Oracle® Solaris Administration: IP Services

Part No: 821–1453–11
March 2012

120602@25097

Contents

Preface

Welcome to Oracle Solaris Administration: 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.

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

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
Booting and Shutting Down Oracle Solaris on SPARC Platforms	Booting and shutting down a system, managing boot services, modifying boot behavior, booting from ZFS, managing the boot archive, and troubleshooting booting on SPARC platforms
Booting and Shutting Down Oracle Solaris on x86 Platforms	Booting and shutting down a system, managing boot services, modifying boot behavior, booting from ZFS, managing the boot archive, and troubleshooting booting on x86 platforms
Oracle Solaris Administration: Common Tasks	Using Oracle Solaris commands, booting and shutting down a system, managing user accounts and groups, managing services, hardware faults, system information, system resources, and system performance, managing software, printing, the console and terminals, and troubleshooting system and software problems
Oracle Solaris Administration: Devices and File Systems	Removable media, disks and devices, file systems, and backing up and restoring data

Book Title	Topics
Oracle Solaris Administration: IP Services	TCP/IP network administration, IPv4 and IPv6 address administration, DHCP, IPsec, IKE, IP Filter, and IPQoS
Oracle Solaris Administration: Naming and Directory Services	DNS, NIS, and LDAP naming and directory services, including transitioning from NIS to LDAP
Oracle Solaris Administration: Network Interfaces and Network Virtualization	Automatic and manual IP interface configuration including WiFi wireless; administration of bridges, VLANs, aggregations, LLDP, and IPMP; virtual NICs and resource management.
Oracle Solaris Administration: Network Services	Web cache servers, time-related services, network file systems (NFS and autofs), mail, SLP, and PPP
Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, 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 Zones, which host Oracle Solaris 10 environments running on the Oracle Solaris 11 kernel
Oracle Solaris Administration: Security Services	Auditing, device management, file security, BART, Kerberos services, PAM, Cryptographic Framework, Key Management, privileges, RBAC, SASL, Secure Shell, and virus scanning
Oracle Solaris Administration: SMB and Windows Interoperability	SMB service, which enables you to configure an Oracle Solaris system to make SMB shares available to SMB clients; 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 Administration: ZFS File Systems	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
Trusted Extensions Configuration and Administration	System installation, configuration, and administration that is specific to Trusted Extensions
Oracle Solaris 11 Security Guidelines	Securing an Oracle Solaris system, as well as usage scenarios for its security features, such as zones, ZFS, and Trusted Extensions
Transitioning From Oracle Solaris 10 to Oracle Solaris 11	Provides system administration information and examples for transitioning from Oracle Solaris 10 to Oracle Solaris 11 in the areas of installation, device, disk, and file system management, software management, networking, system management, security, virtualization, desktop features, user account management, and user environments

Access to Oracle Support

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

Typographic Conventions

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

TABLE P–1 Typographic Conventions

Typeface	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	`$`

TABLE P–2 Shell Prompts *(Continued)*

Shell	Prompt
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 the Network Deployment

This chapter briefly describes the different considerations when you plan your network setup. These issues will help you to deploy your network in an organized and cost-effective manner. Note that the details of planning the network are outside the scope of this book. Only general directions are provided.

This book assumes that you are familiar with basic networking concepts and terminology. Refer to the following resources for introductions to these basic concepts:

- 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*
- For a brief description of how the TCP/IP protocol suite is implemented in this Oracle Solaris release, see Chapter 1, "Overview of the Networking Stack," in *Oracle Solaris Administration: Network Interfaces and Network Virtualization*.

More references to introductions and overviews are provided in the appropriate sections that follow.

Network Planning (Task Map)

The following table lists different tasks for planning the network configuration.

Task	Description	For Information
Identify the hardware requirements of your planned network topology.	Determine the types of equipment that you need for your network site.	"Determining the Network Hardware" on page 26 For information about a specific type of equipment, refer to the equipment manufacturer's documentation.

Task	Description	For Information
Determine the type of IP addresses to use and obtain registered IP addresses.	Select whether you are deploying a purely IPv4 network, an IPv6 network, or a network that uses both types of IP addresses. Obtain unique IP addresses to communicate to public networks in the Internet.	"Deciding on an IP Addressing Format for Your Network" on page 27 "Obtaining Your Network's IP Number" on page 29.
Determine a naming scheme to identify the hosts in the network as well as the name service to use.	Create a list of names to assign to the systems on the network and decide whether to use NIS, LDAP, DNS, or the network databases in the local `/etc` directory.	"Administering Host Names" on page 30 "Selecting a Name Service and Directory Service" on page 30
If necessary, establish administrative subdivisions and design a strategy for subnets.	Decide if your site requires that you divide your network into subnets to service administrative subdivisions	"Using Subnets" on page 31
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.	"Planning for Routers on Your Network" in *System Administration Guide: IP Services*
Decide whether to create virtual networks in the overall network configuration scheme.	You might need to create virtual networks within a system to reduce the hardware footprint of your network.	Part III, "Network Virtualization and Resource Management," in *Oracle Solaris Administration: Network Interfaces and Network Virtualization*

Determining the Network Hardware

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*.

Some of the planning decisions you must make about hardware follow:

- The network topology, the layout, and connections of the network hardware
- The type and number of host systems your network can support, including the servers that might be required
- Network devices to be installed in these systems
- The type of network media to use, such as Ethernet, and so on

- Whether you need bridges or routers extend this media or connect the local network to external networks

---

Note – For a description of how routers function, see "Planning for Routers on Your Network" in *System Administration Guide: IP Services*. For an overview of bridges, see "Bridging Overview" in *Oracle Solaris Administration: Network Interfaces and Network Virtualization*

Deciding on an IP Addressing Format for Your Network

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 multiple network interface cards (NICs) with their own individual IP addresses
- Whether to use private addresses on your network
- Whether to have a DHCP server that manages pools of IPv4 addresses

Briefly, the type of IP addresses include the following:

IPv4 Addresses

These 32-bit addresses are the original IP addressing format for TCP/IP.

For an overview of class-based IPv4 addressing, refer to the following resources:

- "Designing Your IPv4 Addressing Scheme" in *System Administration Guide: IP Services*
- Internet Protocol DARPA Internet Program Protocol Specification (`http://tools.ietf.org/html/rfc791`)

The IETF developed *Classless Inter-Domain Routing (CIDR)* addresses as a short to medium term remedy for the shortage of IPv4 addresses and the limited capacity of the global Internet routing tables.

For more information, refer to the following resources:

- "Designing Your CIDR IPv4 Addressing Scheme" in *System Administration Guide: IP Services*
- Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan (`http://tools.ietf.org/html/rfc4632`)

The following table provides the subnets in both CIDR notation and dotted decimal format.

TABLE 1–1 CIDR Prefixes and Their Decimal Equivalents

CIDR Network Prefix	Dotted Decimal Subnet Equivalent	Available IP Addresses
/19	255.255.224.0	8,192
/20	255.255.240.0	4,096
/21	255.255.248.0	2,048
/22	255.255.252.0	1,024
/23	255.255.254.0	512
/24	255.255.255.0	256
/25	255.255.255.128	128
/26	255.255.255.192	64
/27	255.255.255.224	32

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 10, “About DHCP (Overview).”

IPv6 Addresses

The 128–bit IPv6 addresses provide greater address space than is available with IPv4. As with IPv4 addresses in CIDR format, IPv6 addresses are classless and use prefixes to designate the portion of the address that defines the site's network.

For more information about IPv6 addresses, refer to the following resources:

- “IPv6 Addressing Overview” in *System Administration Guide: IP Services*
- Internet Protocol, Version 6 (IPv6) Specification (`http://tools.ietf.org/html/rc2460`)

Private Addresses and Documentation Prefixes

The IANA has reserved a block of IPv4 addresses and an IPv6 site prefix for use on private networks. These private addresses are used for network traffic within a private network. These addresses are also used in documentation.

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

For IPv6 addresses, the prefix `2001:db8::/32` is a special IPv6 prefix that is used specifically for documentation examples. 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*.

To enable the private network to communicate to external networks in the Internet, you must obtain a registered IP number for your network from the appropriate organization. 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`).

Naming Entities on Your Network

The TCP/IP protocols locate a system on a network by using its IP address. However, a host name enables you to identify systems more easily than IP addresses. 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

Plan a naming scheme for the systems that will comprise the network. For systems that function as servers and have multiple NICs, at least one host name that is associated with the IP address of its primary network interface must be provided.

No two machines on the network can both have the name host name. Thus each host name must be unique to each system. However, a host or a system with its assigned unique name can 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

In Oracle Solaris you can select from 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. You can also use 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 *Oracle Solaris Administration: Naming and Directory Services*.

During the OS installation, you supply the host name and IP address of your server, clients, or standalone system. The installation program adds this information into the `hosts` database to be used by the network service when servicing the network.

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.

For name service, you can select one of the following:

- NIS or DNS — The NIS and DNS name services maintain network databases on several servers on the network. *Oracle Solaris Administration: Naming and Directory Services* describes these name services and explains how to configure the databases. In addition, the guide explain the "namespace" and "administrative domain" concepts in detail.
- Local files— 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 such as `.com`, `.org`, `.edu`, `.gov`, and so forth.

Using Subnets

The use of subnets is connected with the need for administrative subdivisions to address issues of size and control. The more hosts and servers that you have in a network, the more complex your management task. By creating administrative divisions and using subnets, management of a complex network becomes easier. The decision about setting up administrative subdivisions for your network is determined by the following factors:

- **Size of the network**

  Subnets are also useful even in a relatively small network whose subdivisions are located across an extensive geographical area.

- **Common needs shared by groups of users**

  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.

For a general description, see "What Is Subnetting?" in *System Administration Guide: IP Services*.

Deploying Virtual Networks

This Oracle Solaris release supports the creation of virtual networks in a single network by configuring zones as well as virtual network cards (VNICs). VNICs are network interfaces that are created on top of physical NICs. The combination of zones and VNICs is an effective way to consolidate a huge datacenter that contains a large number of physical systems into fewer systems. For more information about virtual networking, see Part III, "Network Virtualization and Resource Management," in *Oracle Solaris Administration: Network Interfaces and Network Virtualization*.

CHAPTER 2

Considerations When Using IPv6 Addresses

This chapter supplements Chapter 1, "Planning the Network Deployment," by describing additional considerations if you decide to use IPv6 addresses on your network.

If you do plan to also use IPv6 addresses in addition to IPv4 addresses, ensure that your current ISP supports both address types. Otherwise, you would need to find a separate ISP to support the IPv6 addresses.

For an introduction to IPv6 concepts, refer to the following resources:.

- "IPv6 Addressing Overview" in *System Administration Guide: IP Services*
- Internet Protocol, Version 6 (IPv6) Specification (`http://tools.ietf.org/html/rc2460`)

IPv6 Planning (Task Map)

The following table lists different considerations when planning to implement IPv6 on your network.

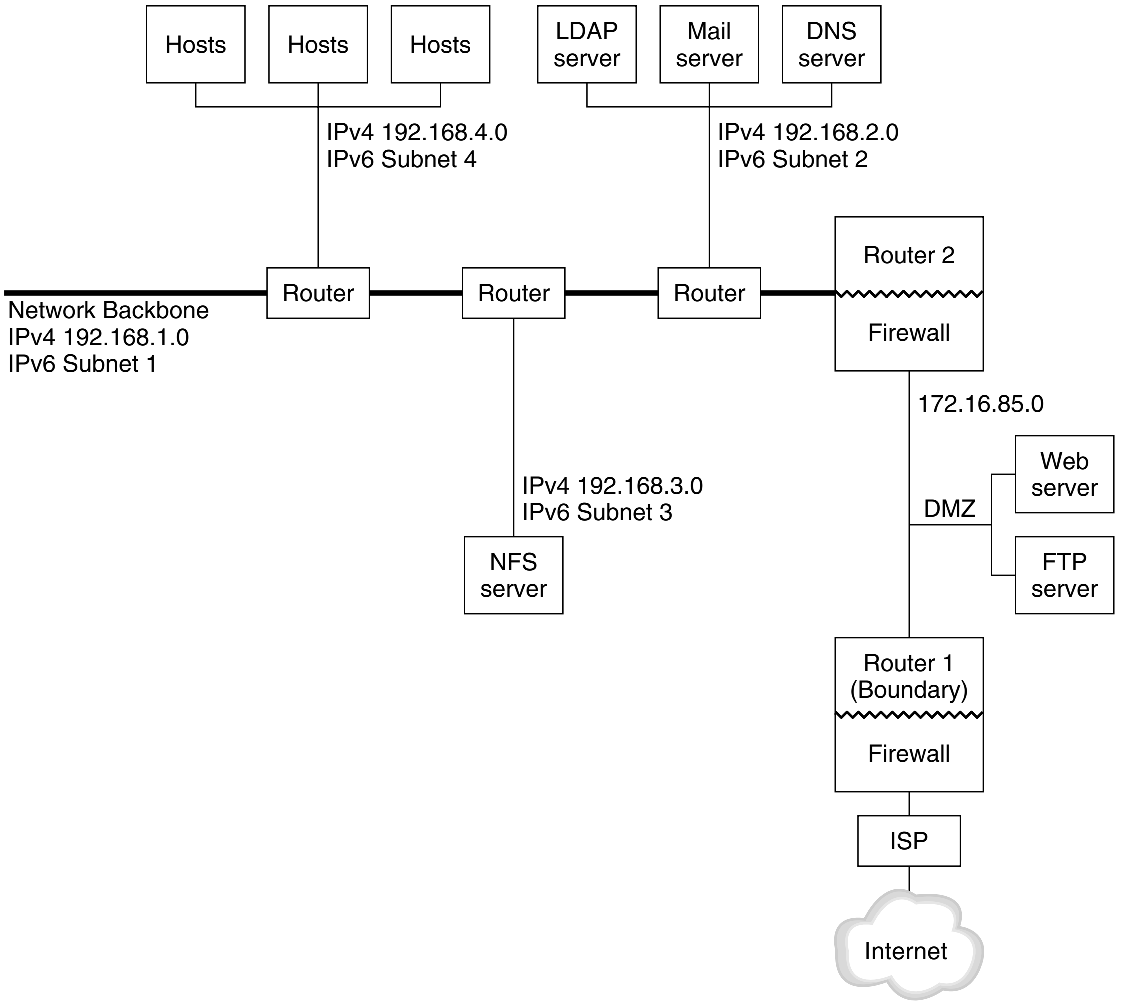
Task	Description	For Instructions
Prepare your hardware to support IPv6.	Ensure that your hardware can be upgraded to IPv6.	"Ensuring Hardware Support for IPv6" on page 36
Ensure that your applications are IPv6 ready.	Verify that your applications can run in an IPv6 environment.	"Configuring Network Services to Support IPv6" on page 39
Design a plan for tunnel usage.	Determine which routers should run tunnels to other subnets or external networks.	"Planning for Tunnel Use in the Network" on page 41

Task	Description	For Instructions
Plan how to secure your networks and develop an IPv6 security policy.	For security purposes, you need an addressing plan for the DMZ and its entities before you configure IPv6. Decide how you would implement security, such as using IP Filter, IP security architecture (IPsec), Internet Key Exchange (IKE), and other security features of this release.	"Security Considerations for the IPv6 Implementation" on page 41 Part III, "IP Security"
Create an addressing plan for systems on the network.	Your plan for addressing servers, routers, and hosts should be in place before IPv6 configuration. This step includes obtaining a site prefix for your network as well as planning IPv6 subnets, if needed.	"Creating an IPv6 Addressing Plan for Nodes" on page 37

IPv6 Network Topology Scenario

Typically, IPv6 is used in a mixed network topology that also uses IPv4, such as shown in the following figure. This figure is used as reference in the description of IPv6 configuration tasks in the subsequent sections.

FIGURE 2–1 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 2–1, 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.

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. When you deploy IPv6 on an operating network that is already using IPv4, ensure that you do not disrupt ongoing operations.

The following sections describe areas that you need to consider when preparing to implement IPv6.

Ensuring Hardware Support for IPv6

Check the manufacturers' documentation for IPv6 readiness regarding the following classes of hardware:

- Routers
- Firewalls
- Servers
- Switches

Note – All procedures in the this book 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 136.

For each NIC of IPv6 servers, manually configure the interface ID portion of the IPv6 address instead of automatically obtaining the ID with the Neighbor Discovery protocol. In this manner, if a NIC is replaced, the same interface ID can be applied to the replacement NIC. A different ID automatically generated by the Neighbor Discovery protocol might cause unexpected behavior by the server.

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 37
- "Creating the IPv6 Numbering Scheme" on page 37

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" in *System Administration Guide: IP Services*.

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 137.

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 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 2–1 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 2–1, 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.

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 2–1. 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`

For a more detailed description of subnets, see "What Is Subnetting?" in *System Administration Guide: IP Services*

Configuring Network Services to Support IPv6

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 137.

▼ 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 41.

- 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 40.
- IPQoS

  Use the same Diffserv policies on a host that were used for IPv4. For more information, see "Classifier Module" on page 469.

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 *Oracle Solaris Administration: Naming and Directory Services*.

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 Tunnel Use in the Network

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 2–1 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 122. For conceptual information regarding tunnels, refer to “Overview of IP Tunnels” on page 113.

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 14, “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 17, "Internet Key Exchange (Overview)."

CHAPTER 3

Configuring an IPv4 Network

Network configuration evolves in two stages: assembling the hardware, and then configuring the daemons, files, and services that implement the TCP/IP protocol.

This chapter explains how to configure a network that implements IPv4 addressing and services.

Many of the tasks in this chapter apply to both IPv4-only and IPv6-enabled networks. Tasks that are specific to IPv6 networks are in Chapter 4, "Enabling IPv6 on the Network."

Note – Before you configure TCP/IP, review the different planning tasks that are listed in Chapter 1, "Planning the Network Deployment." If you are planning to use IPv6 addresses, then refer also to Chapter 2, "Considerations When Using IPv6 Addresses."

This chapter contains the following information:

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 the system's IP interfaces.	Assigns IP addresses to the IP interfaces of the system.	"How to Configure an IP Interface" on page 47
Configure a system for local files mode	Edits specific configuration files in the system's `/etc` directory as well as configures the `nis/domain` SMF service.	"How to Configure a System for Local Files Mode" on page 53
Set up a network configuration server	Enables the `in.tftp` daemon, and edits other configuration files in the system's `/etc` directory.	"How to Set Up a Network Configuration Server" on page 55
Configure a system for network client mode	Edits configuration files in the system's `/etc` directory.	"How to Configure a System for Network Client Mode" on page 54
Specify a routing strategy for the network client	Configures systems to use either static routing or dynamic routing.	"How to Enable Static Routing on a Single-Interface Host" on page 64 and "How to Enable Dynamic Routing on a Single-Interface System" on page 66.

Before You Begin Network Configuration

In this Oracle Solaris release, a system's network configuration is managed by an active *network configuration profile (NCP)*. If the active NCP in the system is `automatic`, then network configuration is automatically managed by the OS. If the active NCP is `DefaultFixed`, then network configuration is performed manually by using the `dladm` and `ipadm` commands.

Note – The `dladm` and `ipadm` commands do not work if the active NCP is `Automatic`.

For procedures to determine the system's active profile and to switch to a fixed NCP, see "Profiles and Configuration Tools" in *Oracle Solaris Administration: Network Interfaces and Network Virtualization*.

For more information about NCPs, refer to Part I, "Network Auto-Magic," in *Oracle Solaris Administration: Network Interfaces and Network Virtualization*.

In this documentation, the procedures assume that the active NCP on all of the systems on the network is `DefaultFixed`.

Configuring Component Systems on the Network

When you configure network systems, you need the following configuration information:

- Host name of each system.
- IP address and netmask of each system. If the network is subdivided into subnets, then you must have the subnet numbers and the IP address schema to apply to the systems in each subnet, including their respective netmasks.
- Domain name to which each system belongs.
- Default router address.

  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 about routers as well as the list of routing protocols that are supported by Oracle Solaris, see "Packet Forwarding and Routing on IPv4 Networks" in *System Administration Guide: IP Services*.

Note – You can configure the network while you are installing Oracle Solaris. For instructions, see *Installing Oracle Solaris 11 Systems*.

In this documentation, the procedures assume that you are configuring the network after you have installed the OS.

Use Figure 3–1 in the following section as reference to configure the component systems of the network.

IPv4 Autonomous System Topology

Sites with multiple routers and networks typically administer their network topology as a single routing domain, or *autonomous system (AS)*.

