter name to the action, class, or filter.
`Array declaration line` *line-number* `is invalid`	In the configuration file, the array declaration for the parameter on line *line-number* is invalid.	For the correct syntax of the array declaration that is called by the `action` statement with the invalid array, refer to "IPQoS Architecture and the Diffserv Model" on page 681. Alternatively, refer to the `ipqosconf`(1M) man page.
`Quoted string exceeds line,` *line-number*	The string does not have the terminating quotation marks on the same line, which is required in the configuration file.	Make sure that the quoted string begins and ends on the same line in the configuration file.
`Invalid value, line` *line-number*	The value that is given on *line-number* of the configuration file is not supported for the parameter.	For the acceptable values for the module that is called by the `action` statement, refer to the module description in "IPQoS Architecture and the Diffserv Model" on page 681. Alternatively, you can refer to the `ipqosconf`(1M) man page.

TABLE 34–1 IPQoS Error Messages *(Continued)*

Error Message	Description	Solution
`Unrecognized value, line` *line-number*	The value on *line-number* of the configuration file is not a supported enumeration value for its parameter.	Check that the enumeration value is correct for the parameter. For a description of the module that is called by the `action` statement with the unrecognized line number, refer to "IPQoS Architecture and the Diffserv Model" on page 681. Alternatively, you can refer to the `ipqosconf`(1M) man page.
`Malformed value list line` *line-number*	The enumeration that is specified on *line-number* of the configuration file does not conform to the specification syntax.	For correct syntax for the module that is called by the `action` statement with the malformed value list, refer to the module description in "IPQoS Architecture and the Diffserv Model" on page 681. Alternatively, you can refer to the `ipqosconf`(1M) man page.
`Duplicate parameter line` *line-number*	A duplicate parameter was specified on *line-number*, which is not allowed in the configuration file.	Remove one of the duplicate parameters.
`Invalid action name line` *line-number*	You gave the action on *line-number* of the configuration file a name that uses the predefined name "continue" or "drop."	Rename the action so that the action does not use a predefined name.
`Failed to resolve src/dst host name for filter at line` *line-number*`, ignoring filter`	`ipqosconf` could not resolve the source or destination address that was defined for the given filter in the configuration file. Therefore, the filter is ignored.	If the filter is important, try applying the configuration at a later time.
`Incompatible address version line` *line-number*	The IP version of the address on *line-number* is incompatible with the version of a previously specified IP address or `ip_version` parameter.	Change the two conflicting entries to be compatible.
`Action at line` *line-number* `has the same name as currently installed action, but is for a different module`	You tried to change the module of an action that already exists in the system's IPQoS configuration, which is not allowed.	Flush the current configuration before you apply the new configuration.

◆ ◆ ◆ CHAPTER 35

Using Flow Accounting and Statistics Gathering (Tasks)

This chapter explains how to obtain accounting and statistical information on traffic that is handled by an IPQoS system. The following topics are discussed:

Setting Up Flow Accounting (Task Map)

The following task map lists the generic tasks for obtaining information about traffic flows by using the `flowacct` module. The map also links to procedures to carry out these tasks.

Task	Description	For Instructions
1. Create a file to contain accounting information for traffic flows.	Use the `acctadm` command to create a file that holds the results of processing by `flowacct`.	"How to Create a File for Flow-Accounting Data" on page 676
2. Define `flowacct` parameters in the IPQoS configuration file.	Define values for the `timer`, `timeout`, and `max_limit` parameters.	"How to Enable Accounting for a Class in the IPQoS Configuration File" on page 652

Recording Information About Traffic Flows

You use the IPQoS `flowacct` module to collect information about traffic flows. For example, you can collect source and destination addresses, number of packets in a flow, and similar data. The process of accumulating and recording information about flows is called *flow accounting*.

The results of flow accounting on traffic of a particular class are recorded in a table of *flow records*. Each flow record consists of a series of attributes. These attributes contain data about traffic flows of a particular class over an interval of time. For a list of the `flowacct` attributes, refer to Table 36–4.

Flow accounting is particularly useful for billing clients as is defined in their service-level agreements (SLAs). You can also use flow accounting to obtain flow statistics for critical applications. This section contains tasks for using `flowacct` with the Oracle Solaris extended accounting facility to obtain data on traffic flows.

The following information is contained in sources outside this chapter:

- For instructions on creating an action statement for `flowacct` in the IPQoS configuration file, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 662.
- To learn how `flowacct` works, refer to "Classifier Module" on page 681.
- For technical information, refer to the `flowacct`(7ipp) man page.

How to Create a File for Flow-Accounting Data

Before you add a `flowacct` action to the IPQoS configuration file, you must create a file for flow records from the `flowacct` module. You use the `acctadm` command for this purpose. `acctadm` can record either basic attributes or extended attributes in the file. All `flowacct` attributes are listed in Table 36–4. For detailed information about `acctadm`, refer to the `acctadm`(1M) man page.