FIGURE 3–1 Autonomous System With Multiple IPv4 Routers

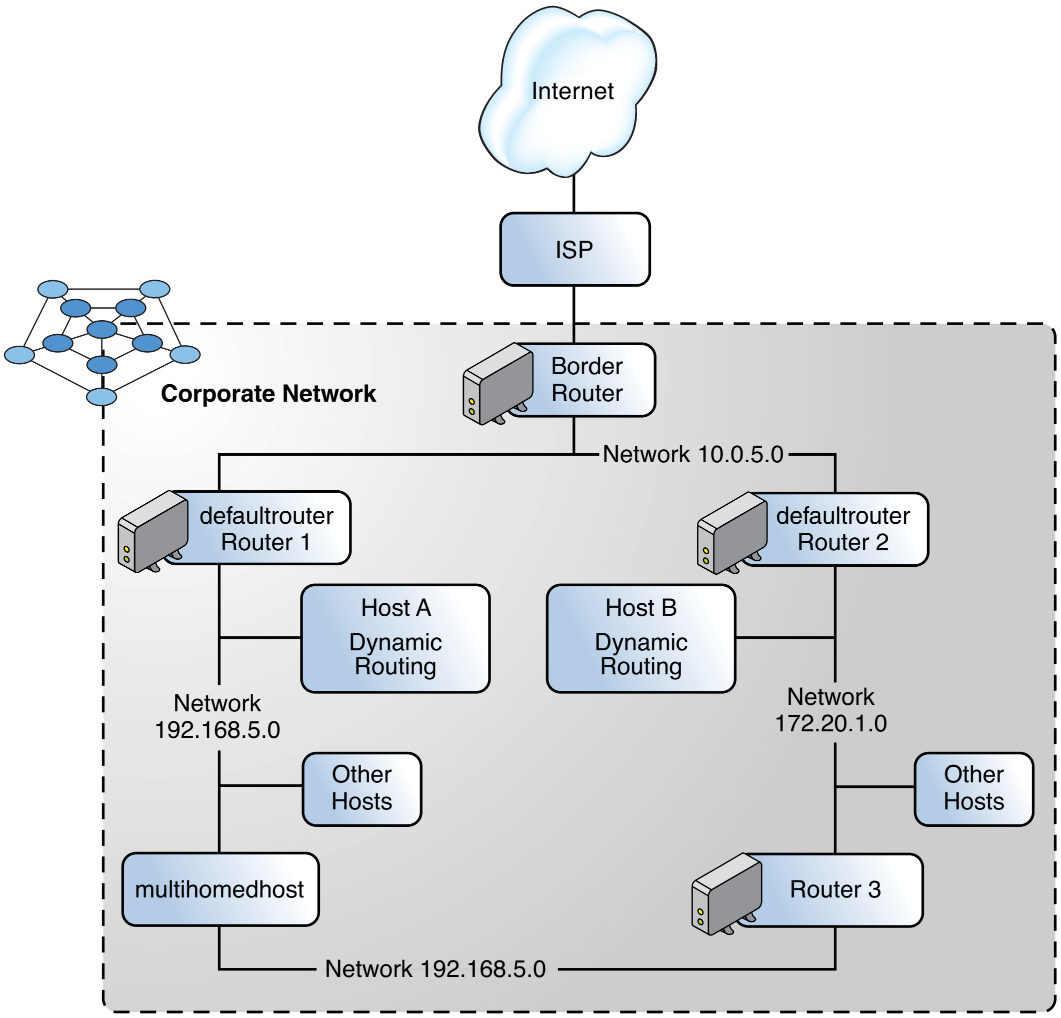


Figure 3–1shows an AS that is divided into three local networks, `10.0.5.0`, `172.20.1.0`, and `192.168.5.0`. The network is comprised of the following types of systems:

- Routers use routing protocols to manage how network packets are directed or routed from their source to their destinations within the local network or to external networks. For information about routing protocols that are supported in Oracle Solaris, see "Tables of Routing Protocols in Oracle Solaris" on page 144.

  Routers are typed as follows:

  - The *border router* connects the local network such as `10.0.5.0` externally to a service provider.

  - *Default routers* manage packet routing in the local network, which itself can include several local networks. For example, in Figure 3–1, Router 1 serves as the default router for `192.168.5`. Contemporaneously, Router 1 is also connected to the `10.0.5.0` internal network. Router 2's interfaces connect to the `10.0.5.0` and `172.20.1.0` internal networks.
  - *Packet-forwarding routers* forward packets between internal networks but do not run routing protocols. In Figure 3–1, Router 3 is a packet-forwarding router with connections to the `172.20.1` and `192.168.5` networks.
- Client systems
  - Multihomed systems or systems that have multiple NICs. In Oracle Solaris, these systems by default can forward packets to other systems in the same network segment.
  - Single-interfaced systems rely on the local routers for both packet forwarding and receiving configuration information.

▼ How to Configure an IP Interface

The following procedure provides an example of performing a basic configuration of an IP interface.

Before You Begin Determine if you want to rename datalinks on the system. Typically, you use the generic names that have been assigned by default to the datalinks. To change link names, see "How to Rename a Datalink" in *Oracle Solaris Administration: Network Interfaces and Network Virtualization*.

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*.

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

```
# dladm show-phys
```

This command shows the physical network cards that are installed on your system and some of their properties. For more information about this command, see How to Display Information About Physical Attributes of Datalinks.

3 **Display information about datalinks currently on the system.**

```
# dladm show-link
```

This command shows the datalinks and certain properties that have been set for them, including the physical cards over which the links have been created.

4 **Create the IP interface.**

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

interface-class	Refers to one of three classes of interfaces that you can create: ■ IP interface. This interface class is the most common that you create when you perform network configuration. To create this interface class, use the `create-ip` subcommand. ■ STREAMS virtual network interface driver (VNI interface). To create this interface class, use the `create-vni` subcommand. For more information about VNI devices or interfaces, see the `vni(7d)` man page. ■ IPMP interface. This interface is used when you configure IPMP groups. To create this interface class, use the `create-ipmp` subcommand. For more information about IPMP groups, see Chapter 14, "Introducing IPMP," in *Oracle Solaris Administration: Network Interfaces and Network Virtualization*.
interface	Refers to the name of the interface. The name is identical to the name of the link over which the interface is being created.

Note – You must create the IP interface before you can assign the IP address to it.

---

5 **Configure the IP interface with a valid IP address.**

The following syntax assigns a static address to an interface. Refer to the `ipadm(1M)` man page for other options for assigning IP addresses.

```
# ipadm create-addr -T address-type -a address/prefixlen addrobj
```

`-T` *address-type*	Specifies the type of IP address that is assigned to the interface, which is one of the following: `static`, `dhcp`, or `addrconf`. `Addrconf` refers to automatically generated IPv6 addresses.
`-a`	Specifies the IP address to configure on the interface. You can specify either just a local address, or both a local address and a remote address in the case of tunnel configuration. Typically, you assign only a local address. In this case, you specify the address directly with the `-a` option, such as: `-a` *address*. The address is automatically considered a local address. If you are configuring tunnels, you might be required to provide both the local address of the system and the remote address of the destination system. In this case, you must specify `local` and `remote` to distinguish the two addresses, as follows: `-a local=`*local-addr*`,remote=`*remote-addr*. For more information about configuring tunnels, see Chapter 6, "Configuring IP Tunnels," in *Oracle Solaris Administration: IP Services*. If you are using a numeric IP address, use the format *address/prefixlen* for addresses in CIDR notation, for example, `1.2.3.4/24`. See the explanation for the *prefixlen* option.

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

Note – During the boot process, the creation of IP addresses precedes naming services being brought online. Therefore you must ensure that any host name that is used in the network configuration must be defined in the `/etc/hosts` file.

/prefixlen — Specifies the length of the network ID that is part of the IPv4 address when you use CIDR notation. In the address `12.34.56.78/24`, `24` is the *prefixlen*. If you do not include *prefixlen*, then the netmask is computed according to the sequence listed for `netmask` in the `name-service/switch` service or by using classful address semantics.

addrobj — Specifies an identifier for the unique IP address or set of addresses that is used in the system. The addresses can be either IPv4 or IPv6 types. The identifier uses the format *interface*/*user_specified_string*.

The *interface* refers to the IP interface to which the address is assigned. The *interface* variable must reflect the name of the datalink on which the IP interface is configured.

user-specified-string refers to a string of alphanumeric characters that begins with an alphabet letter and has a maximum length of 32 characters. Subsequently, you can refer to the `addrobj` instead of the numeric IP address when you use any `ipadm` subcommand that manages addresses in the system, such as `ipadm show-addr`, or `ipadm delete-addr`.

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

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

- Display the general status of the interface.

  ```
  # ipadm show-if [interface]
  ```

  If you do not specify the interface, then information for all interfaces in the system is displayed.

- Display the interface's address information.

  ```
  # ipadm show-addr [addrobj]
  ```

If you do not specify the *addrobj*, then information for all address objects in the system is displayed.

For more information about the output of the `ipadm show-*` subcommand, see "Monitoring IP Interfaces and Addresses" in *Oracle Solaris Administration: Network Interfaces and Network Virtualization*.

7 **(Optional) Add entries for the IP addresses in the `/etc/hosts` file.**

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

Note – This step applies only if you are configuring static IP addresses that use hostnames. If you are configuring DHCP addresses, you do not need to update the `/etc/hosts` file.

Example 3–1 Configuring a Network Interface With a Static Address

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

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

# ipadm create-ip net3
# ipadm create-addr -T static -a 192.168.84.3/24 net3/v4static

# ipadm show-if
IFNAME   CLASS         STATE     ACTIVE     OVER
lo0      loopback      ok        yes        --
net3     ip            ok        yes        --

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

# vi /etc/hosts
# Internet host table
# 127.0.0.1       localhost
10.0.0.14       myhost
192.168.84.3    campus01
```

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

```
# ipadm create-addr -T static -a campus01 net3/v4static
```

Example 3–2 Automatically Configuring a Network Interface With an IP Address

This example uses the same network device as the previous example but configures the IP interface to receive its address from a DHCP server.

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

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

# ipadm create-ip net3

# ipadm create-addr -T dhcp net3/dhcp

# ipadm show-if
IFNAME   CLASS        STATE     ACTIVE     OVER
lo0      loopback     ok        yes        --
net3     ip           ok        yes        --

# ipadm show-addr net3/dhcp
ADDROBJ     TYPE       STATE     ADDR
net3/dhcp   dhcp       ok        10.8.48.242/24

# ipadm show-addr
ADDROBJ     TYPE       STATE     ADDR
lo0/?       static     ok        127.0.0.1/8
net3/dhcp   dhcp       ok        10.8.48.242/24
```

Setting Up System Configuration Modes

This section describes procedures to set up a system to run either in *local files mode* or *network client mode*. When running in local files mode, a system obtains all TCP/IP configuration information from files that are located in the local directory. In network client mode, the configuration information is provided to all the systems in the network by a remote network configuration server.

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

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

Clients can run in either mode. Thus, in the network you can have a combination of these modes with which different systems are configured, as shown in the following figure.

FIGURE 3–2 Systems in an IPv4 Network Topology Scenario

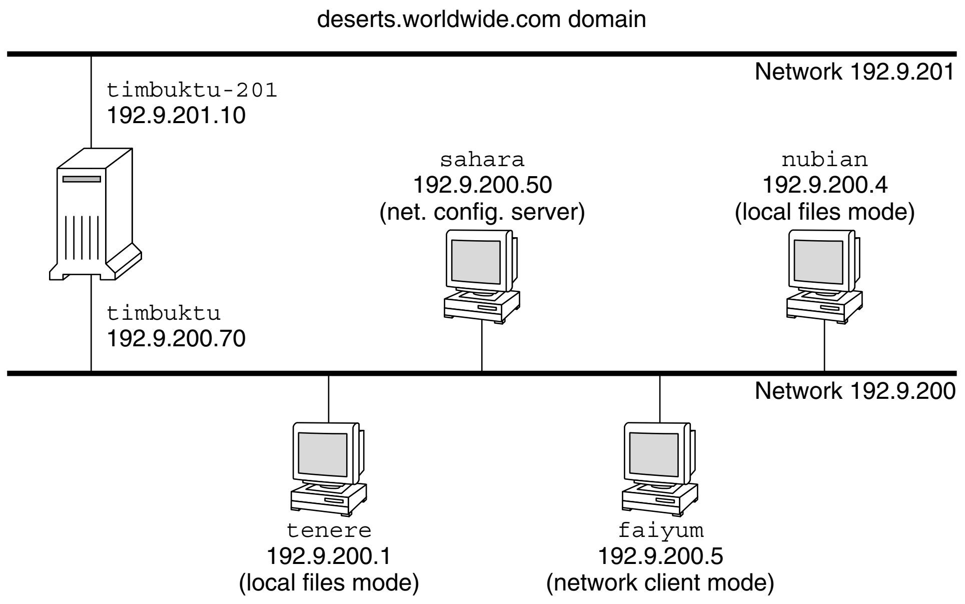


Figure 3–2 shows the systems in a `192.9.200` network.

- All the systems belong to the organizational domain `deserts.worldwide.com`.
- `sahara` is a configuration server. As a server, it runs in local files mode, where TCP/IP configuration information is obtained from the system's local disk.

Note – If you configure clients to run in network client mode, then you must configure at least one network configuration server that will provide configuration information to those clients.

- `tenere`, `nubian`, and `faiyum` are clients in the network. `tenere` and `nubian` run in local files mode. Regardless of `faiyum`'s local disk, the system is configured to operate in network client mode.
- `timbuktu` is configured as a router and therefore operates in local files mode. The system includes two NICs, each with its own configured IP interfaces. The first IP interface is named `timbuktu` and connects to the network `192.9.200`. The second IP interface is named `timbuktu-201` and connects to the network `192.9.201`.

For a more detailed overview of the two configuration modes, refer to "Determining Host Configuration Modes" in *System Administration Guide: IP Services*

▼ How to Configure a System for Local Files Mode

Use this procedure to configure any system to run in local files mode such as those that are listed in "Systems That Should Run in Local Files Mode" in *System Administration Guide: IP Services*.

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

Refer to "How to Configure an IP Interface" on page 47 for the procedure.

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

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.

This file must also include the name of the default router and the router's IP address.

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

b. **(Optional) If the `/usr` file system is NFS mounted, add the IP address or addresses of the file server, .**

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

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

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

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

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

---

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

To create entries, use the format *network-number netmask*. For example, 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 1–1. 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 the SMF property of the switch so that local files are searched first, then refresh the instance.**

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

7 **Reboot the system.**

How to Configure a System for Network Client Mode

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

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

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*.

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

Refer to "How to Configure an IP Interface" on page 47 for the procedure.

3 **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
```

4 **Remove any value that is assigned to the `domainname` property of the `nis/domain` SMF service.**

5 **Ensure that the search paths in the client's `name-service/switch` service reflect the same service requirements for your network.**

▼ How to Set Up a Network Configuration Server

Information for setting up installation servers and boot servers is found in *Installing Oracle Solaris 11 Systems*.

1 Become an administrator.

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*.

2 Turn on the `in.tftpd` daemon as follows:

a. Navigate to the root (/) directory of the designated network configuration server.

b. Create the `/tftpboot` directory:

```
# mkdir /tftpboot
```

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

c. Create a symbolic link to the directory.

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

3 Add the `tftp` line in the `/etc/inetd.conf` file.

The line should read 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`.

4 On the `/etc/hosts` database, add the host names and IP addresses of all the clients on the network.

5 On the `/etc/ethers` database, create entries for every system on the network that runs in network client mode.

Entries in this database use the following format:

MAC Address   *host name*   *#comment*

For more information, see the `ethers(4)` man page.

6 On the `/etc/bootparams` database, create an entry for every system on the network that runs in network client mode.

For information about editing this database, see the `bootparams(4)` man page.

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

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

8 **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 6, "Managing Services (Overview)," in *Oracle Solaris Administration: Common Tasks*.

Configuring an IPv4 Router

A router provides the interface between two or more networks. Therefore, 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 *Oracle Solaris Administration: Network Services*.

▼ 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 System for Local Files Mode" on page 53. This configuration ensures that routers boot if the network configuration server is down.

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*.

2 **For every NIC that is installed on the system, configure the IP interfaces as detailed in "How to Configure an IP Interface" on page 47.**

Make sure that each IP interface is configured with the IP address of the network for which the system will route packets. Thus, if the system serves the `192.168.5.0` and `10.0.5.0` networks, then one NIC must be configured for each network.

Caution – If you want to configure an IPv4 routers to use DHCP, you must be thoroughly knowledgeable with DHCP administration.

---

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

For example, assume that the names you assigned for the Router 1's two interfaces are `krakatoa` and `krakatoa-1`, respectively. The entries in the `/etc/inet/hosts` file would be as follows:

```
192.168.5.1      krakatoa        #interface for network 192.168.5.0
10.0.5.1         krakatoa-1      #interface for network 10.0.5.0
```

4 **Perform the rest of the steps to configure this router to run in local files mode.**

See "How to Configure a System for Local Files Mode" on page 53.

5 **If the router is connected to any subnetted network, add the network number and the netmask to the `/etc/inet/netmasks` file.**

For example, for traditional IPv4 address notation, such as `192.168.5.0`, you would type:

```
192.168.5.0    255.255.255.0
```

6 **Enable IPv4 packet forwarding on the router.**

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

7 **(Optional) Start a routing protocol.**

Use one of the following command syntaxes:

- `# routeadm -e ipv4-routing -u`
- `# svcadm enable route:default`

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

When you start a routing protocol, the routing daemon `/usr/sbin/in.routed` automatically updates the routing table, a process that is known as *dynamic routing*. For more information about the types of routing, see "Routing Tables and Routing Types" on page 59. For information about the `routeadm` command, see the `routeadm(1M)` man page.

Example 3–3 Configuring the Default Router for a Network

This example is based on Figure 3–1. 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 shows how to configure Router 2 to become the default router of the `172.20.1.0` network. The example also assumes that Router 2 has been configured to operate in local files mode, as described in "How to Configure a System for Local Files Mode" on page 53.

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
net0     phys      1500    up      --       --
net1     phys      1500    up      --       --
net2     phys      1500    up      --       --
# ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
net0/v4           static   ok           172.20.1.10/24
```

Only `net0` has been configured with an IP address. To make Router 2 the default router, you would physically connect the `net1` interface to the `10.0.5.0` network.

```
# ipadm create-ip net1
# ipadm create-addr -T static -a 10.0.5.10/24 net1/v4
# ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
net0/v4           static   ok           172.20.1.10/24
net1/v4           static   ok           10.0.5.10/24
```

Next, you would update the following network databases with information about the newly configured 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
# vi /etc/inet/netmasks
172.20.1.0    255.255.255.0
10.0.5.0      255.255.255.0
```

Finally, enable packet forwarding as well as 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 the `172.20.1.0` network 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 64.
- 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 59.

Routing Tables and Routing Types

Both routers and hosts maintain a *routing table*. 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   net0
224.0.0.0            10.0.5.100           U        1      0   net1
10.0.0.0             10.0.5.100           U        1      0   net1
127.0.0.1            127.0.0.1            UH       1     57   lo0
```

You can configure two types of routing on an Oracle Solaris system: static and dynamic. You can configure either or both routing types on a single system. A system that implements *dynamic routing* relies on routing protocols, such as RIP for IPv4 networks, and RIPng for IPv6 networks, to route network traffic as well as to update routing information in the table. With *static routing*, routing information is maintained manually by the use of 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 3–1 combines both static and dynamic routing.

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 14, "Introducing IPMP," in *Oracle Solaris Administration: 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   net0
   224.0.0.0            198.168.5.10          U      1   0       net0
   default              192.168.5.10          UG     1   91908
   127.0.0.1            127.0.0.1             UH     1   811302   lo0
   ```

2. **Become an administrator.**

   For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*.

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

   ```
   # route flush
   ```

4. **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.
`-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 3–4 Adding a Static Route to the Routing Table

The following example shows how to add a static route to Router 2 of Figure 3–1. The static route is needed for 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
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*. The interfaces of a multihomed host connect to different subnets, either on different physical networks, or on the same physical network.

On a system whose multiple interfaces connect to the same subnet, you must configure the interfaces into an IPMP group first. Otherwise, the system cannot be a multihomed host. For more information about IPMP, see Chapter 14, "Introducing IPMP," in *Oracle Solaris Administration: Network Interfaces and Network Virtualization*.

A multihomed host does not forward IP packets, but can be configured 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 *Oracle Solaris Administration: Security Services* or the documentation for your third-party firewall.

▼ How to Create a Multihomed Host

1. **Become an administrator.**

   For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*.

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

   Refer to "How to Configure an IP Interface" on page 47.

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

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

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

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

Use one of the following command syntaxes:

- `# routeadm -e ipv4-routing -u`
- `# svcadm enable route:default`

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

Example 3–5 Configuring a Multihomed Host

The following example shows how to configure the multihomed host that is shown in Figure 3–1. 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
net0      phys      1500    up      --       --
net1      phys      1500    up      --       --

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

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

```
# ipadm create-ip net1
# ipadm create-addr -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
net0/v4        static   ok           192.168.5.82/24
net1/v4        static   ok           192.168.5.85/24
```

Next, you would add the `net1` interface to the `/etc/hosts` database:

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

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

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

# ipadm set-prop -p forwarding=off ipv4

# 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 is currently enabled.

Configuring Routing for Single-Interface Systems

Single-interface systems can be configured with either static or dynamic routing. With static routing, the host must rely on the services of a default router for routing information. The following procedures contain the instructions for enabling both routing types.

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

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

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*.

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

For instructions, see "How to Configure an IP Interface" on page 47.

3 **With a text editor, create or modify the `/etc/defaultrouter` file by adding the IP address of the router the system will use.**

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

5 Ensure that routing is turned off.

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

           Routing services   "route:default ripng:default"

# svcadm disable route:default
```

6 Ensure that packet forwarding is turned off.

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

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

Example 3–6 Configuring Static Routing on a Single-Interface System

The following example shows how to configure static routing for hostb, a single-interface system on the 172.20.1.0 network as shown in Figure 3–1. hostb needs to use Router 2 as its default router. The example assumes that you have already configured the system's IP interface.

First, you would log in to hostb with administrator rights. Next, you would determine whether the /etc/defaultrouter file is present on the system:

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

# vi /etc/defaultrouter
172.20.1.10
```

The IP address 172.20.1.10 belongs to Router 2.

```
# 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

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

# ipadm set-prop -p forwarding=off ipv4

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

```
                IPv6 routing   disabled             disabled

            Routing services   "route:default ripng:default"

# svcadm disable route:default
```

How to Enable Dynamic Routing on a Single-Interface System

Dynamic routing that uses a routing protocol is the easiest way to manage routing on a system.

1. **Become an administrator.**

   For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*.

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

   For instructions, see "How to Configure an IP Interface" on page 47.

3. **Delete any entry in the `/etc/defaultrouter` file.**

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

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

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

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

   Use either of the following commands:

   - `# routeadm -e ipv4-routing -u`
   - `# svcadm enable route:default`

Example 3–7 Running Dynamic Routing on a Single-Interface System

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

First, you would log in to `hosta` with administrator rights. Then, you would determine whether the `/etc/defaultrouter` file is present on the system:

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

# cat defaultrouter
192.168.5.10
```

The file correctly includes the entry `192.168.5.10`, which is the IP address for Router 1.

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

           Routing services   "route:default ripng:default"

# svcadm enable route:default

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

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

Adding a Subnet to a Network

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 list. The list assumes that you have already prepared a subnet schema. For an overview, see "What Is Subnetting?" in *System Administration Guide: IP Services*.

- Assign the IP addresses with the new subnet number to the systems that belong to the subnet.

  For reference, see "How to Configure an IP Interface" on page 47.
- Add the correct IP address and netmask to each system's `/etc/netmasks` file.
- Revise each system's `/etc/inet/hosts` file with the correct IP address to correspond to the host names.
- Reboot all the systems in the subnet.

The following procedure is closely connected to subnets. If you implement subnetting much later after you have originally configured the network without subnetting, perform the following procedure to implement the changes.

▼ 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" on page 47.

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" in *System Administration Guide: IP Services*.

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*.

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

With the `ipadm` command, you cannot modify an IP address directly. You first delete the address object that represents the IP address you want to modify. Then you assign a new address by using the same address object name.

```
# ipadm delete-addr addrobj
# ipadm create-addr -T static IP-address  addrobj
```

3 **If applicable, modify the host name in the `/etc/inet/hosts` file or equivalent `hosts` database.**

4 **If applicable, modify the host name entry in the `system/identity: node` SMF service:**

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

5 **If the subnet mask has changed, modify the subnet entries in the `/etc/netmasks` file.**

6 **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.**

7 **Reboot the system.**

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

Example 3–8 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 delete-addr bge0/v4
# ipadm create-addr -T static -a 192.168.34.100/24 bge0/v4
# 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 *Oracle Solaris Administration: Common Tasks* and "How to Configure an IP Interface" on page 47.

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 6, "Managing Services (Overview)," in *Oracle Solaris Administration: Common Tasks*.

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 140.

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

1. **Become an administrator.**

   For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*.

2. **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 *Oracle Solaris Administration: Common Tasks*.
- 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 3–9 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 *Oracle Solaris Administration: Network Services*.

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*.

2 **Set TCP wrappers to enabled.**

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

3 **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 4

Enabling IPv6 on the Network

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

- "Configuring an IPv6 Interface" on page 75
- "How to Configure a System For IPv6" on page 76
- "Configuring an IPv6 Router" on page 78
- "Modifying an IPv6 Interface Configuration for Hosts and Servers" on page 80
- "Configuring Tunnels (Task Map)" on page 122
- "Configuring Name Service Support for IPv6" on page 86

For various types of information about IPv6, refer to the following resources:

- For an overview of IPv6 concepts:Chapter 3, Introducing IPv6 (Overview)
- For IPv6 planning tasks:Chapter 2, "Considerations When Using IPv6 Addresses"
- For preparation to use IP tunnels: "Planning for Tunnel Use in the Network" on page 41
- For reference information: Chapter 9, "IPv6 Reference"

Configuring an IPv6 Interface

As an initial step to use IPv6 on a network, configure IPv6 on the system's IP interface.

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

- The `name-service/switch` SMF service 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 after Oracle Solaris installation has been completed.

▼ How to Configure a System For IPv6

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" in *System Administration Guide: IP Services*. 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 78.

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

1. **Configure the IP interface by using the appropriate commands.**

   Refer to "How to Configure an IP Interface" on page 47.

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

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

   To add more addresses, use the following syntax:

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

   ---

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

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

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 80. For details about `/etc/inet/ndpd.conf`, refer to the `ndpd.conf(4)` man page and "`ndpd.conf` Configuration File" on page 146.

5. **(Optional) To display the status of the IP interfaces with their IPv6 configurations, type the following command:**

   ```
   # ipadm show-addr
   ```

Example 4–1 Enabling an IPv6 Interface After Installation

This example shows how to enable IPv6 on the `net0` 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
net0/v4    static   ok       172.16.27.74/24
```

Only the `net0` interface is currently configured for this system. Enable IPv6 on this interface as follows:

```
# ipadm create-addr -T addrconf net0/v6
# ipadm create-addr -T static -a 2001:db8:3c4d:15:203/64 net0/v6add
# /usr/lib/inet/in.ndpd

# ipadm show-addr
ADDROBJ       TYPE       STATE    ADDR
lo0/v4        static     ok       127.0.0.1/8
net0/v4       static     ok       172.16.27.74/24
net0/v6       addrconf   ok       fe80::203:baff:fe13:14e1/10
lo0/v6        static     ok       ::1/128
net0/v6add    static     ok       2001:db8:3c4d:15:203/64

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

Next Steps

- To configure the IPv6 node as a router, go to "Configuring an IPv6 Router" on page 78.
- To disable address autoconfiguration on the node, see "How to Turn Off IPv6 Address Autoconfiguration" on page 77.
- To tailor the node as a server, see the suggestions in "Administering IPv6-Enabled Interfaces on Servers" on page 85.

▼ 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 83.

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 146.

2 Update the IPv6 daemon with your changes.

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

Configuring an IPv6 Router

This section describes tasks to configure an IPv6 router. Depending on your site requirements, you might need to perform only selected tasks.

▼ How to Configure an IPv6-Enabled Router

The following procedure assumes that you have already configured the system for IPv6. For the procedures, refer to "Configuring an IPv6 Interface" on page 75.

1 Configure IPv6 packet forwarding on all interfaces of the router.

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

2 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:

  ```
  # routeadm -e ipv6-routing -u
  ```
- Use the appropriate SMF command:

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

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

3 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 146 and the `ndpd.conf(4)`man page.

4 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.

5 **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 `net0` and `net1`.

```
ifdefault AdvSendAdvertisements true
prefixdefault AdvOnLinkFlag on AdvAutonomousFlag on

if net0 AdvSendAdvertisements 1
prefix 2001:0db8:3c4d:15::0/64 net0

if net1 AdvSendAdvertisements 1
prefix 2001:0db8:3c4d:16::0/64 net1
```

6 **Reboot the system.**

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

Example 4–2 `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 78 procedure.

```
ADDROBJ      TYPE       STATE   ADDR
lo0/v4       static     ok      127.0.0.1/8
net0/v4      static     ok      172.16.15.232/24
net1/v4      static     ok      172.16.16.220/24
net0/v6      addrconf   ok      fe80::203:baff:fe11:b115/10
lo0/v6       static     ok      ::1/128
net0/v6add   static     ok      2001:db8:3c4d:15:203:baff:fe11:b115/64
net1/v6      addrconf   ok      fe80::203:baff:fe11:b116/10
net1/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 `net0/mystring`, `net0/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 121.

- 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 80.
- To improve IPv6 support on servers, refer to "Administering IPv6-Enabled Interfaces on Servers" on page 85.
- For detailed information about IPv6 commands, files, and daemons, refer to "Oracle Solaris IPv6 Implementation" on page 145.

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" in *System Administration Guide: IP Services*. However, you can modify the IPv6 address of an interface, if necessary, as explained in the tasks of this section.

You need to perform three general tasks in the following sequence:

1. Turn off IPv6 address autoconfiguration. See "How to Turn Off IPv6 Address Autoconfiguration" on page 77.
2. Create a temporary address for a host. See "How to Configure a Temporary Address" on page 81.
3. Configure an IPv6 token for the interface ID. See "How to Configure a User-Specified IPv6 Token" on page 84.

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 interface on an IPv6 node that you want to keep anonymous. For example, you might want to use temporary addresses for the interfaces of a host that needs to access public web servers. Temporary addresses implement IPv6 privacy enhancements. These enhancements are described in RFC 3041, 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 Configure a System For IPv6" on page 76.

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 109.

---

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 4–4.**

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

Example 4–3 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 4–4 `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
net0/v6   addrconf ok    U----   ---        fe80::a00:20ff:feb9:4c54/10
net0/v6a  static   ok    U----   ---        2001:db8:3c4d:15:a00:20ff:feb9:4c54/64
net0/?    addrconf ok    U--t-   ---        2001:db8:3c4d:15:7c37:e7d1:fc9c:d2cb/64
```

Note that for the address object `net0/?`, the `t` flag is set under the `CURRENT` field. 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 86.
- To configure IPv6 addresses for a server, see "How to Configure a User-Specified IPv6 Token" on page 84.
- To monitor activities on IPv6 nodes, see Chapter 5, "Administering a TCP/IP Network."

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. 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 and that no IPv6 addresses are configured on the interface..**

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

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

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

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

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

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 `net0` with a token:

   ```
   # ipadm create-addr -T addrconf -i ::1a:2b:3c:4d/64 net0/v6add
   ```

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

   ---

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

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

Example 4–5 Configuring a User-Specified Token on an IPv6 Interface

The following example shows `net0` being configured with an IPv6 address and a token.

```
# ipadm show-if
IFNAME   CLASS      STATE    ACTIVE     OVER
```

```
lo0       loopback   ok       yes        ---
net0      ip         ok       yes        ---

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

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

After the token is configured, the address object `net0/v6` has both a link local address as well as an address with `1a:2b:3c:4d`configured for its interface ID. Note that this token can no longer be modified for this interface after `net0/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 86.
- To monitor server performance, see Chapter 5, "Administering a TCP/IP Network."

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

This procedure provides general steps to enable IPv6 on your network's servers. Some of the steps might vary depending on the manner that you want to implement IPv6.

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

   For procedures, refer to "Configuring an IPv6 Interface" on page 75.

2. **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 78.

3. **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" in *System Administration Guide: IP Services*.
- 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 81.
- 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 84.

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:**

hostname `IN` `AAAA` *host-address*

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

hostaddress `IN` `PTR` *hostname*

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

Example 4–6 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
>hostname
```

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

```
>set q=AAAA
hostname
```

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

Example 4–7 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 4–8 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 5

Administering a TCP/IP Network

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 *Oracle Solaris Administration: 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 2, "Considerations When Using IPv6 Addresses."
- To configure IPv6 and create a dual-stack network environment, refer to Chapter 4, "Enabling IPv6 on the Network."

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 91
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 94
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 93
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 96
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 99
Monitor network traffic.	Displays all IP packets by using the `snoop` command.	"How to Monitor IPv6 Network Traffic" on page 105
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 102

Note – To monitor network interfaces, refer to "Monitoring IP Interfaces and Addresses" in *Oracle Solaris Administration: 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 5–1 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 5–2 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 5–3 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 5–4 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
net0  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
net0  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 5–5 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 5–6 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 5–7 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 5–8 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  net0
10.0.0.14            myhost               U         1      0  net0
default              distantrouter        UG        1      2  net0
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      net0:1
fe80::/10             fe80::1a2b:3c4d:5e6f:12a2           U     1   23     net0
ff00::/8              fe80::1a2b:3c4d:5e6f:12a2           U     1   0      net0
default               fe80::1a2b:3c4d:5e6f:12a2           UG    1   0      net0
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 5–9 `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 *Oracle Solaris Administration: Network Interfaces and Network Virtualization* and "Monitoring Network Status With the `netstat` Command" on page 91.

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 5–10 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
  net0/v4       static     ok       10.46.86.54/24
  lo0/v6        static     ok       ::1/128
  net0/v6       addrconf   ok       fe80::a00:fe73:56a8/10
  net0/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
  net0/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 5–11 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 100.

```
-- 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 net0 #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  net0  <NET_SYN>
Add    10.10.48.85/25  -->10.10.48.112      metric=0  net0  <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 5–12 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 net0
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 net0
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 5–13 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 -ahost-name
```

You can run this form of the `traceroute` command from your user account.

Example 5–14 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 5–15.**

3 **Use Control-C to halt the process.**

Example 5–15 Output From the snoop Command

The basic snoop command returns output that resembles the following, for a dual-stack host.

```
% snoop
Using device /dev/net (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 5–16 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 5–17 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 of Checking Packets

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
net0/v4      static     ok      192.68.25.5/24
lo0/?        static     ok      127.0.0.1/8
net0/?       static     ok      172.0.0.3/24
net0/?       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 5–18 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 5–19 Packet Flow in the net0 Device in Local Zones

```
# snoop -I net0
Using device ipnet/net0 (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 `net0` 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 net0 -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 5–19 Packet Flow in the net0 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 5–20 Observing Traffic by Identifying the Zone

```
# snoop -I hme0 sandboxsnoop -I net0 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 149.

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 6

Configuring IP Tunnels

This chapter contains descriptions of IP tunnels as well as procedures for configuring and maintaining tunnels in Oracle Solaris.

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.

IP Tunnel Administration in This Oracle Solaris Release

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 more information about the `dladm` subcommands, see the `dladm`(1M) man page.

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.

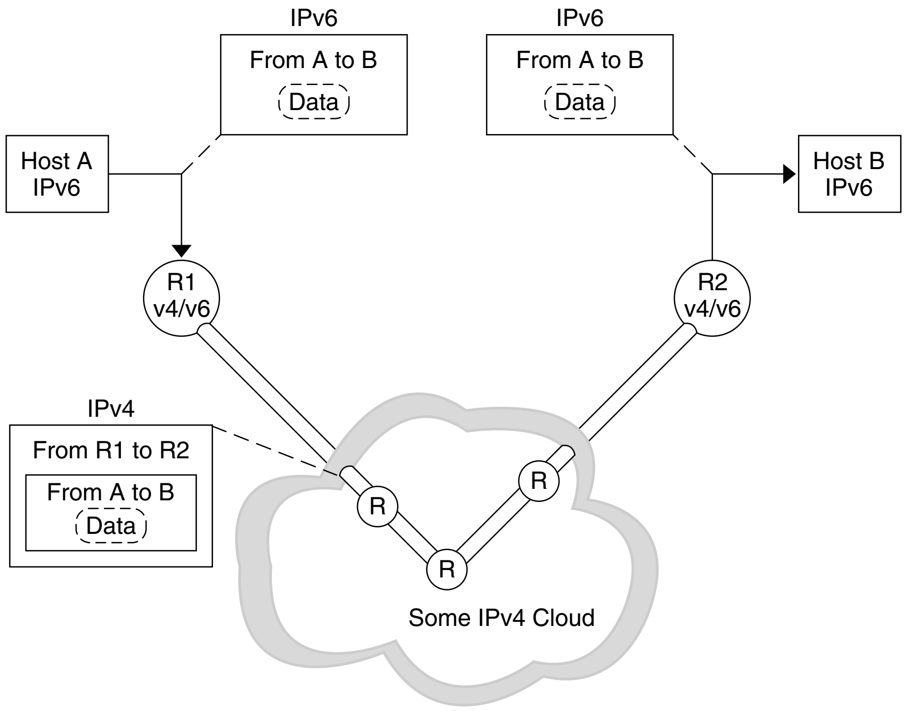
Tunnels differ depending on the type of packet encapsulation. The following types of tunnels are supported in Oracle Solaris:

- *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 6–1 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

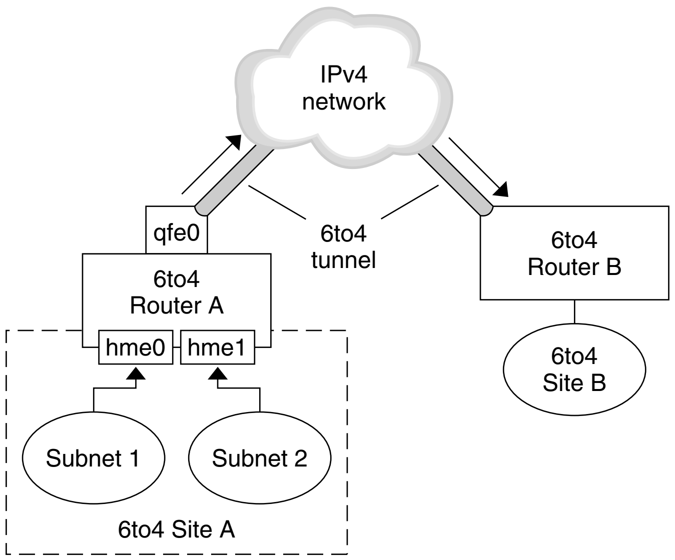
The following table describes additional tasks to configure 6to4 tunnels and the resources to obtain additional useful information.

Task or Detail	For Information
Tasks for configuring a 6to4 tunnel	"How to Configure a 6to4 Tunnel" on page 127
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 6–2 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 6–2. Moreover, the scenario assumes that the 6to4 routers and the 6to4 hosts are already configured.

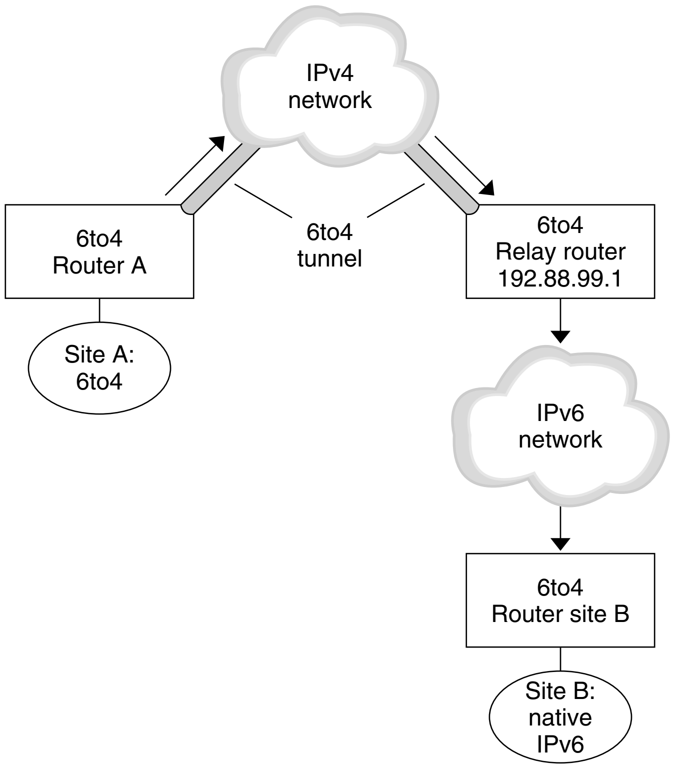
1. A host on Subnet 1 of 6to4 Site A sends a transmission, with a host at 6to4 Site B as the destination. Each packet header has a 6to4-derived source address and 6to4-derived destination address.

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 6–3 Tunnel From a 6to4 Site to a 6to4 Relay Router



In Figure 6–3, 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 6–3.

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 Tunnel Use in the Network” on page 41.

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 6–1 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 with the `ipadm` command.

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

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.

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 15, "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 123
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 130
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 131
Delete a tunnel.	Delete a tunnel configuration.	"How to Delete an IP Tunnel" on page 133

▼ 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 *Oracle Solaris Administration: 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, "Datalink and Interface Configuration," in *Oracle Solaris Administration: 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 6–1 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" in *System Administration Guide: IP Services*.

Example 6–2 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 15, "Configuring IPsec (Tasks)."

Example 6–3 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 162.

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 *Oracle Solaris Administration: 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 86.

Example 6–4 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" in *System Administration Guide: IP Services*.

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 137.

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 123
- 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 6–5 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 6–6 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 6–7 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 6–8 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 6–9 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 6–10 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 7

Troubleshooting Network Problems

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 91 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 "Network Configuration Files" on page 139 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 2, "Considerations When Using IPv6 Addresses."

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 121.

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 8

IPv4 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 chapter contains the following information:

Network Configuration Files

In a network, configuration information is stored in different files and databases that regulate the way the network operates. This section provides a brief description of these files. Some files require updating and maintenance as you implement changes to the network. Other files require little or no administration.

`/etc/defaultrouter`	This file contains the IP interface names of the routers that are directly connected to the network. The existence of this file in the system is optional. If the file exists, then the system is configured to support static routing.
`/etc/inet/hosts`	This file contains the IPv4 addresses in the network together with the corresponding interface names on which the addresses are configured. If you are using NIS or DNS name service, or the LDAP directory service, then the host information is stored in a different database, such as `hosts.byname,` that exists in the servers. For more information, see *Oracle Solaris Administration: Naming and Directory Services*.
`/etc/inet/netmasks`	This file contains the network number, such as `192.168.0.0`, and the netmask information of that network number, such as

	255.255.255.0. In a network that uses NIS or LDAP, this information is stored in a netmask database in the servers. See the netmasks(4) man page for more information.
/etc/bootparams	This file contains parameters that determine the boot processes for systems that are configured to boot in network client mode. For more information see "Setting Up System Configuration Modes" on page 51. The file is the basis for the creation of the bootparams database that the name service uses if you are not using the local files mode. To obtain specific information about the content and format of this file, refer to the bootparams(4) man page.
/etc/ethers	The file associates hostnames with their MAC addresses. The file is the basis for the creation of an ethers database for use in the network where systems are configured as network clients. For more information, see the ethers(4) man page.
/etc/inet/networks	This file associates network names and network numbers. Comments can also be added to further clarify each entry in the database. This file enables applications to use and display the network names instead of network numbers. For example, the netstat program uses the information in this database to produce status tables. All the subnetworks that connect to the local network through routers must be included in this file. For more information, see the networks(4) man page.
/etc/inet/protocols	This file lists the TCP/IP protocols that are installed on your system as well as their protocol numbers. This file seldom requires any administration. For more information, see the protocols(4) man page.
/etc/inet/services	This file lists the names of TCP and UDP services and their well-known port numbers. The list is used by programs that call network services. Generally, this file does not require any administration. For more information, see the services(4) man page.