1 Create a basic flow-accounting file.

The following example shows how to create a basic flow-accounting file for the premium web server that is configured in Example 33–1.

```
# /usr/sbin/acctadm -e basic -f /var/ipqos/goldweb/account.info flow
```

`acctadm -e`	Invokes `acctadm` with the `-e` option. The `-e` option enables the arguments that follow.
`basic`	States that only data for the eight basic `flowacct` attributes is to be recorded in the file.
`/var/ipqos/goldweb/account.info`	Specifies the fully qualified path name of the file to hold the flow records from `flowacct`.
`flow`	Instructs `acctadm` to enable flow accounting.

2 View information about flow accounting on the IPQoS system by typing `acctadm` without arguments.

`acctadm` generates the following output:

```
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,mstate
            Flow accounting: active
       Flow accounting file: /var/ipqos/goldweb/account.info
     Tracked flow resources: basic
   Untracked flow resources: dsfield,ctime,lseen,projid,uid
```

All entries but the last four are for use with the Solaris Resource Manager feature. The next table explains the entries that are specific to IPQoS.

Entry	Description
`Flow accounting: active`	Indicates that flow accounting is turned on.
`Flow accounting file: /var/ipqos/goldweb/account.info`	Gives the name of the current flow-accounting file.
`Tracked flow resources: basic`	Indicates that only the basic flow attributes are tracked.
`Untracked flow resources: dsfield,ctime,lseen,projid,uid`	Lists the `flowacct` attributes that are not tracked in the file.

3 (Optional) Add the extended attributes to the accounting file.

```
# acctadm -e extended -f /var/ipqos/goldweb/account.info flow
```

4 (Optional) Return to recording only the basic attributes in the accounting file.

```
# acctadm -d extended -e basic -f /var/ipqos/goldweb/account.info
```

The `-d` option disables extended accounting.

5 View the contents of a flow-accounting file.

Instructions for viewing the contents of a flow-accounting file are in "Perl Interface to libexacct" in *System Administration Guide: Oracle Solaris Zones, Oracle Solaris 10 Containers, and Resource Management*.

See Also

- For detailed information on the extended accounting feature, refer to Chapter 4, "Extended Accounting (Overview)," in *System Administration Guide: Oracle Solaris Zones, Oracle Solaris 10 Containers, and Resource Management*.

- To define `flowacct` parameters in the IPQoS configuration file, refer to "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 652.
- To print the data in the file that was created with `acctadm`, refer to "Perl Interface to libexacct" in *System Administration Guide: Oracle Solaris Zones, Oracle Solaris 10 Containers, and Resource Management*.

Gathering Statistical Information

You can use the `kstat` command to generate statistical information from the IPQoS modules. Use the following syntax:

`/bin/kstat -m` *ipqos-module-name*

You can specify any valid IPQoS module name, as shown in Table 36–5. For example, to view statistics that are generated by the `dscpmk` marker, you use the following form of `kstat`:

```
/bin/kstat -m dscpmk
```

For technical details, refer to the `kstat`(1M) man page.

EXAMPLE 35–1 `kstat` Statistics for IPQoS

Here is an example of possible results from running `kstat` to obtain statistics about the `flowacct` module.

```
# kstat -m flowacct
module: flowacct                        instance: 3
name:   Flowacct statistics             class:    flacct
        bytes_in_tbl                    84
        crtime                          345728.504106363
        epackets                        0
        flows_in_tbl                    1
        nbytes                          84
        npackets                        1
        snaptime                        345774.031843301
        usedmem                         256
```

`class: flacct` — Gives the name of the class to which the traffic flows belong, in this example `flacct`.

`bytes_in_tbl` — Total number of bytes in the flow table. The total number of bytes is the sum in bytes of all the flow records that currently reside in the flow table. The total number of bytes for this flow table is 84. If no flows are in the table, the value for `bytes_in_tbl` is 0.

`crtime` — The last time that this `kstat` output was created.

`epackets` — Number of packets that resulted in an error during processing, in this example 0.

EXAMPLE 35–1 `kstat` Statistics for IPQoS *(Continued)*

`flows_in_tbl`	Number of flow records in the flow table, which in this example is 1. When no records are in the table, the value for `flows_in_tbl` is 0.
`nbytes`	Total number of bytes that are seen by this `flowacct` action instance, which is 84 in the example. The value includes bytes that are currently in the flow table. The value also includes bytes that have timed out and are no longer in the flow table.
`npackets`	Total number of packets that are seen by this `flowacct` action instance, which is 1 in the example. `npackets` includes packets that are currently in the flow table. `npackets` also includes packets that have timed out—are no longer in the flow table.
`usedmem`	Memory in bytes in use by the flow table that is maintained by this `flowacct` instance. The `usedmem` value is 256 in the example. The value for `usedmem` is 0 when the flow table does not have any flow records.