`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 *Oracle Solaris Administration: Common Tasks*.
- 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 70.
- For information on adding services that handle both IPv4 requests and IPv6 requests, refer to "`inetd` Internet Services Daemon" on page 140

The name-service/switch SMF Service

The `name-service/switch` SMF service defines the search order of the network databases for configuration information. Some of the network configuration information that previously were stored in configuration files, such as the default domain, have been converted to become properties of this SMF service. The properties of this SMF service determines the implementation of the name services on the system. The properties are listed as follows:

```
% svccfg -s name-service/switch listprop config
config                       application
config/value_authorization   astring         solaris.smf.value.name-service.switch
config/default               astring         files
config/password              astring         "files nis"
config/group                 astring         "files nis"
config/host                  astring         "files dns nis"
config/network               astring         "nis [NOTFOUND=return] files"
config/protocol              astring         "nis [NOTFOUND=return] files"
config/rpc                   astring         "nis [NOTFOUND=return] files"
config/ether                 astring         "nis [NOTFOUND=return] files"
config/netmask               astring         "files nis"
config/bootparam             astring         "nis [NOTFOUND=return] files"
config/publickey             astring         "nis [NOTFOUND=return] files"
config/netgroup              astring         nis
config/automount             astring         "files nis"
config/alias                 astring         "files nis"
config/service               astring         "files nis"
config/printer               astring         "user nis"
config/auth_attr             astring         "files nis"
config/prof_attr             astring         "files nis"
config/project               astring         "files nis"
```

The values that are set for each of the properties determine which name service to search for information that would affect network users, such as passwords, aliases, or network masks. In the example, the automount and password properties are set to `files` and `nis`. Thus, automount information and password information are obtained from files and from the NIS service.

If you want to change from one name service to another name service, you must set the appropriate properties of the `name-service/switch` SMF service to enable the selected name service.

For example, suppose that you want to use LDAP naming service on your network. The following properties of the SMF service need to be configured;

- `config/default` needs to be set to use files and LDAP.
- `config/host` needs to be set to use files and DNS.
- `config/netgroup` needs to be set to use LDAP.
- `config/printer` needs to be set to use user, files and LDAP.

Therefore, you need to type the following commands to set these properties correctly.

```
# svccfg -s name-service/switch setprop config/default = astring: "files ldap"
# svccfg -s name-service/switch setprop config/host = astring: "files dns"
# svccfg -s name-service/switch setprop config/netgroup = astring: "ldap"
# svccfg -s name-service/switch setprop config/printer = astring: "user files ldap"
# svccfg -s name-service/switch:default refresh
```

For complete details on the name service switch, refer to *Oracle Solaris Administration: Naming and Directory Services*.

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.

Refer to *Oracle Solaris Administration: Naming and Directory Services* for information on network databases correspondences in NIS, DNS, and LDAP.

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 8–1 and Table 8–2.

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.

Tables of Routing Protocols in Oracle Solaris

The following table lists all the routing protocols that are supported in Oracle Solaris

TABLE 8–1 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 56
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 64 and "How to Enable Dynamic Routing on a Single-Interface System" on page 66
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 78
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 75

The following table lists the Quagga protocols that are also supported in Oracle Solaris.

TABLE 8–2 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.

CHAPTER 9

IPv6 Reference

This chapter contains the following reference information about Oracle Solaris IPv6 implementation.

- "Oracle Solaris IPv6 Implementation" on page 145
- "IPv6 Neighbor Discovery Protocol" on page 156
- "IPv6 Routing" on page 162
- "IPv6 Extensions to Oracle Solaris Name Services" on page 164
- "NFS and RPC IPv6 Support" on page 164
- "IPv6 Over ATM Support" on page 165

For an overview of IPv6, refer to Chapter 3, Introducing IPv6 (Overview). For tasks on configuring an IPv6-enabled network, refer to Chapter 4, "Enabling IPv6 on the Network." For all information about IP tunnels, refer to Chapter 6, "Configuring IP Tunnels."

Oracle Solaris IPv6 Implementation

This section describes the files, commands, and daemons that enable IPv6 in Oracle Solaris. For an in depth overview of IPv6 addressing and IPv6 header formatting, see "IPv6 Addressing Formats Beyond the Basics" in *System Administration Guide: IP Services*.

IPv6 Configuration Files

This section describes the configuration files that are part of an IPv6 implementation:

- "`ndpd.conf` Configuration File" on page 146
- "`/etc/inet/ipaddrsel.conf` Configuration File" on page 149

`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 9–1 `/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 9–2 `/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.

TABLE 9–2 /etc/inet/ndpd.conf Interface Configuration Variables *(Continued)*

Variable	Default	Definition
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.
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 78.
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 84.
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 81.
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 81.
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 81.

TABLE 9–2 `/etc/inet/ndpd.conf` Interface Configuration Variables *(Continued)*

Variable	Default	Definition
`TmpValidLifetime`	False	Sets the valid lifetime for a temporary address. For more information, refer to "How to Configure a Temporary Address" on page 81.

The next table shows the variables that are used for configuring IPv6 prefixes.

TABLE 9–3 `/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 9–1 `/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
```

EXAMPLE 9–1 `/etc/inet/ndpd.conf` File *(Continued)*

```
#
#
#AdvValidLifetime
#AdvOnLinkFlag
#AdvPreferredLifetime
#AdvAutonomousFlag
#AdvValidExpiration
#AdvPreferredExpiration

ifdefault AdvReachableTime 30000 AdvRetransTimer 2000
prefixdefault AdvValidLifetime 240m AdvPreferredLifetime 120m

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 9–4.

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 109. For more information on `ippaddrsel.conf`, refer to “Reasons for Modifying the IPv6 Address Selection Policy Table” on page 150 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 9–4 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 118. 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 123.

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 9–2 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 9–3 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 9–4 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 94 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 105 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 97.

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 101.

IPv6-Related Daemons

This section discusses the IPv6-related daemons.

in.ndpd Daemon, for Neighbor Discovery

Thein.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 9–1 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 70.

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" in *System Administration Guide: IP Services*.

This section discusses the following features of the Neighbor Discovery protocol:

- "ICMP Messages From Neighbor Discovery" on page 157
- "Autoconfiguration Process" on page 157
- "Neighbor Solicitation and Unreachability" on page 159
- "Duplicate Address Detection Algorithm" on page 160
- "Comparison of Neighbor Discovery to ARP and Related IPv4 Protocols" on page 161

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 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).

P A R T I I

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 10

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:

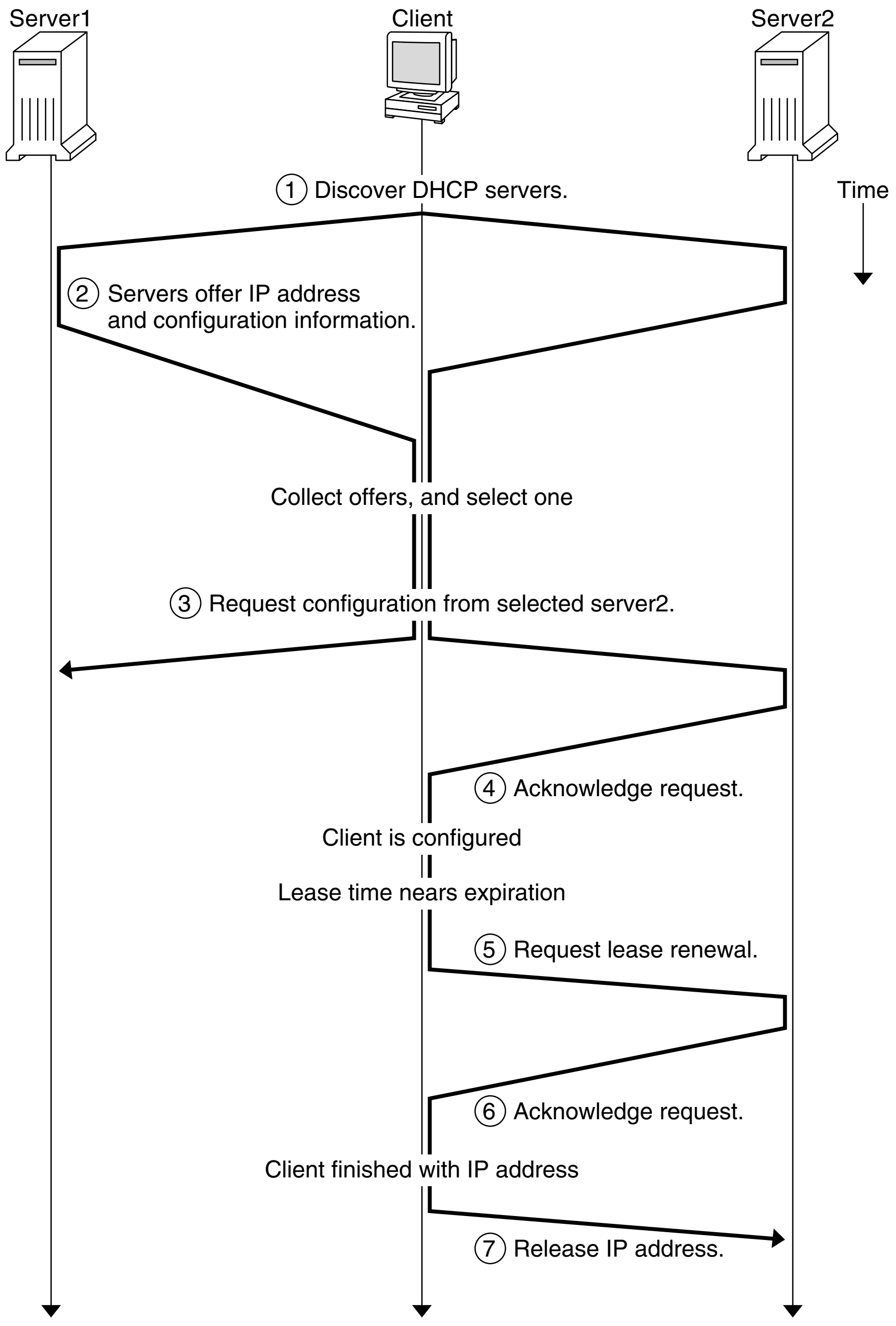
- **IP address management** – A primary advantage of DHCP is easier management of IP addresses. In a network without DHCP, you must manually assign IP addresses. You must be careful to assign unique IP addresses to each client and to configure each client individually. If a client moves to a different 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 10–1 Sequence of Events for DHCP Service


Server1
Client
Server2
1 Discover DHCP servers.
Time
2 Servers offer IP address and configuration information.
Collect offers, and select one
3 Request configuration from selected server2.
4 Acknowledge request.
Client is configured
Lease time nears expiration
5 Request lease renewal.
6 Acknowledge request.
Client finished with IP address
7 Release IP address.

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.

ISC DHCP Server

An implementation of the Internet Systems Consortium (ISC) DHCP server has been added to Oracle Solaris. Because this software is not automatically installed, you can add this server to your system by typing the following command:

```
# pkg install pkg:/service/network/dhcp/isc-dhcp
```

The ISC DHCP server, `dhcpd`, implements the Dynamic Host Configuration Protocol (DHCP) and the Internet Bootstrap Protocol (BOOTP). DHCP allows hosts on a TCP/IP network to request and be assigned IP addresses, and also to discover information about the network to which they are attached. BOOTP provides similar functionality.

The following lists some of the important additions to the release for DHCP:

- Several services have been added to support ISC DHCP and the legacy Sun DHCP service. See “SMF Services Used by the DHCP Service” on page 202 for a list of all of the services used by DHCP.
- Three commands have been added: `dhcpd`, `dhcprelay`, and `omshell`. See “Files Used by the DHCP Service” on page 200 for a list of all of the commands that are associated with DHCP.
- For ISC DHCP, the server configuration files are `/etc/inet/dhcpd4.conf` for DHCPv4 and `/etc/inet/dhcpd6.conf` for DHCPv6.
- A user called `dhcpserv` has been added for the ISC DHCP service.
- Access to the commands three new commands can be managed by using the `solaris.smf.manage.dhcp` and `solaris.smf.value.dhcp` authorizations.

For more information about ISC DHCP, see the ISC DHCP Documentation web page.

Legacy Sun DHCP Server

The legacy Sun DHCP server software is still included in the Oracle Solaris 11 release, but it has been marked as obsolete and will be removed in a future release. For more information about the legacy DHCP service, see Chapter 11, Administering the ISC DHCP Service.

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. The DHCP client can interoperate with both the legacy Sun DHCP server and the ISC DHCP server.

See Chapter 12, "Configuring and Administering the DHCP Client," for detailed information about the DHCP client.

CHAPTER 11

Administering the ISC DHCP Service

This chapter describes tasks that you might find useful when you administer the ISC DHCP service. The following topics are covered:

Setting Up User Access to DHCP Commands

By default, only the `root` user can execute `svcadm` and other commands that are required to configure the DHCP service. 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 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 User Access to DHCP Commands

1 **Become superuser or assume a role or user name that has been assigned to the DHCP Management profile.**

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

2 **Add a user or role to the `/etc/user_attr` file.**

Edit the `/etc/user_attr` file 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
```

DHCP Server Tasks

▼ How to Configure an ISC DHCP Server

You can use these steps to initially configure an ISC DHCP server.

1 **Become superuser or assume a role or user name that has been assigned to the DHCP Management profile.**

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

2 **Edit the DHCP configuration file.**

Create either the `/etc/dhcp/dhcpd4.conf` or `/etc/dhcp/dhcpd6.conf` file. For more information, see the `dhcpd.conf`(5) man page.

3 **Enable the required service.**

```
# svcadm enable service
```

service can be one of the following values:

`svc:/network/dhcp/server:ipv4`	Provides DHCP and BOOTP requests from IPv4 clients
`svc:/network/dhcp/server:ipv6`	Provides DHCP and BOOTP requests from IPv6 clients
`svc:/network/dhcp/relay:ipv4`	Relays DHCP and BOOTP requests from IPv4 clients to a network with a DHCP server
`svc:/network/dhcp/relay:ipv6`	Relays DHCP and BOOTP requests from IPv6 clients to a network with a DHCP server

▼ How to Modify the Configuration of the DHCP Service

1 **Become superuser or assume a role or user name that has been assigned to the DHCP Management profile.**

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

2 **Edit the DHCP configuration file.**

Edit either the `/etc/dhcp/dhcpd4.conf` or `/etc/dhcp/dhcpd6.conf` file. For more information, see the `dhcpd.conf`(5) man page.

3 **Refresh the SMF data.**

```
# svcadm refresh service
```

◆ ◆ ◆ CHAPTER 12

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. If you install Oracle Solaris by using the LiveCD GUI installer, then the DHCPv4 and DHCPv6 protocols are enabled on the installed system. If you install Oracle Solaris by using the text installer, you are prompted to select how the network should be configured on the installed system. If you specify Automatic Network Configuration, then the DHCPv4 and DHCPv6 protocols are enabled on the installed system.

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 189 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.

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 DHCP 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 by using 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 a DHCP Client" on page 189.

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:

```
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 171.

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 10, "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 196 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 a file, if the file exists. The filename for DHCPv4 is `/etc/dhcp/`*interface*`.dhc`, and `/etc/dhcp/`*interface*`.dh6` is for DHCPv6. By default, the lease is saved rather than released, so the DHCP server can not detect 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 171. 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 a DHCP Client

This procedure is necessary only if DHCPv4 was not enabled during Oracle Solaris installation. It is never necessary for DHCPv6.

1. **Become superuser or assume a role or user name that has been assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 177.

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

2. **Reconfigure the system.**

   Choose one of the following configuration methods:

   - **Interactively reconfigure the system.**

     ```
     # sysconfig configure
     ```

     When the System Configuration Interactive Tool starts, select Automatic network configuration on the Network screen.

   - **Non-interactively reconfigure the system.**

     ```
     # sysconfig configure -c sc_profile
     ```

     See the `sysconfig`(1M) man page for more information about using the `sc_profile` configuration file.

▼ How to Disable a DHCP Client

1. **Become superuser or assume a role or user name that has been assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 177.

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

2. **Reconfigure the system.**

   Choose one of the following configuration methods:

   - **Interactively reconfigure the system.**

     ```
     # sysconfig configure
     ```

     When the System Configuration Interactive Tool starts, select either Manual or None as the network configuration on the Network screen.

   - **Non-interactively reconfigure the system.**

     ```
     # sysconfig configure -c sc_profile
     ```

     See the `sysconfig`(1M) man page for more information about using the `sc_profile` configuration file.

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:

- **Create the IP interface** – The command `ipadm create-ip` creates the IP interface which you then configure with IP addresses. The addresses can either be static or dynamic. Creating the IP interface is a prerequisite command before you can assign the addresses.

- **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 `ifconfig [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 193.

- 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 the `dhcpd`(8) man page and Working With DHCP Macros (Task Map) 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 192.
- 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 194 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.

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 196.

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 checks the value that is set in the `config/nodename` property in the `svc:/system/identity:node` service 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 value that is set in the `config/nodename` property in the `svc:/system/identity:node` service. 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

The steps to perform depend on whether an IP interface already exists with a DHCP address.

1. **If the IP interface already exists with a DHCP address, do the following:**

   a. **Delete the existing DHCP address.**

   ```
   # ipadm delete-addr -r dhcp-addrobj
   ```

   b. **Register a new DHCP address with a specific host name that you want to use.**

   ```
   # ipadm create-addr -T dhcp -h hostname  dhcp-addrobj
   ```

2. **If the IP interface does not yet exist, do the following:**

   a. **Create the IP interface.**

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

   b. **Register a DHCP address with a specific host name that you want to use.**

   ```
   # ipadm create-addr -T dhcp -h hostname  dhcp-addrobj
   ```

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` SMF service must also be appropriately configured. See the `nsswitch.conf(4)` man page for more information.

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 cross-references to documentation that can help you set up clients for each name service.

TABLE 12–1 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:
	
	`# /usr/sbin/dhcpinfo NISdmain`
	
	`# /usr/sbin/dhcpinfo NISservs`
	
	Note – For DHCPv6, include `-v6` and different protocol keywords in the command as follows:
	