◆ ◆ ◆ CHAPTER 36

IPQoS in Detail (Reference)

This chapter contains reference materials that provide in-depth details about the following IPQoS topics:

For an overview, refer to Chapter 31, "Introducing IPQoS (Overview)." For planning information, refer to Chapter 32, "Planning for an IPQoS-Enabled Network (Tasks)." For procedures for configuring IPQoS, refer to Chapter 33, "Creating the IPQoS Configuration File (Tasks)."

IPQoS Architecture and the Diffserv Model

This section describes the IPQoS architecture and how IPQoS implements the differentiated services (Diffserv) model that is defined inRFC 2475, An Architecture for Differentiated Services (`http://www.ietf.org/rfc/rfc2475.txt?number=2475`). The following elements of the Diffserv model are included in IPQoS:

- Classifier
- Meter
- Marker

In addition, IPQoS includes the flow-accounting module and the `dlcosmk` marker for use with virtual local area network (VLAN) devices.

Classifier Module

In the Diffserv model, the *classifier* is responsible for organizing selected traffic flows into groups on which to apply different service levels. The classifiers that are defined in RFC 2475 were originally designed for boundary routers. In contrast, the IPQoS classifier `ipgpc` is

designed to handle traffic flows on hosts that are internal to the local network. Therefore, a network with both IPQoS systems and a Diffserv router can provide a greater degree of differentiated services. For a technical description of `ipgpc`, refer to the `ipgpc`(7ipp) man page.

The `ipgpc` classifier does the following:

1. Selects traffic flows that meet the criteria specified in the IPQoS configuration file on the IPQoS-enabled system

   The QoS policy defines various criteria that must be present in packet headers. These criteria are called *selectors*. The `ipgpc` classifier compares these selectors against the headers of packets that are received by the IPQoS system. `ipgpc` then selects all matching packets.
2. Separates the packet flows into *classes*, network traffic with the same characteristics, as defined in the IPQoS configuration file
3. Examines the value in the packet's differentiated service (DS) field for the presence of a differentiated services codepoint (DSCP)

   The presence of the DSCP indicates whether the incoming traffic has been marked by the sender with a forwarding behavior.
4. Determines what further action is specified in the IPQoS configuration file for packets of a particular class
5. Passes the packets to the next IPQoS module specified in the IPQoS configuration file, or returns the packets to the network stream

For an overview of the classifier, refer to "Classifier (`ipgpc`) Overview" on page 613. For information on invoking the classifier in the IPQoS configuration file, refer to "IPQoS Configuration File" on page 693.

IPQoS Selectors

The `ipgpc` classifier supports a variety of selectors that you can use in the `filter` clause of the IPQoS configuration file. When you define a filter, always use the minimum number of selectors that are needed to successfully retrieve traffic of a particular class. The number of filters you define can impact IPQoS performance.

The next table lists the selectors that are available for `ipgpc`.

TABLE 36–1 Filter Selectors for the IPQoS Classifier

Selector	Argument	Information Selected
`saddr`	IP address number.	Source address.
`daddr`	IP address number.	Destination address.
`sport`	Either a port number or service name, as defined in `/etc/services`.	Source port from which a traffic class originated.

TABLE 36–1 Filter Selectors for the IPQoS Classifier *(Continued)*

Selector	Argument	Information Selected
`dport`	Either a port number or service name, as defined in `/etc/services`.	Destination port to which a traffic class is bound.
`protocol`	Either a protocol number or protocol name, as defined in `/etc/protocols`.	Protocol to be used by this traffic class.
`dsfield`	DS codepoint (DSCP) with a value of 0–63.	DSCP, which defines any forwarding behavior to be applied to the packet. If this parameter is specified, the `dsfield_mask` parameter must also be specified.
`dsfield_mask`	Bit mask with a value of 0–255.	Used in tandem with the `dsfield` selector. `dsfield_mask` is applied to the `dsfield` selector to determine which of its bits to match against.
`if_name`	Interface name.	Interface to be used for either incoming or outgoing traffic of a particular class.
`user`	Number of the UNIX user ID or user name to be selected. If no user ID or user name is on the packet, the default –1 is used.	User ID that is supplied to an application.
`projid`	Number of the project ID to be selected.	Project ID that is supplied to an application.
`priority`	Priority number. Lowest priority is 0.	Priority that is given to packets of this class. Priority is used to order the importance of filters for the same class.
`direction`	Argument can be one of the following:	Direction of packet flow on the IPQoS machine.
	`LOCAL_IN`	Input traffic local to the IPQoS system.
	`LOCAL_OUT`	Output traffic local to the IPQoS system.
	`FWD_IN`	Input traffic to be forwarded.
	`FWD_OUT`	Output traffic to be forwarded.
`precedence`	Precedence value. Highest precedence is 0.	Precedence is used to order filters with the same priority.
`ip_version`	`V4` or `V6`	Addressing scheme that is used by the packets, either IPv4 or IPv6.

Meter Module

The *meter* tracks the transmission rate of flows on a per-packet basis. The meter then determines whether the packet conforms to the configured parameters. The meter module determines the next action for a packet from a set of actions that depend on packet size, configured parameters, and flow rate.

The meter consists of two metering modules, `tokenmt` and `tswtclmt`, which you configure in the IPQoS configuration file. You can configure either module or both modules for a class.

When you configure a metering module, you can define two parameters for rate:

- `committed-rate` – Defines the acceptable transmission rate in bits per second for packets of a particular class
- `peak-rate` – Defines the maximum transmission rate in bits per second that is allowable for packets of a particular class

A metering action on a packet can result in one of three outcomes:

- `green` – The packet causes the flow to remain within its committed rate.
- `yellow` – The packet causes the flow to exceed its committed rate but not its peak rate.
- `red` – The packet causes the flow to exceed its peak rate.

You can configure each outcome with different actions in the IPQoS configuration file. Committed rate and peak rate are explained in the next section.

`tokenmt` Metering Module

The `tokenmt` module uses *token buckets* to measure the transmission rate of a flow. You can configure `tokenmt` to operate as a single-rate or two-rate meter. A `tokenmt` action instance maintains two token buckets that determine whether the traffic flow conforms to configured parameters.

The `tokenmt(7ipp)` man page explains how IPQoS implements the token meter paradigm. You can find more general information about token buckets in Kalevi Kilkki's *Differentiated Services for the Internet* and on a number of web sites.

Configuration parameters for `tokenmt` are as follows:

- `committed_rate` – Specifies the committed rate of the flow in bits per second.
- `committed_burst` – Specifies the committed burst size in bits. The `committed_burst` parameter defines how many outgoing packets of a particular class can pass onto the network at the committed rate.
- `peak_rate` – Specifies the peak rate in bits per second.
- `peak_burst` – Specifies the peak or excess burst size in bits. The `peak_burst` parameter grants to a traffic class a peak-burst size that exceeds the committed rate.

- `color_aware` – Turns on awareness mode for `tokenmt`.
- `color_map` – Defines an integer array that maps DSCP values to green, yellow, or red.

Configuring tokenmt as a Single-Rate Meter

To configure `tokenmt` as a single-rate meter, do not specify a `peak_rate` parameter for `tokenmt` in the IPQoS configuration file. To configure a single-rate `tokenmt` instance to have a red, green, or a yellow outcome, you must specify the `peak_burst` parameter. If you do not use the `peak_burst` parameter, you can configure `tokenmt` to have only a red outcome or green outcome. For an example of a single-rate `tokenmt` with two outcomes, see Example 33–3.

When `tokenmt` operates as a single-rate meter, the `peak_burst` parameter is actually the excess burst size. `committed_rate`, and either `committed_burst` or `peak_burst`, must be nonzero positive integers.

Configuring tokenmt as a Two-Rate Meter

To configure `tokenmt` as a two-rate meter, specify a `peak_rate` parameter for the `tokenmt` action in the IPQoS configuration file. A two-rate `tokenmt` always has the three outcomes, red, yellow, and green. The `committed_rate`, `committed_burst`, and `peak_burst` parameters must be nonzero positive integers.

Configuring tokenmt to Be Color Aware

To configure a two-rate `tokenmt` to be color aware, you must add parameters to specifically add "color awareness." The following is an example action statement that configures `tokenmt` to be color aware.

EXAMPLE 36–1 Color-Aware `tokenmt` Action for the IPQoS Configuration File

```
action {
    module tokenmt
    name meter1
    params {
          committed_rate 4000000
          peak_rate 8000000
          committed_burst 4000000
          peak_burst 8000000
          global_stats true
          red_action_name continue
          yellow_action_name continue
          green_action_name continue
          color_aware true
          color_map {0-20,22:GREEN;21,23-42:RED;43-63:YELLOW}
    }
}
```

You turn on color awareness by setting the `color_aware` parameter to `true`. As a color-aware meter, `tokenmt` assumes that the packet has already been marked as red, yellow, or green by a previous `tokenmt` action. Color-aware `tokenmt` evaluates a packet by using the DSCP in the packet header in addition to the parameters for a two-rate meter.

The `color_map` parameter contains an array into which the DSCP in the packet header is mapped. Consider the following `color_map` array:

```
color_map {0-20,22:GREEN;21,23-42:RED;43-63:YELLOW}
```

Packets with a DSCP of 0–20 and 22 are mapped to green. Packets with a DSCP of 21 and 23–42 are mapped to red. Packets with a DSCP of 43–63 are mapped to yellow. `tokenmt` maintains a default color map. However, you can change the default as needed by using the `color_map` parameters.

In the *color*`_action_name` parameters, you can specify `continue` to complete processing of the packet. Or, you can add an argument to send the packet to a marker action, for example, `yellow_action_name mark22`.