	`# /usr/sbin/dhcpinfo -v6 NISDomain`
	
	`# /usr/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, then properties of the `dns/client` SMF service are also automatically configured. 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 interface is first configured for DHCP, the `dhcpagent` invokes the event script as follows:

```
/etc/dhcp/eventhook net0 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.

CHAPTER 13

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" on page 199
- "Files Used by the DHCP Service" on page 200
- "SMF Services Used by the DHCP Service" on page 202

DHCP Commands

The following table lists the commands that you can use to manage DHCP on your network.

TABLE 13–1 Commands Used in DHCP

Command	Description
`/usr/lib/inet/dhcpd`	ISC DHCP only: The ISC DHCP server daemon. For more information, see the `dhcpd`(8) man page.
`/usr/lib/inet/dhcrelay`	ISC DHCP only: Enables a means for relaying DHCP and BOOTP requests from a client on a network with no DHCP servers to servers on other networks. For more information, see the `dhcrelay`(8) man page.
`/usr/lib/inet/in.dhcpd`	Legacy Sun DHCP only: The legacy Sun 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. For more information, see the `in.dhcpd`(1M) man page.
`/usr/sadm/admin/bin/dhcpmgr`	Legacy Sun DHCP only: DHCP Manager, a graphical user interface (GUI) tool used to configure and manage the DHCP service. DHCP Manager is the recommended DHCP management tool. For more information, see the `dhcpmgr`(1M) man page.

TABLE 13–1 Commands Used in DHCP *(Continued)*

Command	Description
`/usr/sbin/dhcpagent`	The DHCP client daemon, which implements the client side of the DHCP protocol. For more information, see the `dhcpagent(1M)` man page.
`/usr/sbin/dhcpconfig`	Legacy Sun DHCP only: 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. For more information, see the `dhcpconfig(1M)` man page.
`/usr/sbin/dhcpinfo`	Legacy Sun DHCP only: 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. For more information, see the `dhcpinfo(1)` man page.
`/usr/sbin/dhtadm`	Legacy Sun DHCP only: Used to make changes to the options and macros in the `dhcptab` table. 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. For more information, see the `dhtadm(1M)` man page.
`/usr/sbin/ipadm`	Used at system boot to assign IP addresses to network interfaces, configure network interface parameters, or both. On a DHCP client, `ipadm` starts DHCP to get the parameters (including the IP address) needed to configure a network interface. For more information, see the `ipadm(1M)` man page.
`/usr/sbin/omshell`	ISC DHCP only: Provides a way to query and change the ISC DHCP server's state by using the Object Management API (OMAPI). For more information, see the `omshell(1)` man page.
`/usr/sbin/pntadm`	Legacy Sun DHCP only: Used to make changes to the DHCP network tables that map client IDs to IP addresses and optionally associate configuration information with IP addresses. For more information, see the `pntadm(1M)` man page.
`/usr/sbin/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. For more information, see the `snoop(1M)` man page.

Files Used by the DHCP Service

The following table lists the files that are associated with DHCP.

TABLE 13–2 Files and Tables Used by DHCP Daemons and Commands

File or Table Name	Description
`dhcptab`	Legacy Sun DHCP only: 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 that you use for DHCP information. For more information, see the `dhcptab(4)` man page.
DHCP network table	Legacy Sun DHCP only: 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 that you use for DHCP information. For more information, see the `dhcp_network(4)` man page.
`/etc/dhcp/eventhook`	Legacy Sun DHCP only: A script or executable that the `dhcpagent` daemon can automatically run. For more information, see the `dhcpagent(1M)` man page.

`/etc/inet/dhcpd4.conf`
 	
`/etc/inet/dhcpd6.conf`
	ISC DHCP only: Contains configuration information for the ISC DHCP server, `dhcpd`. For more information, see the `dhcpd.conf(5)` man page.

`/etc/inet/dhcpsvc.conf`	Legacy Sun DHCP only: 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. For more information, see the `dhcpsvc.conf(4)` man page.

`/etc/dhcp/`*interface*`.dhc`
 	
`/etc/dhcp/`*interface*`.dh6`
	Contains the configuration parameters that are obtained from DHCP for the given network interface. For DHCPv4 the filename ends with `dhc`. For DHCPv6, the filename ends with `dh6`. 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.

`/etc/default/dhcpagent`	Sets parameter values for the `dhcpagent` client daemon. See the `/etc/default/dhcpagent` file or the `dhcpagent(1M)` man page for information about the parameters.

TABLE 13–2 Files and Tables Used by DHCP Daemons and Commands *(Continued)*

File or Table Name	Description

`/etc/dhcp/inittab`
 	
`/etc/dhcp/inittab6`
	Legacy Sun DHCP only: 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. The `/etc/dhcp/inittab6` is used by the DHCPv6 clients.
	
	On the client, the information in the `/etc/dhcp/inittab` file is used by the `dhcpinfo` command 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.

`/var/db/isc-dhcp/dhcp4.leases`
 	
`/var/db/isc-dhcp/dhcp4.leases-`
 	
`/var/db/isc-dhcp/dhcp6.leases`
 	
`/var/db/isc-dhcp/dhcp6.leases-`
	ISC DHCP only: Lists leases for DHCPv4 and DHCPv6 servers. Files with "-" at end of the file name are previous copies.

SMF Services Used by the DHCP Service

The following table lists the SMF services associated with DHCP.

TABLE 13–3 SMF Services Used by DHCP Daemons and Commands

SMF Service Name	Description
`svc:/network/dhcp-server:default`	Contains information for the legacy Sun DHCP service.

`svc:/network/dhcp/server:ipv4`
 	
`svc:/network/dhcp/server:ipv6`
	Contains information for the ISC DHCP service.

`svc:/network/dhcp/relay:ipv4`
 	
`svc:/network/dhcp/relay:ipv6`
	Contains information for the service that can relay DHCP or BOOTP requests to a remote ISC DHCP server.

`svc:/network/dns/client`	Contains information used to resolve DNS queries. During DHCP server configuration, this SMF service is consulted for information about the DNS domain and DNS server.
`svc:/system/name-service/switch`	Specifies the location of name service databases and the order in which to search name services for various kinds of information. This service provides accurate configuration information when you configure a DHCP service.

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. The IP Filter feature of Oracle Solaris provides a firewall.

◆ ◆ ◆ CHAPTER 14

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 15, "Configuring IPsec (Tasks)." For reference information, see Chapter 16, "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. 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 the following main components:

- **Security protocols** – The IP datagram protection mechanisms. The authentication header (AH) includes a hash of the IP packet 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 (SA)** – The cryptographic parameters and the IP security protocol as applied to a specific flow of network traffic. Each SA has a unique reference called the Security Parameters Index (SPI).
- **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.

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 17–1.

TABLE 14–1 IPsec Terms, Acronyms, and Uses

IPsec Term	Acronym	Definition
Security association	SA	The cryptographic parameters and the IP security protocol that are applied to a specific flow of network traffic. The SA is defined by a triplet: a security protocol, a unique security parameter index (SPI), and an IP destination.
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 14–1 IPsec Terms, Acronyms, and Uses *(Continued)*

IPsec Term	Acronym	Definition
Key exchange		The process of generating keys by using asymmetric cryptographic algorithms. The two main methods are RSA and Diffie-Hellman.
Diffie-Hellman	DH	A key exchange algorithm that allows key generation and key authentication. Often called *authenticated key exchange.*
RSA	RSA	A key exchange algorithm that allows 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 an IKE exchange.

IPsec Packet Flow

Figure 14–1 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 14–2 shows the IPsec inbound process.

FIGURE 14–1 IPsec Applied to Outbound Packet Process

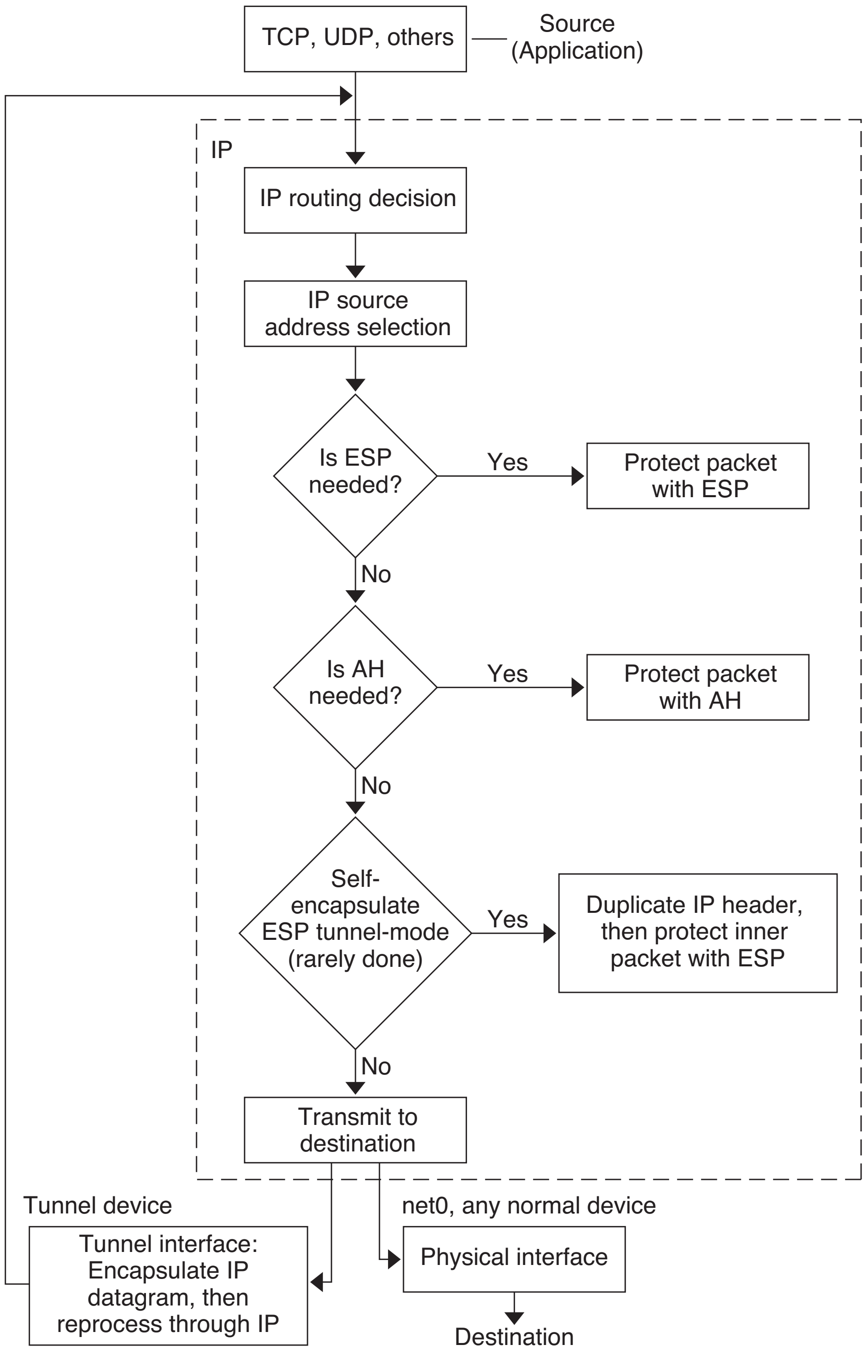
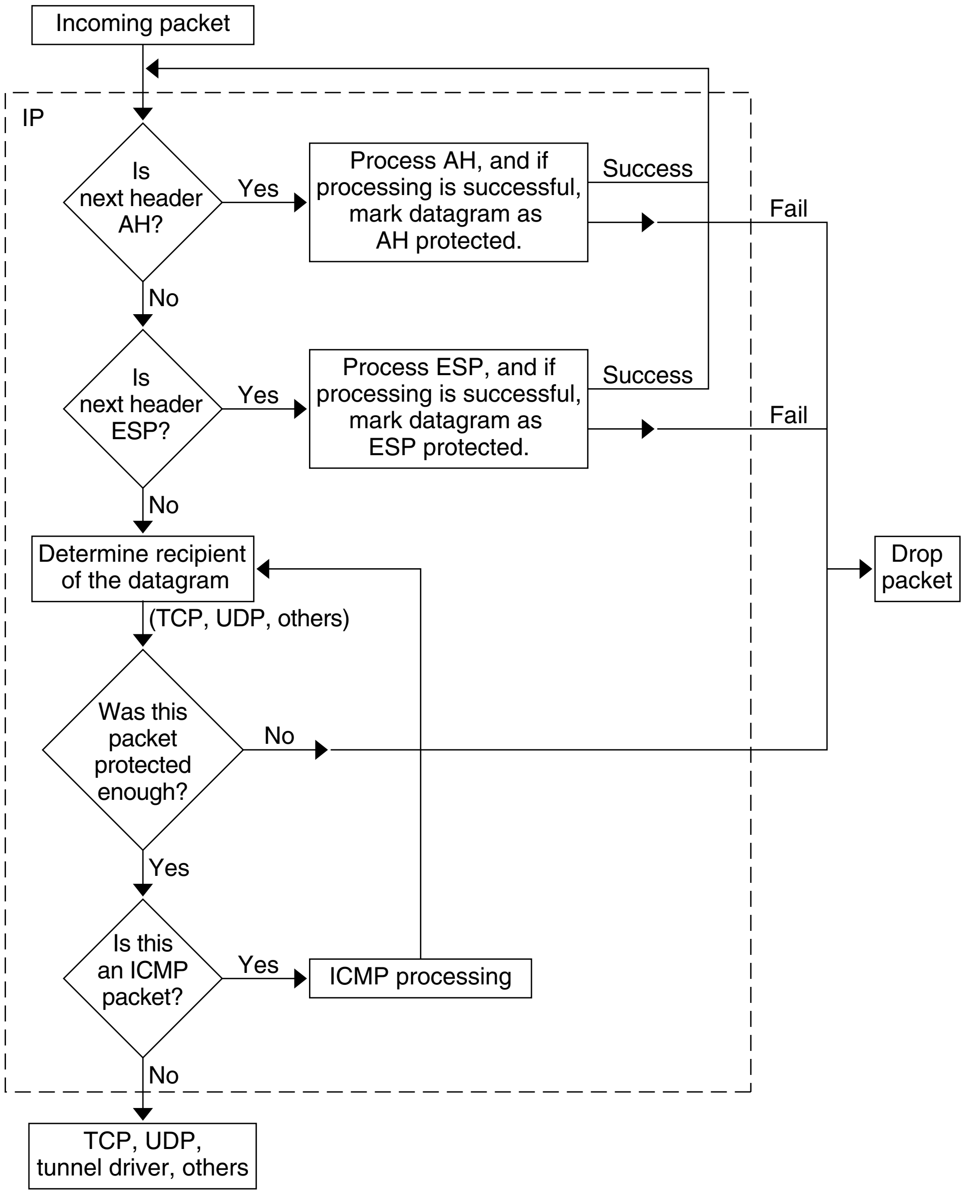


FIGURE 14–2 IPsec Applied to Inbound Packet Process


Incoming packet
IP
Is next header AH?
Yes
Process AH, and if processing is successful, mark datagram as AH protected.
Success
Fail
No
Is next header ESP?
Yes
Process ESP, and if processing is successful, mark datagram as ESP protected.
Success
Fail
No
Determine recipient of the datagram
Drop packet
(TCP, UDP, others)
Was this packet protected enough?
No
Yes
Is this an ICMP packet?
Yes
ICMP processing
No
TCP, UDP, tunnel driver, others

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 administrative interface, `PF_KEY` 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 246.
- 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, IKE is preferred.

The Service Management Facility (SMF) feature of Oracle Solaris 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 17, "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" on page 289.
- `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 SA Generation in IPsec" on page 247. For a detailed description of the `ipseckey` command options, see the `ipseckey(1M)` man page.

IPsec Protection Mechanisms

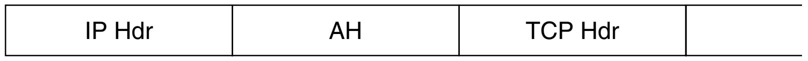
IPsec provides two security protocols for protecting data:

- Authentication Header (AH)
- Encapsulating Security Payload (ESP)

AH protects data with an authentication algorithm. An ESP protects data with an encryption algorithm. ESP can and should be used with an authentication mechanism. If you are not traversing a NAT, you can combine ESP with AH. Otherwise, you can use an authentication algorithm and an encryption mechanism with ESP. A combined mode algorithm, such as AES-GCM, provides encryption and authentication within a single algorithm.

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.

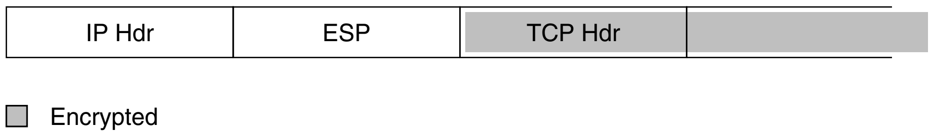


The transport header can be TCP, UDP, SCTP, or ICMP. If a tunnel is being used, the transport header can be another IP header.

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.



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 215.

Security Considerations When Using AH and ESP

The following table compares the protections that are provided by AH and ESP.

TABLE 14–2 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
ESP	Protects packet following the beginning of ESP in the datagram.	With encryption option, encrypts the IP payload. Ensures confidentiality	Eavesdropping
		With authentication option, provides the same payload 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 uses the Cryptographic Framework feature of Oracle Solaris to access the algorithms. The Cryptographic 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.

For more information, see the following:

- Chapter 11, “Cryptographic Framework (Overview),” in *Oracle Solaris Administration: Security Services*
- Chapter 8, “Introduction to the Oracle Solaris Cryptographic Framework,” in *Developer’s Guide to Oracle Solaris 11 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.

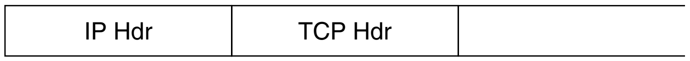
Therefore, in tunnel mode, IPsec policy can be specified for subnets of a LAN behind a router and for ports on those subnets. IPsec policy can also be specified for particular IP addresses, that is, hosts, on those subnets. The ports of those hosts can also have a specific IPsec policy. However, if a 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 229.

In Oracle Solaris, tunnel mode can be enforced only on an IP tunneling network interface. For information about tunneling interfaces, see Chapter 6, "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.

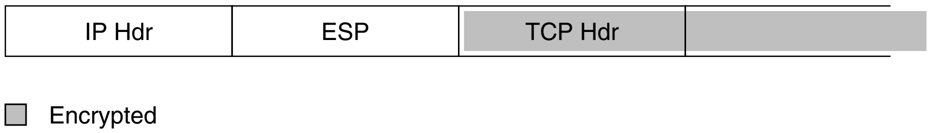
The following figure shows an IP header with an unprotected TCP packet.

FIGURE 14–3 Unprotected IP Packet Carrying TCP Information



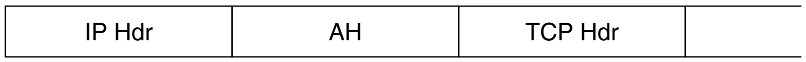
In transport mode, ESP protects the data as shown in the following figure. The shaded area shows the encrypted part of the packet.

FIGURE 14–4 Protected IP Packet Carrying TCP Information



In transport mode, AH protects the data as shown in the following figure.

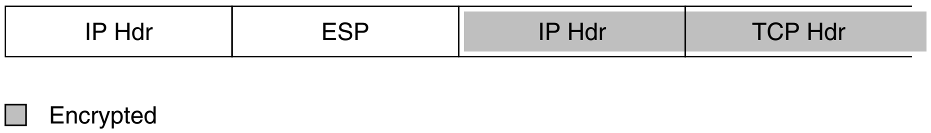
FIGURE 14–5 Packet Protected by an Authentication Header



AH protection, even in transport mode, covers most of the IP header.

In tunnel mode, the entire datagram is *inside* the protection of an IPsec header. The datagram in Figure 14–3 is protected in tunnel mode by an outer IPsec header, and in this case ESP, as is shown in the following figure.

FIGURE 14–6 IPsec Packet Protected in Tunnel Mode



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 227.
- For more information about tunnels, see the ipsecconf(1M) man page.
- For an example of tunnel configuration, see "How to Protect a VPN With IPsec in Tunnel Mode" on page 232.

Virtual Private Networks and IPsec

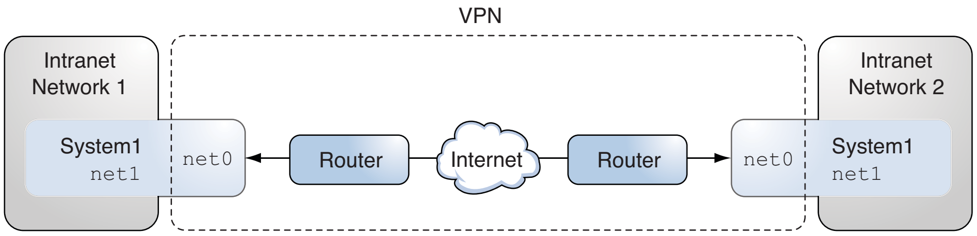
A configured tunnel is a point-to-point interface. The tunnel enables one IP packet to be encapsulated within another IP packet. A correctly configured tunnel requires both a tunnel source and a tunnel destination. For more information, see "How to Create and Configure an IP Tunnel" on page 123.

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 add IPsec protections to a VPN. IPsec secures the connection. For example, an organization that uses VPN technology to connect offices with separate networks can add IPsec to secure the traffic between the two offices.

The following figure illustrates how two offices form a VPN with IPsec deployed on their network systems.

FIGURE 14–7 Virtual Private Network



For a detailed example of the setup procedure, see "How to Protect a VPN With IPsec in Tunnel Mode" on page 232.

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 305.

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:

- 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 278.

IPsec and SCTP

Oracle Solaris 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 Oracle 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 per non-global zone.

For information about how to use IPsec with zones, see "Protecting Traffic With IPsec" on page 223. For information about zones, see Chapter 15, "Introduction to Oracle Solaris Zones," in *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

IPsec and Logical Domains

IPsec works with logical domains. The logical domain must be running a version of Oracle Solaris that includes IPsec, such as the Oracle 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 *Oracle VM Server for SPARC 2.1 Administration Guide* or *Oracle VM Server for SPARC 2.0 Administration Guide*.

IPsec Utilities and Files

Table 14–3 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 6, "Managing Services (Overview)," in *Oracle Solaris Administration: Common Tasks*.

- For instructions on implementing IPsec on your network, see "Protecting Traffic With IPsec" on page 223.
- For more details about IPsec utilities and files, see Chapter 16, "IP Security Architecture (Reference)."

TABLE 14–3 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.	`ipsecalgs(1M)`
`svc:/network/ipsec/manual-key`	The SMF service that manages manually keyed IPsec SAs.	`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 by using IKE.	`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 14–3 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	Contains manually keyed 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 252. 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 15

Configuring IPsec (Tasks)

This chapter provides procedures for implementing IPsec on your network. The procedures are described in the following sections:

- "Protecting Traffic With IPsec" on page 223
- "Protecting a VPN With IPsec" on page 229
- "Managing IPsec and IKE" on page 236

For overview information about IPsec, see Chapter 14, "IP Security Architecture (Overview)." For reference information about IPsec, see Chapter 16, "IP Security Architecture (Reference)."

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" on page 229. For additional procedures to manage IPsec and to use SMF commands with IPsec and IKE, see "Managing IPsec and IKE" on page 236.

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. For an exclusive-IP zone, you configure IPsec policy in the non-global zone.
- **IPsec and RBAC –** To use roles to administer IPsec, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *Oracle Solaris Administration: Security Services*. For an example, see "How to Configure a Role for Network Security" on page 238.
- **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 219.

- **IPsec and Trusted Extensions labels –** On systems that are configured with the Trusted Extensions feature of Oracle Solaris, labels can be added to IPsec packets. For more information, see "Administration of Labeled IPsec" in *Trusted Extensions Configuration and Administration*.
- **IPv4 and IPv6 addresses –** The IPsec example in this guide use IPv4 addresses. Oracle Solaris supports IPv6 addresses as well. To configure IPsec for an IPv6 network, substitute IPv6 addresses in the examples. When protecting tunnels with IPsec, you can mix IPv4 and IPv6 addresses for the inner and outer addresses. Such a configuration enables you to tunnel IPv6 over an IPv4 network, for example.

The following task map points to procedures that set up IPsec between one or more systems. The `ipsecconf`(1M), `ipseckey`(1M), and `ipadm`(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 224
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 227
Display IPsec policies.	Displays the IPsec policies that are currently being enforced, in the order of enforcement.	"How to Display IPsec Policies" on page 228
Use IKE to automatically create keying material for IPsec SAs.	Provides the raw data for security associations.	"Configuring IKE (Task Map)" on page 257
Set up a secure virtual private network (VPN).	Sets up IPsec between two systems across the Internet.	"Protecting a VPN With IPsec" on page 229

▼ 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 an IP address. This can be an IPv4 address, an IPv6 address, or both.
- Each system requires ESP encryption with the AES algorithm, which requires a key of 128 bits, and ESP authentication with a SHA-2 message digest, which requires a key of 512 bits.
- 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 *Trusted Extensions Configuration and Administration*.

Before You Begin IPsec policy can be configured in the global zone or in an exclusive-IP stack zone. The policy for a shared-IP stack zone must be configured in the global 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 *Oracle Solaris Administration: Security Services*. If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 15–1.

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
```

b. **On a system that is named `enigma`, type the following in the `hosts` file:**

```
# Secure communication with partym
192.168.13.213 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 sha512 sa shared}
```

b. **On the `partym` system, add the identical policy:**

```
{laddr partym raddr enigma} ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
```

For the syntax of IPsec policy entries, see the `ipsecconf(1M)` man page.