`tswtclmt` Metering Module

The `tswtclmt` metering module estimates average bandwidth for a traffic class by using a time-based *rate estimator*. `tswtclmt` always operates as a three-outcome meter. The rate estimator provides an estimate of the flow's arrival rate. This rate should approximate the running average bandwidth of the traffic stream over a specific period or time, its *time window*. The rate estimation algorithm is taken from RFC 2859, *A Time Sliding Window Three Colour Marker*.

You use the following parameters to configure `tswtclmt`:

- `committed_rate` – Specifies the committed rate in bits per second
- `peak_rate` – Specifies the peak rate in bits per second
- `window` – Defines the time window, in milliseconds over which history of average bandwidth is kept

For technical details on `tswtclmt`, refer to the`tswtclmt(7ipp)` man page. For general information on rate shapers that are similar to `tswtclmt`, see RFC 2963, A Rate Adaptive Shaper for Differentiated Services (`http://www.ietf.org/rfc/rfc2963.txt?number=2963`).

Marker Module

IPQoS includes two marker modules, `dscpmk` and `dlcosmk`. This section contains information for using both markers. Normally, you should use `dscpmk` because `dlcosmk` is only available for IPQoS systems with VLAN devices.

For technical information about `dscpmk`, refer to the `dscpmk`(7ipp) man page. For technical information about `dlcosmk`, refer to the `dlcosmk`(7ipp) man page.

Using the `dscpmk` Marker for Forwarding Packets

The marker receives traffic flows after the flows are processed by the classifier or by the metering modules. The marker marks the traffic with a forwarding behavior. This forwarding behavior is the action to be taken on the flows after the flows leaving the IPQoS system. Forwarding behavior to be taken on a traffic class is defined in the *per-hop behavior (PHB)*. The PHB assigns a priority to a traffic class, which indicates the precedence flows of that class in relation to other traffic classes. PHBs only govern forwarding behaviors on the IPQoS system's contiguous network. For more information on PHBs, refer to "Per-Hop Behaviors" on page 617.

Packet forwarding is the process of sending traffic of a particular class to its next destination on a network. For a host such as an IPQoS system, a packet is forwarded from the host to the local network stream. For a Diffserv router, a packet is forwarded from the local network to the router's next hop.

The marker marks the DS field in the packet header with a well-known forwarding behavior that is defined in the IPQoS configuration file. Thereafter, the IPQoS system and subsequent Diffserv-aware systems forward the traffic as indicated in the DS field until the mark changes. To assign a PHB, the IPQoS system marks a value in the DS field of the packet header. This value is called the differentiated services codepoint (DSCP). The Diffserv architecture defines two types of forwarding behaviors, EF and AF, which use different DSCPs. For overview information about DSCPs, refer to "DS Codepoint" on page 617.

The IPQoS system reads the DSCP for the traffic flow and evaluates the flow's precedence in relation to other outgoing traffic flows. The IPQoS system then prioritizes all concurrent traffic flows and releases each flow onto the network by its priority.

The Diffserv router receives the outgoing traffic flows and reads the DS field in the packet headers. The DSCP enables the router to prioritize and schedule the concurrent traffic flows. The router forwards each flow by the priority that is indicated by the PHB. Note that the PHB cannot apply beyond the boundary router of the network unless Diffserv-aware systems on subsequent hops also recognize the same PHB.

Expedited Forwarding (EF) PHB

Expedited forwarding (EF) guarantees that packets with the recommended EF codepoint 46 (101110) receive the best treatment that is available on release to the network. Expedited forwarding is often compared to a leased line. Packets with the 46 (101110) codepoint are guaranteed preferential treatment by all Diffserv routers en route to the packets' destination. For technical information about EF, refer to RFC 2598, *An Expedited Forwarding PHB*.

Assured Forwarding (AF) PHB

Assured forwarding (AF) provides four different classes of forwarding behaviors that you can specify to the marker. The next table shows the classes, the three drop precedences that are

provided with each class, and the recommended DSCPs that are associated with each precedence. Each DSCP is represented by its AF value, its value in decimal, and its value in binary.

TABLE 36–2 Assured Forwarding Codepoints

	Class 1	Class 2	Class 3	Class 4
Low-Drop Precedence	AF11 = 10 (001010)	AF21 = 18 (010010)	AF31 = 26 (011010)	AF41 = 34 (100010)
Medium-Drop Precedence	AF12 = 12 (001100)	AF22 = 20 (010100)	AF32 = 28 (011100)	AF42 = 36 (100100)
High-Drop Precedence	AF13 = 14 (001110)	AF23 = 22 (010110)	AF33 = 30 (011110)	AF43 = 38 (100110)

Any Diffserv-aware system can use the AF codepoint as a guide for providing differentiated forwarding behaviors to different classes of traffic.

When these packets reach a Diffserv router, the router evaluates the packets' codepoints along with DSCPs of other traffic in the queue. The router then forwards or drops packets, depending on the available bandwidth and the priorities that are assigned by the packets' DSCPs. Note that packets that are marked with the EF PHB are guaranteed bandwidth over packets that are marked with the various AF PHBs.

Coordinate packet marking between any IPQoS systems on your network and the Diffserv router to ensure that packets are forwarded as expected. For example, suppose IPQoS systems on your network mark packets with AF21 (010010), AF13 (001110), AF43 (100110), and EF (101110) codepoints. You then need to add the AF21, AF13, AF43, and EF DSCPs to the appropriate file on the Diffserv router.

For a technical explanation of the AF codepoint table, refer to RFC 2597. Router manufacturers Cisco Systems and Juniper Networks have detailed information about setting the AF PHB on their web sites. You can use this information to define AF PHBs for IPQoS systems as well as routers. Additionally, router manufacturers' documentation contains instructions for setting DS codepoints on their equipment.

Supplying a DSCP to the Marker

The DSCP is 6 bits in length. The DS field is 1 byte long. When you define a DSCP, the marker marks the first 6 significant bits of the packet header with the DS codepoint. The remaining 2 least-significant bits are unused.

To define a DSCP, you use the following parameter within a marker action statement:

```
dscp_map{0-63:DS_codepoint}
```

The `dscp_map` parameter is a 64-element array, which you populate with the (DSCP) value. `dscp_map` is used to map incoming DSCPs to outgoing DSCPs that are applied by the `dscpmk` marker.

You must specify the DSCP value to `dscp_map` in decimal notation. For example, you must translate the EF codepoint of 101110 into the decimal value 46, which results in `dscp_map{0-63:46}`. For AF codepoints, you must translate the various codepoints that are shown in Table 36–2 to decimal notation for use with `dscp_map`.

Using the `dlcosmk` Marker With VLAN Devices

The `dlcosmk` marker module marks a forwarding behavior in the MAC header of a datagram. You can use `dlcosmk` only on an IPQoS system with a VLAN interface.

`dlcosmk` adds four bytes, which are known as the *VLAN tag*, to the MAC header. The VLAN tag includes a 3-bit user-priority value, which is defined by the IEEE 801.D standard. Diffserv-aware switches that understand VLAN can read the user-priority field in a datagram. The 801.D user priority values implement the class-of-service (CoS) marks, which are well known and understood by commercial switches.

You can use the user-priority values in the `dlcosmk` marker action by defining the class of service marks that are listed in the next table.

TABLE 36–3 801.D User-Priority Values

Class of Service	Definition
0	Best effort
1	Background
2	Spare
3	Excellent effort
4	Controlled load
5	Video less than 100ms latency
6	Video less than 10ms latency
7	Network control

For more information on `dlcosmk`, refer to the `dlcosmk(7ipp)` man page.

IPQoS Configuration for Systems With VLAN Devices

This section introduces a simple network scenario that shows how to implement IPQoS on systems with VLAN devices. The scenario includes two IPQoS systems, `machine1` and

machine2, that are connected by a switch. The VLAN device on machine1 has the IP address 10.10.8.1. The VLAN device on machine2 has the IP address 10.10.8.3.

The following IPQoS configuration file for machine1 shows a simple solution for marking traffic through the switch to machine2.

EXAMPLE 36–2 IPQoS Configuration File for a System With a VLAN Device

```
fmt_version 1.0
action {
        module ipgpc
          name ipgpc.classify

        filter {
                name myfilter2
                daddr 10.10.8.3
                class myclass
        }

        class {
                name myclass
                next_action mark4
        }
}

action {
        name mark4
        module dlcosmk
        params {
                cos 4
                next_action continue
        global_stats true
        }
}
```

In this configuration, all traffic from machine1 that is destined for the VLAN device on machine2 is passed to the dlcosmk marker. The mark4 marker action instructs dlcosmk to add a VLAN mark to datagrams of class myclass with a CoS of 4. The user-priority value of 4 indicates that the switch between the two machines should give controlled load forwarding to myclass traffic flows from machine1.

flowacct Module

The IPQoS flowacct module records information about traffic flows, a process that is referred to as *flow accounting*. Flow accounting produces data that can be used for billing customers or for evaluating the amount of traffic to a particular class.

Flow accounting is optional. flowacct is typically the final module that metered or marked traffic flows might encounter before release onto the network stream. For an illustration of flowacct's position in the Diffserv model, see Figure 31–1. For detailed technical information about flowacct, refer to the flowacct(7ipp) man page.

To enable flow accounting, you need to use the Oracle Solaris `exacct` accounting facility and the `acctadm` command, as well as `flowacct`. For the overall steps in setting up flow accounting, refer to "Setting Up Flow Accounting (Task Map)" on page 675.

`flowacct` Parameters

The `flowacct` module gathers information about flows in a *flow table* that is composed of *flow records*. Each entry in the table contains one flow record. You cannot display a flow table.