5 **On each system, configure IKE to add a pair of IPsec SAs between the two systems.**

Configure IKE by following one of the configuration procedures in "Configuring IKE (Task Map)" on page 257. For the syntax of the IKE configuration file, see the `ike.config(4)` man page.

Note – If you must generate and maintain your keys manually, see "How to Manually Create IPsec Keys" on page 236.

---

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 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, complete "How to Manually Create IPsec Keys" on page 236 to activate the keys.

9 **Verify that packets are being protected.**

For the procedure, see "How to Verify That Packets Are Protected With IPsec" on page 241.

Example 15–1 Adding IPsec Policy When Using an `ssh` Connection

In this example, the administrator in the `root` role 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 6 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 7 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-2 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 224 so that the following conditions are in effect:

- Communication between the two systems is protected by IPsec.
- Keying material is being generated by IKE.
- You have verified that packets are being protected.

1 Become an administrator.

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*. If you log in remotely, use the ssh command for a secure remote login. For an example, see Example 15–1.

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-2.
# Use a unique SA for outbound traffic from the port
{} ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
```

This configuration allows only secure traffic to access the system, with the bypass exceptions that are described in Step 2.

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.**

Restart the `ike` service.

```
# svcadm restart svc:/network/ipsec/ike
```

If you manually configured the keys, follow the instructions in "How to Manually Create IPsec Keys" on page 236.

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.**

Add the following lines to a remote system's `/etc/inet/ipsecinit.conf` file:

```
# Communicate with web server about nonweb stuff
#
{laddr webserver} ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
```

Verify the syntax then refresh the IPsec policy to activate it.

```
remote-system # ipsecconf -c -f /etc/inet/ipsecinit.conf
remote-system # svcadm refresh svc:/network/ipsec/policy:default
```

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.**

To create a discrete role for network security and assign that role to a user, see "How to Configure a Role for Network Security" on page 238.

2 **Display IPsec policies.**

- **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.

- **Display the IPsec policy entries in the order in which a match occurs.**

  ```
  $ ipsecconf -l -n
  ```

- **Display the IPsec policy entries, including per-tunnel entries, in the order in which a match occurs.**

  ```
  $ ipsecconf -L -n
  ```

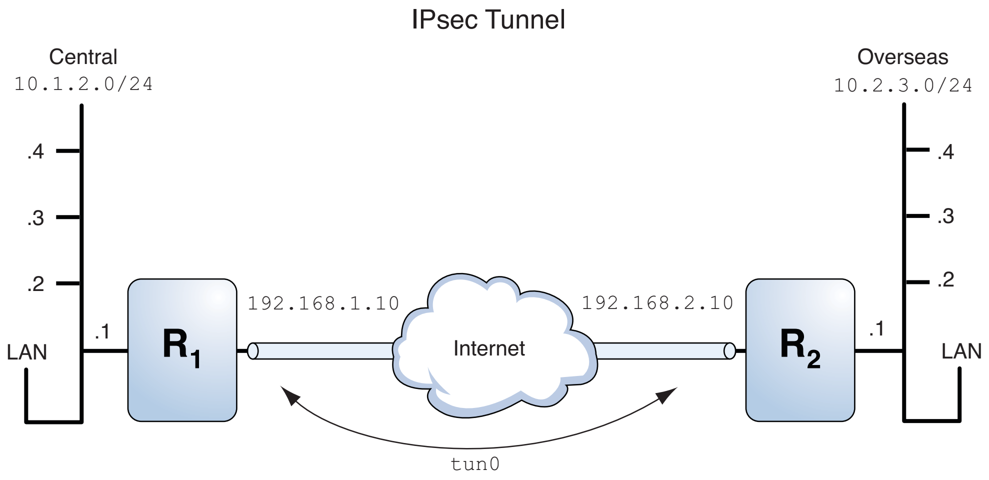
Protecting a VPN With IPsec

Oracle Solaris can configure a VPN that is protected by IPsec. Tunnels can be created in *tunnel mode* or in *transport mode*. For a discussion, see "Transport and Tunnel Modes in IPsec" on page 215. The examples and procedures in this section use IPv4 addresses, but the examples and procedures apply to IPv6 VPNs as well. For a short discussion, see "Protecting Traffic With IPsec" on page 223.

For examples of IPsec policies for tunnels in tunnel mode, see "Examples of Protecting a VPN With IPsec by Using Tunnel Mode" on page 229.

Examples of Protecting a VPN With IPsec by Using Tunnel Mode

FIGURE 15–1 Tunnel Protected by IPsec



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 15–2 Creating a Tunnel That All Subnets Can Use

In this example, all traffic from the local LANs of the Central LAN in Figure 15–1 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 sha512 sa shared}
```

EXAMPLE 15–3 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}
```

EXAMPLE 15–3 Creating a Tunnel That Connects Two Subnets Only *(Continued)*

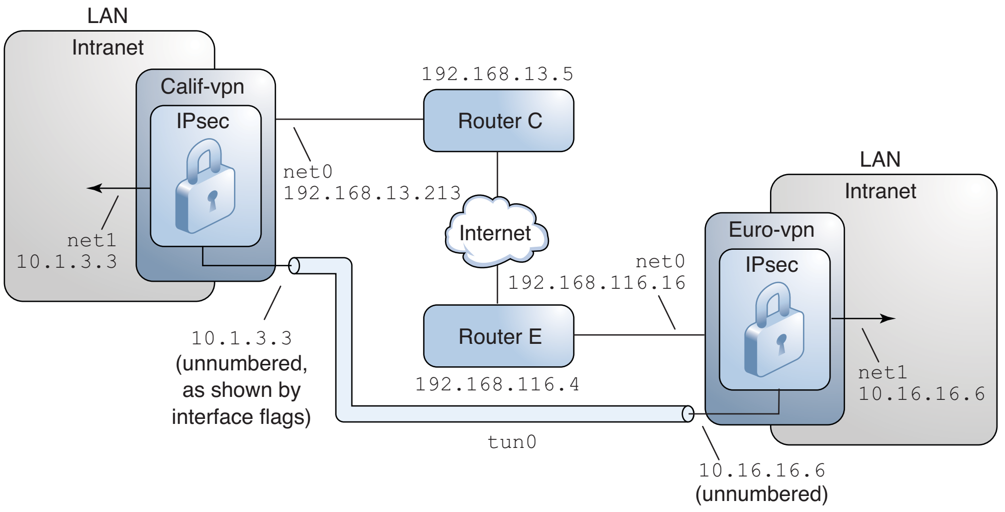
```
ipsec {encr_algs aes encr_auth_algs sha512 shared}
```

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 15–2.

- Each system is using an IPv4 address space.
- Each system has two interfaces. The `net0` interface connects to the Internet. In this example, Internet IP addresses begin with `192.168`. The `net1` 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-2 algorithm. In this example, the SHA-2 algorithm requires a 512-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 15–2 Sample VPN Between Offices Connected Across the Internet



As the preceding illustration shows, the procedures use the following configuration parameters.

Parameter	Europe	California
System name	`euro-vpn`	`calif-vpn`
System intranet interface	`net1`	`net1`
System intranet address, also the *-point* address in Step 7	`10.16.16.6`	`10.1.3.3`
System intranet address object	`net1/inside`	`net1/inside`
System Internet interface	`net0`	`net0`
System Internet address, also the *tsrc* address in Step 7	`192.168.116.16`	`192.168.13.213`
Name of Internet router	`router-E`	`router-C`
Address of Internet router	`192.168.116.4`	`192.168.13.5`
Tunnel name	`tun0`	`tun0`
Tunnel name address object	`tun0/v4tunaddr`	`tun0/v4tunaddr`

For information about tunnel names, see "Tunnel Configuration and Administration With the `dladm` Command" on page 121. For information about address objects, see "How to Configure an IP Interface" on page 47 and the `ipadm(1M)` man page.

▼ How to Protect a VPN With IPsec 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 224. The setup is described in "Description of the Network Topology for the IPsec Tasks to Protect a VPN" on page 231.

For a fuller description of the reasons for running particular commands, see the corresponding steps in "How to Secure Traffic Between Two Systems With IPsec" on page 224.

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 *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 *Oracle Solaris Administration: Security Services*. If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 15–1.

2 **Control the flow of packets before configuring IPsec.**

a. **Disable IP forwarding and IP dynamic routing.**

```
# routeadm -d ipv4-routing
# ipadm set-prop -p forwarding=off ipv4
# 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.

b. **Turn on IP strict multihoming.**

```
# ipadm set-prop -p hostmodel=strong ipv4
```

Turning on IP strict multihoming requires that packets for one of the system's destination addresses arrive at the correct destination address.

When the `hostmodel` parameter is set to `strong`, 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.

c. **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 IPsec policy.**

Edit the `/etc/inet/ipsecinit.conf` file to add the IPsec policy for the VPN. For additional examples, see "Examples of Protecting a VPN With IPsec by Using Tunnel Mode" on page 229.

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 `euro-vpn` 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-2.
{tunnel tun0 negotiate tunnel}
 ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
```

b. **On the `calif-vpn` 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-2.
{tunnel tun0 negotiate tunnel}
 ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
```

4 **On each system, configure IKE to add a pair of IPsec SAs between the two systems.**

Configure IKE by following one of the configuration procedures in "Configuring IKE (Task Map)" on page 257. For the syntax of the IKE configuration file, see the `ike.config`(4) man page.

Note – If you must generate and maintain your keys manually, see "How to Manually Create IPsec Keys" on page 236.

---

5 **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.

6 **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
```

7 **Create and configure the tunnel, *tunnel-name*.**

The following commands configure the internal and external interfaces, create the `tun0` tunnel, and assign IP addresses to the tunnel.

a. **On the `calif-vpn` system, create the tunnel and configure it.**

If the interface `net1` does not already exist, the first command creates it.

```
# ipadm create-addr -T static -a local=10.1.3.3 net1/inside
# dladm create-iptun -T ipv4 -a local=10.1.3.3,remote=10.16.16.6 tun0
# ipadm create-addr -T static \
-a local=192.168.13.213,remote=192.168.116.16 tun0/v4tunaddr
```

b. **On the `euro-vpn` system, create the tunnel and configure it.**

```
# ipadm create-addr -T static -a local=10.16.16.6 net1/inside
# dladm create-iptun -T ipv4 -a local=10.16.16.6,remote=10.1.3.3 tun0
# ipadm create-addr -T static \
-a local=192.168.116.16,remote=192.168.13.213 tun0/v4tunaddr
```

Note – The `-T` option to the `ipadm` command specifies the type of address to create. The `-T` option to the `dladm` command specifies the tunnel.

For information about these commands, see the `dladm`(1M) and `ipadm`(1M) man pages, and "How to Configure an IP Interface" on page 47. For information about customized names, see "Network Devices and Datalink Names" in *Oracle Solaris Administration: Network Interfaces and Network Virtualization*.

8 **On each system, configure forwarding.**

```
# ipadm set-ifprop -m ipv4 -p forwarding=on net1
# ipadm set-ifprop -m ipv4 -p forwarding=off net0
```

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 `net1` interface is *inside* the intranet, IP forwarding must be turned on for `net1`. Because `tun0` connects the two systems through the Internet, IP forwarding must remain on for `tun0`. The `net0` 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.

9 **On each system, prevent the advertising of the private interface.**

```
# ipadm set-addrprop -p private=on net0
```

Even if `net0` has IP forwarding turned off, a routing protocol implementation might still advertise the interface. For example, the `in.routed` protocol might still advertise that `net0` is available to forward packets to its peers inside the intranet. By setting the interface's *private* flag, these advertisements are prevented.

10 **Restart the network services.**

```
# svcadm restart svc:/network/initial:default
```

11 **Manually add a default route over the `net0` interface.**

The default route must be a router with direct access to the Internet.

a. **On the `calif-vpn` system, add the following route:**

```
# route -p add net default 192.168.13.5
```

b. **On the `euro-vpn` system, add the following route:**

```
# route -p add net default  192.168.116.4
```

Even though the `net0` interface is not part of the intranet, `net0` does need to reach across the Internet to its peer system. To find its peer, `net0` 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.

Managing IPsec and IKE

The following task map points to tasks that you might use when managing IPsec.

Task	Description	For Instructions
Create or replace security associations manually.	Provides the raw data for security associations: ■ IPsec algorithm name and keying material ■ The security parameter index (SPI) ■ IP source and destination addresses, and other parameters	"How to Manually Create IPsec Keys" on page 236
Create a Network Security role.	Creates a role that can set up a secure network, but has fewer powers than the `root` role.	"How to Configure a Role for Network Security" on page 238
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 IPsec and IKE Services" on page 240
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 241

▼ How to Manually Create IPsec Keys

The following procedure provides the keying material for Step 5 in "How to Secure Traffic Between Two Systems With IPsec" on page 224. 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 SHA-2 algorithm for AH. Each number must be 512 characters long. One number is for `dst enigma`. One number is for `dst partym`.
- Two hexadecimal random numbers for the 3DES algorithm for ESP. Each number must be 168 characters long. One number is for `dst enigma`. One number is for `dst partym`.

b. Generate the required keys.

- If you have a random number generator at your site, use the generator.
- Use the `pktool` command, as shown in "How to Generate a Symmetric Key by Using the pktool Command" in *Oracle Solaris Administration: Security Services* and the IPsec example in that section.

2 In the `root` role on each system, add the keys to the manual keys file for IPsec.

a. 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 sha512  \
   encrkey  d41fb74470271826a8e7a80d343cc5aa... \
   authkey  e896f8df7f78d6cab36c94ccf293f031...
#
# for inbound packets
add esp spi 0x122a43e4 \
   src 192.168.13.213 dst 192.168.116.16 \
   encr_alg 3des \
   auth_alg sha512  \
   encrkey dd325c5c137fb4739a55c9b3a1747baa... \
   authkey ad9ced7ad5f255c9a8605fba5eb4d2fd...
```

b. Protect the file with read-only permissions.

```
# chmod 400 /etc/inet/secret/ipseckeys
```

c. **Verify the syntax of the file.**

```
# ipseckey -c -f /etc/inet/secret/ipseckeys
```

Note – The keying material on the two systems *must* be identical.

---

3 **Activate the keys for IPsec.**

- **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 ipsec/manual-key
  ```

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy.

▼ How to Configure a Role for Network Security

If you are using the role-based access control (RBAC) feature of Oracle Solaris to administer your systems, you use this procedure to provide a network management role or network security role.

1 **List the available network-related rights profiles.**

```
% getent prof_attr | grep Network | more
Console User:RO::Manage System as the Console User...
Network Management:RO::Manage the host and network configuration...
Network Autoconf Admin:RO::Manage Network Auto-Magic configuration via nwamd...
Network Autoconf User:RO::Network Auto-Magic User...
Network ILB:RO::Manage ILB configuration via ilbadm...
Network LLDP:RO::Manage LLDP agents via lldpadm...
Network VRRP:RO::Manage VRRP instances...
Network Observability:RO::Allow access to observability devices...
Network Security:RO::Manage network and host security...:profiles=Network Wifi
Security,Network Link Security,Network IPsec Management...
Network Wifi Management:RO::Manage wifi network configuration...
Network Wifi Security:RO::Manage wifi network security...
Network Link Security:RO::Manage network link security...
Network IPsec Management:RO::Manage IPsec and IKE...
System Administrator:RO::Can perform most non-security administrative tasks:profiles=...Network Management...
Information Security:RO::Maintains MAC and DAC security policies:profiles=...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 **List the commands in the Network Management rights profile.**

```
% getent exec_attr | grep "Network Management"
...
Network Management:solaris:cmd:::/sbin/dlstat:euid=dladm;egid=sys
...
Network Management:solaris:cmd:::/usr/sbin/snoop:privs=net_observability
Network Management:solaris:cmd:::/usr/sbin/spray:euid=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`, `ipseckey`, and `snoop` commands, among others, with appropriate privilege.

To create the role, assign the role to a user, and register the changes with the naming service, see "Initially Configuring RBAC (Task Map)" in *Oracle Solaris Administration: Security Services*.

Example 15–4 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 IPsec and IKE 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
  # svccfg -s policy listprop config/config_file
  config/config_file  astring  /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, restart the `ike` service.**

  ```
  # svcadm restart svc:/network/ipsec/ike:default
  ```

- **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 = astring: "modkeys"
  # svccfg -s ike listprop config/admin_privilege
  config/admin_privilege  astring  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:default
  ```

- **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
  # svccfg -s manual-key listprop config/config_file
  config/config_file  astring  /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.

▼ 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 in the `root` role to create the `snoop` output. You must have access to both systems to test the connection.

1 **On one system, such as `partym`, assume the `root` role.**

```
% 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 -d net0 -v enigma
Using device /dev/bge (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, assume the `root` role 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 -----
...
```

CHAPTER 16

IP Security Architecture (Reference)

This chapter contains the following reference information:

For instructions on how to implement IPsec on your network, see Chapter 15, "Configuring IPsec (Tasks)." For an overview of IPsec, see Chapter 14, "IP Security Architecture (Overview)."

IPsec Services

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" on page 289.

For information about SMF, see Chapter 6, "Managing Services (Overview)," in *Oracle Solaris Administration: Common Tasks*. 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 IPsec policy options, see the ipsecconf(1M) man page.

You must be in the root 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 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" on page 223.

ipsecinit.conf File

To enable the IPsec security policy when you start Oracle Solaris, 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 the policy is configured, you can refresh the policy with the svcadm refresh ipsec/policy command.

Sample ipsecinit.conf File

The Oracle 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:

```
...
# In the following simple example, outbound network traffic between the local
# host and a remote host will be encrypted. Inbound network traffic between
# these addresses is required to be encrypted as well.
#
# This example assumes that 10.0.0.1 is the IPv4 address of this host (laddr)
# and 10.0.0.2 is the IPv4 address of the remote host (raddr).
```

```
#

{laddr 10.0.0.1 raddr 10.0.0.2} ipsec
    {encr_algs aes encr_auth_algs sha256 sa shared}

# The policy syntax supports IPv4 and IPv6 addresses as well as symbolic names.
# Refer to the ipsecconf(1M) man page for warnings on using symbolic names and
# many more examples, configuration options and supported algorithms.
#
# This example assumes that 10.0.0.1 is the IPv4 address of this host (laddr)
# and 10.0.0.2 is the IPv4 address of the remote host (raddr).
#
# The remote host will also need an IPsec (and IKE) configuration that mirrors
# this one.
#
# The following line will allow ssh(1) traffic to pass without IPsec protection:

{lport 22 dir both} bypass {}

#
# {laddr 10.0.0.1 dir in} drop {}
#
# Uncommenting the above line will drop all network traffic to this host unless
# it matches the rules above. Leaving this rule commented out will allow
# network packets that does not match the above rules to pass up the IP
# network stack. ,,,
```

Security Considerations for `ipsecinit.conf` and `ipsecconf`

IPsec policy cannot be changed for established connections. 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. If you are in doubt, restart the connection.

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 `ssh`, or a console or other hard-connected TTY for the safest mode of operation.

ipsecalgs Command

The Cryptographic Framework feature of Oracle Solaris 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 about the Cryptographic Framework, see Chapter 11, "Cryptographic Framework (Overview)," in *Oracle Solaris Administration: 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 the root 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 SA Generation in IPsec

The IKE protocol provides automatic key management for IPv4 and IPv6 addresses. See Chapter 18, "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 Keys" on page 236.

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 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.

Note – Use IKE, not manual keying with `ipseckey`, if possible.

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 `ssh`, or 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 241.

Third-party network analyzers are also available, such as the free open-source software Wireshark (`http://www.wireshark.org/about.html`), which is bundled with this release.

CHAPTER 17

Internet Key Exchange (Overview)

Internet Key Exchange (IKE) automates key management for IPsec. Oracle Solaris implements IKEv1. This chapter contains the following information about IKE:

For instructions on implementing IKE, see Chapter 18, "Configuring IKE (Tasks)." For reference information, see Chapter 19, "Internet Key Exchange (Reference)." For information about IPsec, see Chapter 14, "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. Oracle Solaris uses Internet Key Exchange version 1 (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.

IKE can take advantage of available hardware acceleration and hardware storage. Hardware accelerators permit intensive key operations to be handled off the system. Key storage on hardware provides an additional layer of protection.

IKE Key Negotiation

The IKE daemon, `in.iked`, negotiates and authenticates keying material for IPsec SAs in a secure manner. The daemon uses random seeds for keys from internal functions provided by the OS. 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.

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 17–1 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 the RSA and the Diffie-Hellman protocols.
Diffie-Hellman algorithm	DH	A key exchange algorithm that provides key generation and key authentication. Often called *authenticated key exchange*.
RSA algorithm	RSA	A key exchange algorithm that provides 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. 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 group		A method for establishing keys for Phase 2 in a secure manner. 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.

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 Key Authentication

Preshared keys are used to authenticate two peer systems. The preshared key is a hexadecimal number or ASCII string that is created by an administrator on one system. The key is then shared with administrators of the peer system in a secure way. If the preshared key is intercepted by an adversary, that adversary might be able to impersonate one of the peer systems.

The preshared key on the peers that use this authentication method must be identical. The keys are tied to a particular IP address or range of addresses. The keys are placed in the `/etc/inet/secret/ike.preshared` file on each system. 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 algorithm (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 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 6, "Managing Services (Overview)," in *Oracle Solaris Administration: Common Tasks*.