In the IPQoS configuration file, you define the following `flowacct` parameters to measure flow records and to write the records to the flow table:

- `timer` – Defines an interval, in milliseconds, when timed-out flows are removed from the flow table and written to the file that is created by `acctadm`
- `timeout` – Defines an interval, in milliseconds, which specifies how long a packet flow must be inactive before the flow times out

  Note – You can configure `timer` and `timeout` to have different values.

- `max_limit` – Places an upper limit on the number of flow records that can be stored in the flow table

For an example of how `flowacct` parameters are used in the IPQoS configuration file, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 662.

Flow Table

The `flowacct` module maintains a flow table that records all packet flows that are seen by a `flowacct` instance. A flow is identified by the following parameters, which include the `flowacct` 8–tuple:

- Source address
- Destination address
- Source port
- Destination port
- DSCP
- User ID
- Project ID
- Protocol Number

If all the parameters of the 8–tuple for a flow remain the same, the flow table contains only one entry. The `max_limit` parameter determines the number of entries that a flow table can contain.

The flow table is scanned at the interval that is specified in the IPQoS configuration file for the `timer` parameter. The default is 15 seconds. A flow "times out" when its packets are not seen by

the IPQoS system for at least the `timeout` interval in the IPQoS configuration file. The default time out interval is 60 seconds. Entries that have timed out are then written to the accounting file that is created with the `acctadm` command.

`flowacct` Records

A `flowacct` record contains the attributes described in the following table.

TABLE 36–4 Attributes of a `flowacct` Record

Attribute Name	Attribute Contents	Type
`src-addr-`*address-type*	Source address of the originator. *address-type* is either `v4` for IPv4 or `v6` for IPv6, as specified in the IPQoS configuration file.	Basic
`dest-addr-`*address-type*	Destination address for the packets. *address-type* is either `v4` for IPv4 or `v6` for IPv6, as specified in the IPQoS configuration file.	Basic
`src-port`	Source port from which the flow originated.	Basic
`dest-port`	Destination port number to which this flow is bound.	Basic
`protocol`	Protocol number for the flow.	Basic
`total-packets`	Number of packets in the flow.	Basic
`total-bytes`	Number of bytes in the flow.	Basic
action-name	Name of the `flowacct` action that recorded this flow.	Basic
`creation-time`	First time that a packet is seen for the flow by `flowacct`.	Extended only
`last-seen`	Last time that a packet of the flow was seen.	Extended only
`diffserv-field`	DSCP in the outgoing packet headers of the flow.	Extended only
`user`	Either a UNIX User ID or user name, which is obtained from the application.	Extended only
`projid`	Project ID, which is obtained from the application.	Extended only

Using `acctadm` with the `flowacct` Module

You use the `acctadm` command to create a file in which to store the various flow records that are generated by `flowacct`. `acctadm` works in conjunction with the extended accounting facility. For technical information about `acctadm`, refer to the `acctadm(1M)` man page.

The `flowacct` module observes flows and fills the flow table with flow records. `flowacct` then evaluates its parameters and attributes in the interval that is specified by `timer`. When a packet

is not seen for at least the `last_seen` plus `timeout` values, the packet times out. All timed-out entries are deleted from the flow table. These entries are then written to the accounting file each time the interval that is specified in the `timer` parameter elapses.

To invoke `acctadm` for use with the `flowacct` module, use the following syntax:

`acctadm -e` *file-type* `-f` *filename* `flow`

`acctadm -e`	Invokes `acctadm` with the `-e` option. The -e indicates that a resource list follows.
file-type	Specifies the attributes to be gathered. *file-type* must be replaced by either `basic` or `extended`. For a list of attributes in each file type, refer to Table 36–4.
`-f`*file-name*	Creates the file*file-name* to hold the flow records.
`flow`	Indicates that `acctadm` is to be run with IPQoS.

IPQoS Configuration File

This section contains full details about the parts of the IPQoS configuration file. The IPQoS boot-time activated policy is stored in the file `/etc/inet/ipqosinit.conf`. Although you can edit this file, the best practice for a new IPQoS system is to create a configuration file with a different name. Tasks for applying and debugging an IPQoS configuration are in Chapter 33, "Creating the IPQoS Configuration File (Tasks)."

The syntax of the IPQoS configuration file is shown in Example 36–3. The example uses the following conventions:

- `computer-style type` – Syntactical information that is provided to explain the parts of the configuration file. You do not type any text that appears in computer-style type.
- **`bold type`** – Literal text that you must type in the IPQoS configuration file. For example, you must always begin the IPQoS configuration file with **`fmt_version`**.
- *italic type* – Variable text that you replace with descriptive information about your configuration. For example, you must always replace *action-name* or *module-name* with information that pertains to your configuration.

EXAMPLE 36–3 Syntax of the IPQoS Configuration File

```
file_format_version ::= fmt_version version

action_clause ::= action {
     name action-name
     module module-name
     params-clause | ""
     cf-clauses
}
action_name ::= string
```

EXAMPLE 36–3 Syntax of the IPQoS Configuration File *(Continued)*

```
module_name ::= ipgpc | dlcosmk | dscpmk | tswtclmt | tokenmt | flowacct

params_clause ::= params {
     parameters
     params-stats |   ""
     }
parameters ::=    prm-name-value parameters |  ""
prm_name_value ::= param-name param-value

params_stats ::= global-stats boolean

cf_clauses ::= class-clause cf-clauses |
               filter-clause cf-clauses | ""

class_clause ::= class {
     name class-name
     next_action next-action-name
     class-stats | ""
                  }
class_name  ::= string
next_action_name  ::= string
class_stats ::= enable_stats boolean
boolean ::= TRUE | FALSE

filter_clause ::= filter {
                  name filter-name
                  class class–name
                  parameters
                  }
filter_name ::= string
```

The remaining text describes each major part of the IPQoS configuration file.

`action` Statement

You use `action` statements to invoke the various IPQoS modules that are described in "IPQoS Architecture and the Diffserv Model" on page 681.

When you create the IPQoS configuration file, you must always begin with the version number. Then, you must add the following `action` statement to invoke the classifier:

```
fmt_version 1.0

action {
    module ipgpc
    name ipgpc.classify
}
```

Follow the classifier `action` statement with a `params` clause or a `class` clause.

Use the following syntax for all other `action` statements:

```
action {
name action-name
module module-name
params-clause | ""
cf-clauses
}
```

`name` *action_name*	Assigns a name to the action.
`module` *module_name*	Identifies the IPQoS module to be invoked, which must be one of the modules in Table 36–5.
params_clause	Can be parameters for the classifier to process, such as global statistics or the next action to process.
cf_clauses	A set of zero or more `class` clauses or `filter` clauses

Module Definitions

The module definition indicates which module is to process the parameters in the `action` statement. The IPQoS configuration file can include the following modules.

TABLE 36–5 IPQoS Modules

Module Name	Definition
`ipgpc`	IP classifier
`dscpmk`	Marker to be used to create DSCPs in IP packets
`dlcosmk`	Marker to be used with VLAN devices
`tokenmt`	Token bucket meter
`tswtclmt`	Time-sliding window meter
`flowacct`	Flow-accounting module

`class` Clause

You define a `class` clause for each class of traffic.

Use this syntax to define the remaining classes in the IPQoS configuration:

```
class {

      name class-name
      next_action next-action-name
}
```

To enable statistics collection on a particular class, you must first enable global statistics in the `ipgpc.classify` `action` statement. For more information, refer to "`action` Statement" on page 694.

Use the `enable_stats` `TRUE` statement whenever you want to turn on statistics collection for a class. If you do not need to gather statistics for a class, you can specify `enable_stats` `FALSE`. Alternatively, you can eliminate the `enable_stats` statement.

Traffic on an IPQoS-enabled network that you do not specifically define is relegated to the *default class*.

`filter` Clause

Filters are made up of selectors that group traffic flows into classes. These selectors specifically define the criteria to be applied to traffic of the class that was created in the class clause. If a packet matches all selectors of the highest-priority filter, the packet is considered to be a member of the filter's class. For a complete list of selectors that you can use with the `ipgpc` classifier, refer to Table 36–1.

You define filters in the IPQoS configuration file by using a *filter clause*, which has the following syntax:

```
filter {
       name filter-name
       class class-name
       parameters (selectors)
       }
```

`params` Clause

The `params` clause contains processing instructions for the module that is defined in the action statement. Use the following syntax for the `params` clause:

```
params {
           parameters
           params-stats | ""
       }
```

In the `params` clause, you use parameters that are applicable to the module.

The *params-stats* value in the params clause is either global_stats TRUE or global_stats FALSE. The global_stats TRUE instruction turns on UNIX style statistics for the action statement where global statistics is invoked. You can view the statistics by using the kstat command. You must enable action statement statistics before you can enable per-class statistics.

ipqosconf Configuration Utility

You use the ipqosconf utility to read the IPQoS configuration file and to configure IPQoS modules in the UNIX kernel. ipqosconf performs the following actions:

- Applies the configuration file to the IPQoS kernel modules (ipqosconf -a *filename*)
- Lists the IPQoS configuration file currently resident in the kernel (ipqosconf -l)
- Ensures that the current IPQoS configuration is read and applied each time the machine reboots (ipqosconf -c)
- Flushes the current IPQoS kernel modules (ipqosconf -f)

For technical information, refer to the ipqosconf(1M) man page.

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 Oracle Solaris implementation of IPv6 does not support the creation of anycast addresses and groups. However, Oracle 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.

datagram See IP datagram.

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 an Oracle Solaris 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.

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 Oracle Solaris, 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.

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 `igb` 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, `igb`.

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.

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).

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 (datalink) 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.

visited network A network other than a mobile node's home network, to which the mobile node is currently connected.

visitor list The list of mobile nodes that are visiting a foreign agent.

Index

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

T

U