TABLE 17–2 IKE Configuration Files, Key Storage Locations, Commands, and Services

File, Location, Command, or Service	Description	Man Page
`svc:/network/ipsec/ike`	The SMF service that manages IKE.	`smf`(5)
`/usr/lib/inet/in.iked`	Internet Key Exchange (IKE) daemon. Activates automated key management when the `ike` service is enabled.	`in.iked`(1M)
`/usr/sbin/ikeadm`	IKE administration command for viewing and temporarily modifying the IKE policy. Enables you to view IKE administrative objects, such as Phase 1 algorithms and available Diffie-Hellman groups.	`ikeadm`(1M)
`/usr/sbin/ikecert`	Certificate database management command for manipulating local databases that hold public key certificates. The databases can also be stored on attached hardware.	`ikecert`(1M)
`/etc/inet/ike/config`	Default configuration file for the IKE policy. 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`	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`	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 6000 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. The Sun Crypto Accelerator 6000 board is a FIPS 140-2 certified device at Level 3.	`ikecert`(1M)

CHAPTER 18

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 17, "Internet Key Exchange (Overview)." For reference information about IKE, see Chapter 19, "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 aes }
```

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 3des }
```

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 sha256 encr_alg aes }
```

See Also For tasks to configure IKE rules that require these values, see "Configuring IKE (Task Map)" on page 257.

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 IKE 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 257
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 262
Cross a NAT boundary.	Configures IPsec and IKE to communicate with a mobile system	"Configuring IKE for Mobile Systems (Task Map)" on page 278
Configure IKE to use a hardware keystore to generate a certificate pair.	Enables a Sun Crypto Accelerator 6000 board to accelerate IKE operations and to store public key certificates.	"Configuring IKE to Find Attached Hardware" on page 286

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 configuration file and one key to be shared.	"How to Configure IKE With Preshared Keys" on page 258
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 Update IKE for a New Peer System" on page 260

Configuring IKE With Preshared Keys

Preshared keys is the simplest authentication method for IKE. If you are configuring peer system to use IKE, and you are the administrator of these systems, using preshared keys is a good choice. However, unlike public key certificates, preshared keys are tied to IP addresses. You can associate preshared keys with specific IP addresses 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 specified range of IP addresses.

▼ 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 ASCII format.

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 *Trusted Extensions Configuration and Administration*.

---

1. **Become an administrator.**

   For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*. If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 15–1.

2. **On each system, create an `/etc/inet/ike/config` file.**

   You can use the `/etc/inet/ike/config.sample` as a template.

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 configuration examples work with the `ipsecinit.conf` examples in "How to Secure Traffic Between Two Systems With IPsec" on page 224.

   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
   ```

```
#
## Defaults that individual rules can override.
p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg 3des }
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 sha256 encr_alg aes }
  p2_pfs 5
}
```

b. **Modify the `/etc/inet/ike/config` file on the `partym` system:**

```
### ike/config file on partym, 192.168.13.213
## Global Parameters
#
p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg 3des }
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 sha256 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 **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
    # The preshared key can also be represented in hex
# as in 0xf47cb0f432e14480951095f82b
# key "This is an ASCII Cqret phrAz, use str0ng p@ssword tekniques"
}
```

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
    # The preshared key can also be represented in hex
# as in 0xf47cb0f432e14480951095f82b
    key "This is an ASCII Cqret phrAz, use str0ng p@ssword tekniques"
    }
```

6 Enable the IKE service.

```
# svcadm enable ipsec/ike
```

Example 18–1 Refreshing an IKE Preshared Key

When IKE administrators want to refresh the preshared key, they edit the files on the peer systems and restart the `in.iked` daemon.

First, the administrator adds a preshared key entry, valid for any host on the `192.168.13.0/24` 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 "LOooong key Th@t m^st Be Ch*angEd \"reguLarLy)"
    }
```

Then, the administrator restarts the IKE service on every system.

```
# svcadm enable ipsec/ike
```

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy.

▼ How to Update IKE for a New Peer System

If you add IPsec policy entries to a working configuration between the same peers, you need to refresh the IPsec policy service. You do not need to reconfigure or restart IKE.

If you add a new peer to the IPsec policy, in addition to the IPsec changes, you must modify the IKE configuration.

Before You Begin You have updated the `ipsecinit.conf` file and refreshed IPsec policy for the peer systems.

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*. If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 15–1.

2 **Create a rule for IKE to manage the keys for the new system that is using IPsec.**

a. **For example, 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 sha256 encr_alg aes}
 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 sha256 encr_alg aes}
 p2_pfs 5
}
```

3 **Create an IKE preshared key for the peer systems.**

a. **On the `enigma` system, add the following information to the `/etc/inet/secret/ike.preshared` file:**

```
# 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
  key "Twas brillig and the slivey toves did *s0mEtHiNg* be CareFULL hEEEr"
}
```

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
        key "Twas brillig and the slivey toves did *s0mEtHiNg* be CareFULL hEEEr"
      }
```

4 **On each system, refresh the `ike` service.**

```
# svcadm refresh ike
```

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy.

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.

A public certificate must be unique, so the creator of a public key certificate generates an arbitrary, unique name for the certificate. Typically, an X.509 distinguished name is used. An alternate name can also be used for identification. The format of these names is *tag=value*. The values are arbitrary, though the format of the value must correspond to its tag type. For example, the format of the `email` tag is *name@domain.suffix*.

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 peer system	"How to Configure IKE With Self-Signed Public Key Certificates" on page 263
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 268
Configure public key certificates in local hardware.	Involves one of: ■ Generating a self-signed certificate in the local hardware, then adding the public key from a remote system to the hardware. ■ Generating a certificate request in the local hardware, then adding the public key certificates from the CA to the hardware.	"How to Generate and Store Public Key Certificates in Hardware" on page 273

Task	Description	For Instructions
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 276

Note – To label packets and IKE negotiations on a Trusted Extensions system, follow the procedures in "Configuring Labeled IPsec (Task Map)" in *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 generated and stored in attached hardware. For the procedure, see "Configuring IKE to Find Attached Hardware" on page 286.

▼ How to Configure IKE With Self-Signed Public Key Certificates

In this procedure, you create a certificate pair. The private key is stored on disk in the local certificate database and can be referenced by using the `certlocal` subcommand. The public portion of the certificate pair is stored in the public certificate database. It can be referenced by using the `certdb` subcommand. You exchange the public portion with a peer system. The combination of the two certificates is used to authenticate the IKE transmissions.

Self-signed certificates require less overhead than public certificates from a CA, but do not scale very easily. Unlike certificates that are issued by a CA, self-signed certificates must be verified out of band.

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*. If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 15–1.

2 Create a self-signed certificate in the `ike.privatekeys` database.

```
# ikecert certlocal -ks -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.
`-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 2048 -t rsa-sha1 \
-D "O=exampleco, OU=IT, C=US, CN=partym" \
-A IP=192.168.13.213
Creating private key.
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
MIIC1TCCAb2gAwIBAgIEfdZgKjANBgkqhkiG9w0BAQUFADAaMRgwFgYDVQQDEw9T
a...+
zBGi4QkNdI3f
-----END X509 CERTIFICATE-----
```

Note – The values of the `-D` and `-A` options are arbitrary. The values are used to identify the certificate only. They are not used to identify a system, such as 192.168.13.213. In fact, because these values are idiosyncratic, you must verify out of band that the correct certificate is installed on the peer systems.

---

b. The command on the `enigma` system would appear similar to the following:

```
# ikecert certlocal -ks -m 2048 -t rsa-sha1 \
-D "O=exampleco, OU=IT, C=US, CN=enigma" \
-A IP=192.168.116.16
Creating private key.
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
```

```
MIIC1TCCAb2gAwIBAgIEBl5JnjANBgkqhkiG9w0BAQUFADAaMRgwFgYDVQQDEw9T
...
y85m6LHJYtC6
-----END X509 CERTIFICATE-----
```

3 **Save the certificate and send it to the remote system.**

The output is an encoded version of the public portion of the certificate. You can safely paste this certificate into an email. The receiving party must verify out of band that they installed the correct certificate, as shown in Step b.

a. **For example, you would send the public portion of the `partym` certificate to the `enigma` administrator.**

```
To: admin@ja.enigmaexample.com
From: admin@us.partyexample.com
Message: -----BEGIN X509 CERTIFICATE-----
MIIC1TCCAb2gAwIBAgIEfdZgKjANBgkqhkiG9w0BAQUFADAaMRgwFgYDVQQDEw9T
a...+
zBGi4QkNdI3f
-----END X509 CERTIFICATE------
```

b. **The `enigma` administrator would send you the public portion of the `enigma` certificate.**

```
To: admin@us.partyexample.com
From: admin@ja.enigmaexample.com
Message: ----BEGIN X509 CERTIFICATE-----
MIIC1TCCAb2gAwIBAgIEBl5JnjANBgkqhkiG9w0BAQUFADAaMRgwFgYDVQQDEw9T
...
y85m6LHJYtC6
-----END X509 CERTIFICATE-----
```

4 **On each system, add the certificate that you received to the public key database.**

a. **Save the administrator's email to a file that is readable by `root`.**

b. **Redirect the file to the `ikecert` command.**

```
# ikecert certdb -a < /tmp/certificate.eml
```

The command imports the text between the `BEGIN` and `END` tags.

5 **Verify with the other administrator that the certificate is from that administrator.**

For example, you can telephone the other administrator to verify that the hash of their public certificate, which you have, matches the hash of their private certificate, which only they have.

a. **List the stored certificate on `partym`.**

In the following example, Note 1 indicates the distinguished name (DN) of the certificate in slot 0. The private certificate in slot 0 has the same hash, so these certificates are the same certificate pair. For the public certificates to work, you must have a matching pair. The `certdb` subcommand lists the public portion, while the `certlocal` subcommand lists the private portion.

```
partym # ikecert certdb -l

Certificate Slot Name: 0   Key Type: rsa
    (Private key in certlocal slot 0)
    Subject Name: <O=exampleco, OU=IT, C=US, CN=partym>     Note 1
    Key Size: 2048
    Public key hash: 80829EC52FC5BA910F4764076C20FDCF

Certificate Slot Name: 1   Key Type: rsa
    (Private key in certlocal slot 1)
    Subject Name: <O=exampleco, OU=IT, C=US, CN=Ada>
    Key Size: 2048
    Public key hash: FEA65C5387BBF3B2C8F16C019FEBC388
partym # ikecert certlocal -l
Local ID Slot Name: 0   Key Type: rsa
    Key Size: 2048
    Public key hash: 80829EC52FC5BA910F4764076C20FDCF     Note 3

Local ID Slot Name: 1   Key Type: rsa-sha1
        Key Size: 2048
        Public key hash: FEA65C5387BBF3B2C8F16C019FEBC388

Local ID Slot Name: 2   Key Type: rsa
    Key Size: 2048
    Public key hash: 2239A6A127F88EE0CB40F7C24A65B818
```

This check has verified that the `partym` system has a valid certificate pair.

b. Verify that the `enigma` system has `partym`'s public certificate.

You can read the public key hash over the telephone.

Compare the hashes from Note 3 on `partym` in the preceding step with Note 4 on `enigma`.

```
enigma # ikecert certdb -l

Certificate Slot Name: 0   Key Type: rsa
    (Private key in certlocal slot 0)
    Subject Name: <O=exampleco, OU=IT, C=US, CN=Ada>
    Key Size: 2048
    Public key hash: 2239A6A127F88EE0CB40F7C24A65B818

Certificate Slot Name: 1   Key Type: rsa
    (Private key in certlocal slot 1)
    Subject Name: <O=exampleco, OU=IT, C=US, CN=enigma>
    Key Size: 2048
    Public key hash: FEA65C5387BBF3B2C8F16C019FEBC388

Certificate Slot Name: 2   Key Type: rsa
```

```
    (Private key in certlocal slot 2)
    Subject Name: <O=exampleco, OU=IT, C=US, CN=partym>
    Key Size: 2048
    Public key hash: 80829EC52FC5BA910F4764076C20FDCF     Note 4
```

The public key hash and subject name of the last certificate stored in `enigma`'s public certificate database matches the hash of the private certificate for `partym` from the preceding step.

6 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 self-signed certs
# that we verified out of band. The local certificate
# is implicitly trusted because we have access to the private key.

cert_trust "O=exampleco, OU=IT, C=US, CN=enigma"

# We could also use the Alternate name of the certificate,
# if it was created with one.  In this example, the Alternate Name
# is in the format of an IP address:
# cert_trust "192.168.116.16"

## Parameters that may also show up in rules.

p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha256 encr_alg 3des }
p2_pfs 5

{
 label "US-partym to JA-enigmax"
 local_id_type dn
 local_id "O=exampleco, OU=IT, C=US, CN=partym"
 remote_id "O=exampleco, OU=IT, C=US, CN=enigma"

 local_addr  192.168.13.213
# We could explicitly enter the peer’s IP address here, but we don’t need
# to do this with certificates, so use a wildcard address. The wildcard
# allows the remote device to be mobile or behind a NAT box
 remote_addr 0.0.0.0/0


 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg sha256 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 on the local system.

```
...
{
 label "JA-enigmax to US-partym"
 local_id_type dn
 local_id "O=exampleco, OU=IT, C=US, CN=enigma"
 remote_id "O=exampleco, OU=IT, C=US, CN=partym"

 local_addr  192.168.116.16
 remote_addr 0.0.0.0/0
...
```

7 **On the peer systems, enable IKE.**

```
partym # svcadm enable ipsec/ike

enigma # svcadm enable ipsec/ike
```

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy.

▼ 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 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*. If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 15–1.

2 **Use the `ikecert certlocal -kc` command to create a certificate request.**

For a description of the arguments to the command, see Step b in "How to Configure IKE With Self-Signed Public Key Certificates" on page 263.

```
# 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 2048 -t rsa-sha1 \
> -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 2048 -t rsa-sha1 \
> -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-----
```

3 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 276 for an example of an embedded URI and an embedded DN entry in a public key certificate.

4 **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 252.

a. **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*. If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 15–1.

b. **Add the public key certificate that you received from the PKI organization.**

```
# ikecert certdb -a < /tmp/PKIcert.eml
```

c. **Add the CA from the PKI organization.**

```
# ikecert certdb -a < /tmp/PKIca.eml
```

d. **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
```

5 **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 sha384 encr_alg aes}
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 sha256 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
...
```

6 **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 276 for examples.

Example 18–2 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 < /tmp/saved.cert.eml
   ```

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.

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy.

▼ How to Generate and Store Public Key Certificates in 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 6000 Board" on page 286 for setup instructions.

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*. If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 15–1.

2 **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 6000 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 2048 -t rsa-sha1 \
> -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 6000 board.

- **For a certificate request, use this syntax.**

  ```
  # ikecert certlocal -kc -m 2048 -t rsa-sha1 \
  > -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.

3 At the prompt for a PIN, type the Sun Crypto Accelerator 6000 user, a colon, and the user's password.

If the Sun Crypto Accelerator 6000 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-----
```

4 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 3 of "How to Configure IKE With Certificates Signed by a CA" on page 268.

5 **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 sha256 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 sha256 encr_alg aes}
  }
```

6 Place the certificates from the other party in the hardware.

Respond to the PIN request as you responded in Step 3.

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 276.

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy.

▼ 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 6 in "How to Configure IKE With Certificates Signed by a CA" on page 268.
- 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 18–3.

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 Oracle. Details have been altered.

```
# ikecert certdb -lv example-protect.oracle.com
Certificate Slot Name: 0   Type: dsa-sha1
   (Private key in certlocal slot 0)
 Subject Name: <O=Oracle, CN=example-protect.oracle.com>
 Issuer Name: <CN=Oracle CA (Cl B), O=Oracle>
 SerialNumber: 14000D93
   Validity:
      Not Valid Before: 2011 Sep 19th, 21:11:11 GMT
      Not Valid After:  2015 Sep 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.oracle.com
      Key Usage: DigitalSignature KeyEncipherment
      [CRITICAL]
   CRL Distribution Points:
      Full Name:
         URI = #Ihttp://www.oracle.com/pki/pkismica.crl#i
         DN = <CN=Oracle CA (Cl B), O=Oracle>
      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.oracle.com:389,ldap2.oracle.com"
  ...
  ```

IKE retrieves the CRL and caches the CRL until the certificate expires.

Example 18–3 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 < Oracle.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 279

Task	Description	For Instructions
Use a CA's public 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 18–4
Use a CA's public 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 18–5
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 18–6
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 18–7

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 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*. If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 15–1.

2 Configure the central system to recognize mobile systems.

a. Configure 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 sha256 sa shared}
```

b. Configure the IKE configuration 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
# CA's public 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 aes auth_alg sha256 }
}
```

3 **Log in to each mobile system, and configure the system to find the central system.**

a. **Configure 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
central 192.xxx.xxx.x
```

b. **Configure 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 sha256 sa shared}
```

c. **Configure the IKE configuration 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 aes auth_alg sha256 }
   }
```

4 **Enable the `ike` service.**

```
# svcadm enable svc:/network/ipsec/ike
```

Example 18–4 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, the CA's public certificate has 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 18–5.

```
## /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 sha256 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

# CA's public 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 aes auth_alg sha256}
p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg aes auth_alg sha256}
p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg aes auth_alg sha256}
p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg aes auth_alg sha256}
}
```

Example 18–5 Configuring a System Behind a NAT With IPsec

In the following example, the CA's public certificate is 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 18–4.

```
## /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 sha256 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 aes auth_alg sha256 }
}
```

Example 18–6 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 18–7.

```
## /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 sha256 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 aes auth_alg sha256 }
}
```

Example 18–7 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 18–6.

```
## /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 sha256 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 aes auth_alg sha256 }
}
```

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy.

Configuring IKE to Find Attached Hardware

Public key certificates can also be stored on attached hardware. The Sun Crypto Accelerator 6000 board provides storage, and enables public key operations to be offloaded from the system to the board.

▼ How to Configure IKE to Find the Sun Crypto Accelerator 6000 Board

Before You Begin The following procedure assumes that a Sun Crypto Accelerator 6000 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 6000 Board Version 1.1 User's Guide*.

1. **Become an administrator.**

   For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*. If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 15–1.

2. **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 6000 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
   $
   ```

3. **Find the token ID for the attached Sun Crypto Accelerator 6000 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 in Hardware" on page 273.

   The trailing spaces are automatically padded by the `ikecert` command.

Example 18–8 Finding and Using Metaslot Tokens

Tokens can be stored on disk, on an attached board, or in the softtoken keystore that the Cryptographic 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
```

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy.

◆ ◆ ◆ CHAPTER 19

Internet Key Exchange (Reference)

This chapter contains the following reference information about IKE:

For instructions on implementing IKE, see Chapter 18, "Configuring IKE (Tasks)." For overview information, see Chapter 17, "Internet Key Exchange (Overview)."

IKE Service

`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 "`ikeadm` Command" on page 291.

For information about SMF, see Chapter 6, "Managing Services (Overview)," in *Oracle Solaris Administration: Common Tasks*. 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 an Oracle 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. To modify properties of the IKE service, see "How to Manage IPsec and IKE Services" on page 240. 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 Configuration File

The IKE configuration file, /etc/inet/ike/config, manages the keys for the interfaces that are being protected in the IPsec policy file, /etc/inet/ipsecinit.conf.

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 257. 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 the policies in the IPsec configuration file, /etc/inet/ipsecinit.conf. The IKE policy file determines if perfect forward security (PFS) is used when creating the IPsec SAs.

The /etc/inet/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 245.

ikeadm 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 IPsec and IKE Services" on page 240.

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 247.

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 Keys" on page 236. The IKE daemon does not use this file. The keying material that IKE generates for IPsec SAs is stored in the kernel.

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 board, or in a softtoken keystore. The softtoken keystore is available when the metaslot in the 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 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 19–1 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 dsa_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.

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 263. 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 268.

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 276.

`/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 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. 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 supported hardware.

`/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 20

IP Filter in Oracle Solaris (Overview)

This chapter provides an overview of IP Filter, an Oracle Solaris feature. For IP Filter tasks, see Chapter 21, "IP Filter (Tasks)."

This chapter contains the following information:

Introduction to IP 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 IP Filter software. To view license terms, attribution, and copyright statements for open source 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.

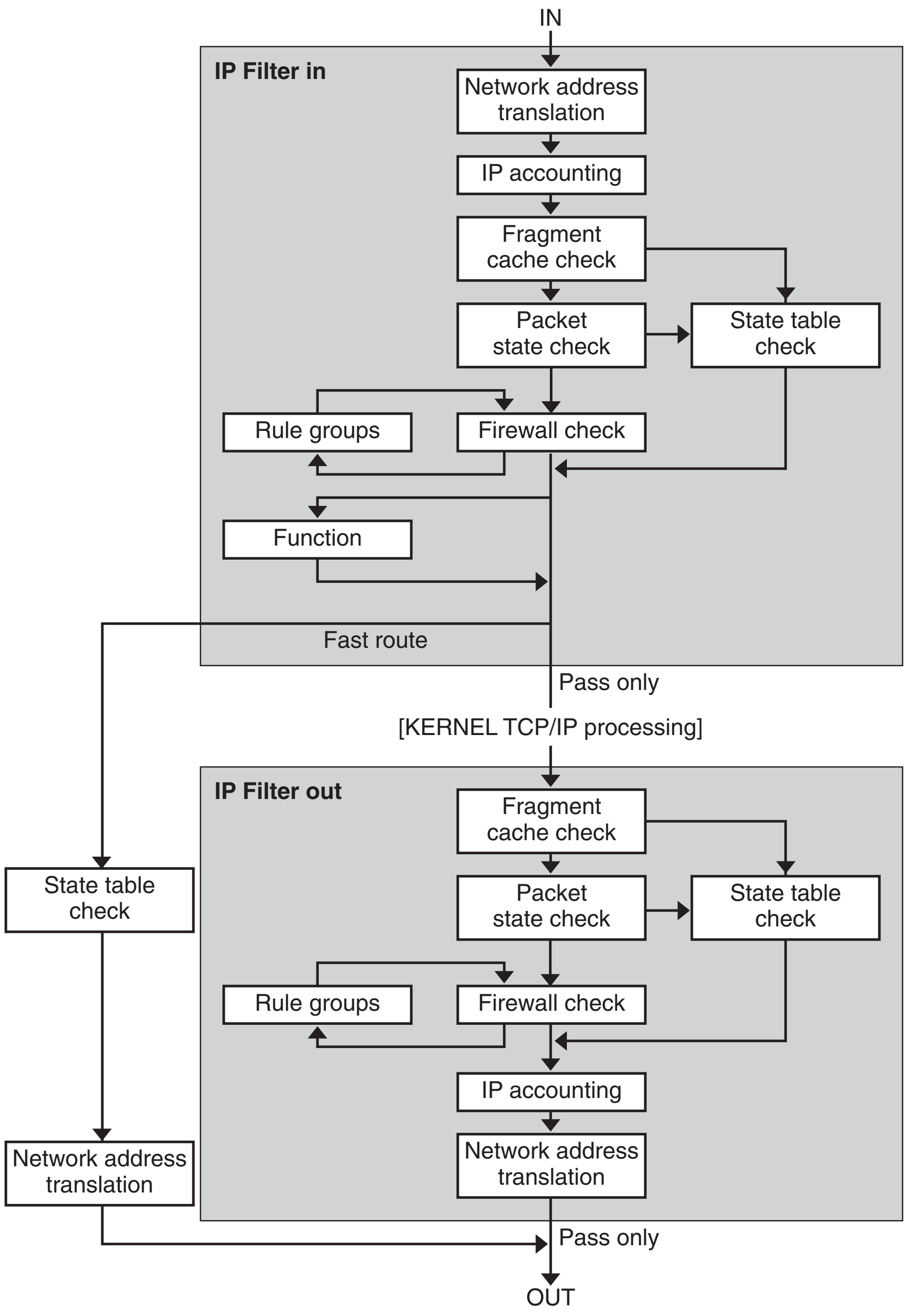
Information Sources for Open Source IP Filter

The home page for the open source IP Filter software by Darren Reed is found at `http://coombs.anu.edu.au/~avalon/ip-filter.html`. This site includes information for open source IP Filter, 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 IP Filter.

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 20–1 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 18, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*. For information on the step-by-step procedures that are associated with SMF, see Chapter 19, "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 311.
- 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 RBAC (Task Map)" in *System Administration Guide: Security Services*.
- IP Network Multipathing (IPMP) supports stateless filtering only.

  For IP Filter to perform stateless filtering on traffic to and from an IPMP group, you must set the `ipmp_hook_emulation` parameter. By default, the parameter is set to zero (`0`), which means that IP Filter cannot perform stateful packet inspection of traffic on physical interfaces that belong to an IPMP group. To enable IPMP packet filtering, issue the following command:

  ```
  ndd -set /dev/ip ipmp_hook_emulation 1
  ```
- Oracle Solaris Cluster software does not support filtering with IP Filter for scalable services, but does support IP Filter for failover services. For guidelines and restrictions when configuring IP Filter in a cluster, see "Oracle Solaris OS Feature Restrictions" in *Oracle Solaris Cluster Software Installation Guide*.
- Filtering between zones is supported provided that the IP Filter rules are implemented in a zone that functions as a virtual router for the other zones on the system.

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 334.

Using 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 318. For an explanation of the IP address scheme (`192.168.0.0/16`) shown in the example, see Chapter 1, "Planning the Network Deployment."

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 21, "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.

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.

Once packet filtering rules for IPv6 have been set up, activate IPv6 packet filtering capabilities by creating the interface.

For more information on IPv6, see Chapter 3, Introducing IPv6 (Overview). For tasks associated with IP Filter, see Chapter 21, "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.

Man Page	Description
ipf(4)	Contains the grammar and syntax for creating IP Filter packet filtering rules.
ipfilter(5)	Provides open source IP Filter licensing information.
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 21

IP Filter (Tasks)

This chapter provides step-by-step instructions for tasks. For overview information about IP Filter, see Chapter 20, "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 21–1 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 312
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 313

TABLE 21–1 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 314

▼ 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 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 302. For information about working with configuration files, see "Creating and Editing IP Filter Configuration Files" on page 334.

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 305.

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 306.

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 314. 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 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 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 21–2 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 315
Deactivate NAT.	Deactivate NAT using the `ipnat` command.	"How to Deactivate NAT" on page 316
Disable packet filtering and NAT.	Disable packet filtering and NAT using the `ipf` command.	"How to Disable Packet Filtering" on page 316

▼ 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 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 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 313.

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 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 21–3 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 318
	View an active packet filtering rule set.	"How to View the Active Packet Filtering Rule Set" on page 318
	View an inactive packet filtering rule set.	"How to View the Inactive Packet Filtering Rule Set" on page 319
	Activate a different active rule set.	"How to Activate a Different or Updated Packet Filtering Rule Set" on page 319
	Remove a rule set.	"How to Remove a Packet Filtering Rule Set" on page 320
	Add rules to the rule sets.	"How to Append Rules to the Active Packet Filtering Rule Set" on page 321 "How to Append Rules to the Inactive Packet Filtering Rule Set" on page 322
	Move between active and inactive rule sets.	"How to Switch Between Active and Inactive Packet Filtering Rule Sets" on page 322
	Delete an inactive rule set from the kernel.	"How to Remove an Inactive Packet Filtering Rule Set From the Kernel" on page 323
Manage, view and modify IP Filter NAT rules.		"Managing NAT Rules for IP Filter" on page 324
	View active NAT rules.	"How to View Active NAT Rules" on page 324
	Remove NAT rules.	"How to Remove NAT Rules" on page 325
	Add additional rules to NAT rules.	"How to Append Rules to the NAT Rules" on page 325

TABLE 21–3 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 326
	View active address pools.	"How to View Active Address Pools" on page 326
	Remove an address pool.	"How to Remove an Address Pool" on page 326
	Add additional rules to an address pool.	"How to Append Rules to an Address Pool" on page 327

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 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 21–1 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 RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **View the inactive packet filtering rule set.**

   ```
   # ipfstat -I -io
   ```

Example 21–2 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 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 21–3 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 21–4 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.)

# ipf -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 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 21–5 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 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 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 21–6 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 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 21–7 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 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 21–8 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 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 21–9 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 RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **View the active NAT rules.**

```
# ipnat -l
```

Example 21–10 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 RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Remove the current NAT rules.**

```
# ipnat -C
```

Example 21–11 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 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 21–12 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 RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **View the active address pool.**

```
# ippool -l
```

Example 21–13 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 RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Remove the entries in the current address pool.**

```
# ippool -F
```

Example 21–14 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 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 21–15 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 21–4 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 328
View state statistics.	View statistics on packet state information using the `ipfstat -s` command.	"How to View State Statistics for IP Filter" on page 329
View NAT statistics.	View NAT statistics using the `ipnat -s` command.	"How to View NAT Statistics for IP Filter" on page 330
View address pool statistics.	View address pool statistics using the `ippool -s` command.	"How to View Address Pool Statistics for IP Filter" on page 330

▼ 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 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 21–16 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 RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **View the state statistics.**

   ```
   # ipfstat -s
   ```

Example 21–17 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 RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **View NAT statistics.**

```
# ipnat -s
```

Example 21–18 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 RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **View address pool statistics.**

```
# ippool -s
```

Example 21–19 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 21–5 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 331
View log files.	View state, NAT, and normal log files using the `ipmon` command.	"How to View IP Filter Log Files" on page 332
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 333
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 334

▼ 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 RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Edit the `/etc/syslog.conf` file by adding the following two lines:**

   ```
   # Save IP Filter 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 21–20 Creating a IP Filter Log

The following example shows how to create `ipmon.log` to archive IP Filter information.

In `/etc/syslog.conf`:

```
# Save IP Filter 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 331.

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 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 21–21 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 RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Flush the packet log buffer.**

```
# ipmon -F
```

Example 21–22 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 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 21–23 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 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 319.

  See "Using IP Filter's Packet Filtering Feature" on page 302 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 305 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 306 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 21–24 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 21–25 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 21–25 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 21–26 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 21–26 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 21–26 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
```

P A R T I V

Networking Performance

This part discusses networking performance features such as integrated load balancing and virtual router redundancy protocol.

CHAPTER 22

Integrated Load Balancer Overview

Integrated Load Balancer (ILB), a feature of Oracle Solaris, provides Layer 3 and Layer 4 load-balancing capabilities for Oracle Solaris 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:

- "ILB Terminology" on page 344
- "Features of ILB" on page 346
- "ILB Processes" on page 350
- "Guidelines for Using ILB" on page 351
- "ILB and the Service Management Facility" on page 352
- "ILB Command and Subcommands" on page 352

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 the IP Filter feature of Oracle Solaris. 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 "Initially Configuring RBAC (Task Map)" in *Oracle Solaris Administration: 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 352 . 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 22–1 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 "Initially Configuring RBAC (Task Map)" in *Oracle Solaris Administration: Security Services*.
- To enable auditing of ILB configuration commands, you must preselect the system-wide administration audit class. To do so, see "Configuring the Audit Service (Task Map)" in *Oracle Solaris Administration: Security Services*.
- ILB userland components are delivered as separate IPS package in the Oracle Solaris repository with package names starting with `SUNWilb`. You must download these packages from the Oracle Solaris repository using the `pkg install` command. For instructions on installing ILB, see "Installing the Integrated Load Balancer" on page 355.
- 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 6, "Managing Services (Overview)," in *Oracle Solaris Administration: Common Tasks*. For step-by-step procedures that are associated with SMF, see Chapter 7, "Managing Services (Tasks)," in *Oracle Solaris Administration: Common Tasks*.

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 22–2 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 22–2 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 23

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.

1ILB has two portions, the kernel and the userland. The kernel portion is automatically installed as a part of the Oracle Solaris 11 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, 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 "Initially Configuring RBAC (Task Map)" in *Oracle Solaris Administration: Security Services*.

2 **Enable the appropriate forwarding service either IPv4 or IPv6 or both of them.**

```
#ipadm set-prop -p forwarding=on ipv4
# ipadm set-prop -p forwarding=on ipv6
```

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 “Initially Configuring RBAC (Task Map)” in *Oracle Solaris Administration: Security Services*.

2 **Disable the ILB service.**

```
# svcadm disable ilb
```

3 **Verify that the ILB service is disabled.**

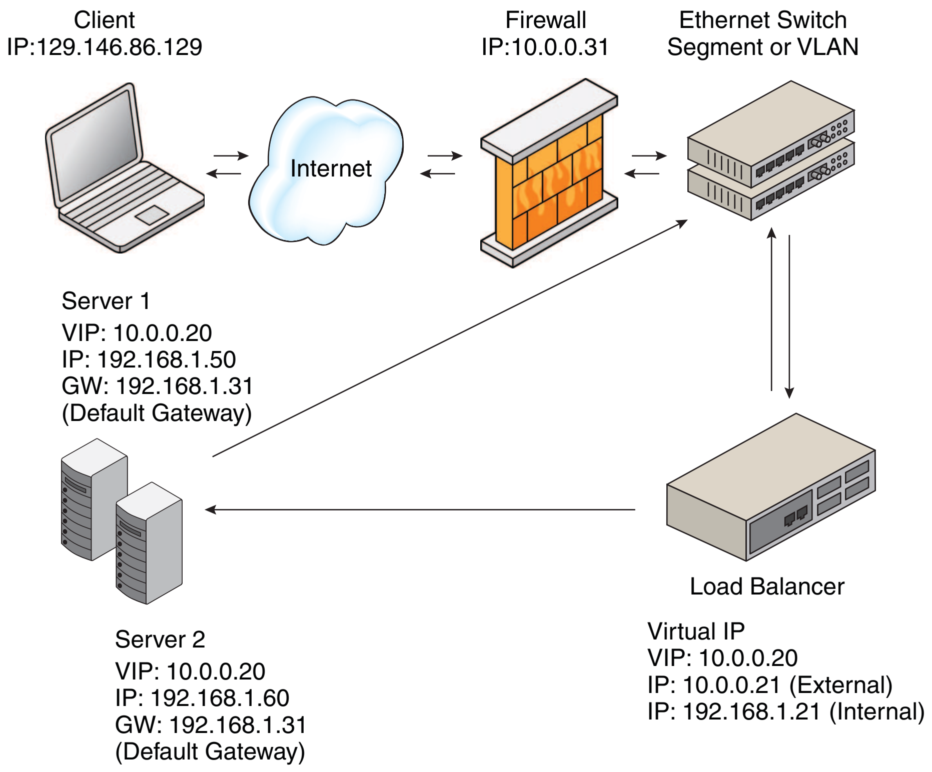
```
# svcs ilb
```

Configuring ILB

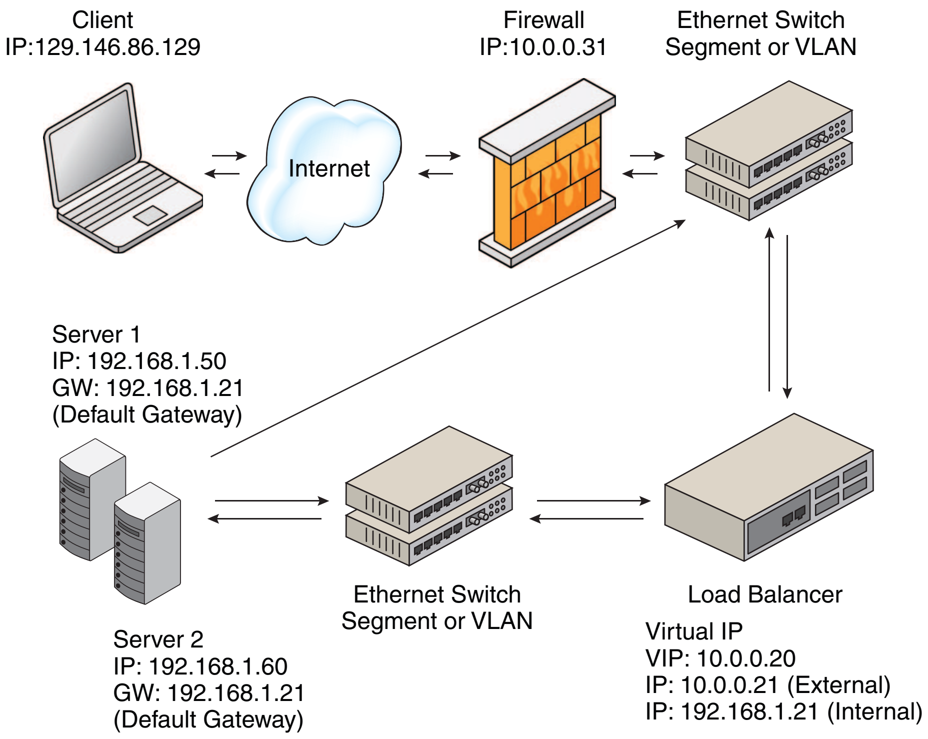
This section describes the implementation of ILB with DSR, half-NAT, and full-NAT topologies.

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 23–1 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 23–2 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 23–3 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

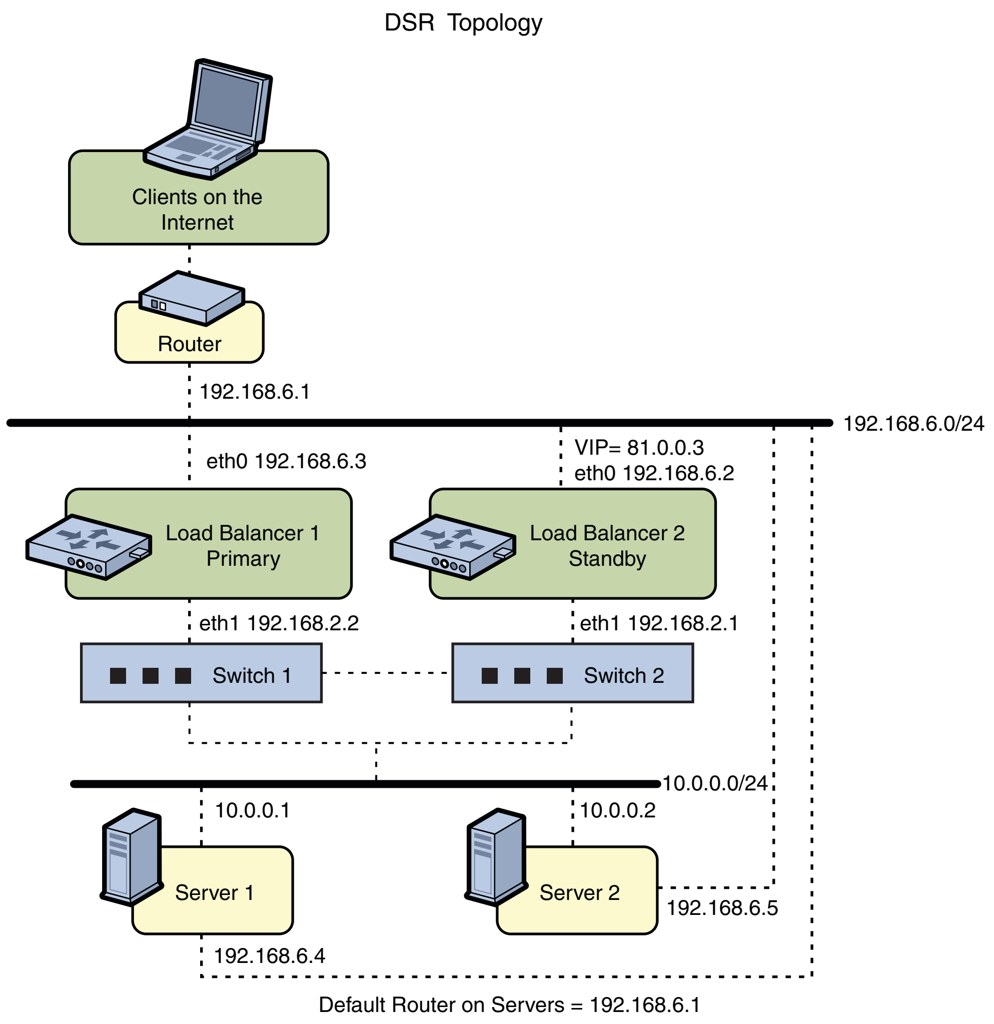
ILB High-Availability Configuration (Active-Passive Mode Only)

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`.

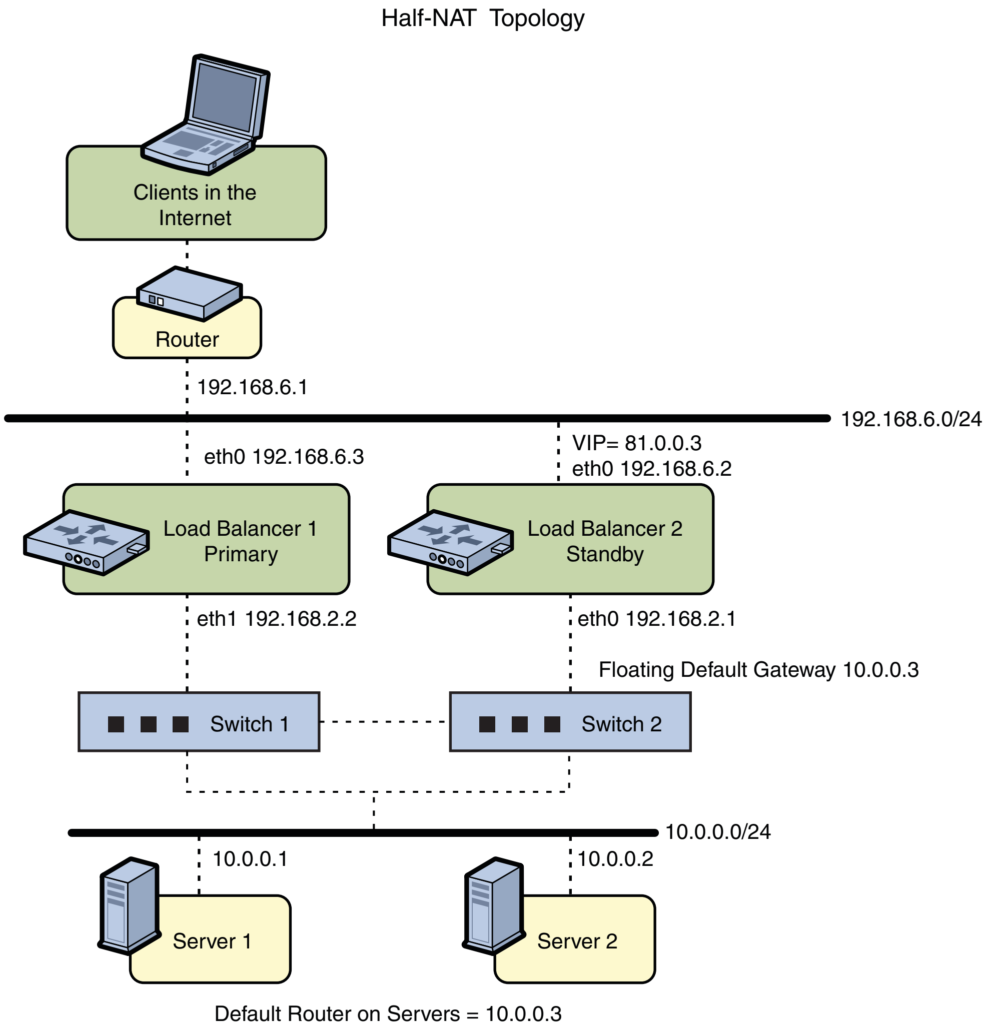
  When Load Balancer 1 recovers, Load Balancer 2 returns to the standby mode.
- If one or both of the Load Balancer 1's interfaces fails, Load Balancer 2 takes over as the primary. Thus, the Load Balancer 2 takes over address resolution for VIP `81.0.0.3` and handles all the packets from clients with the 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 *Oracle Solaris Administration: 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 344.

▼ 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 23–1 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 23–2 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
specgroup	_specgroup.0	7001	7001	199.199.67.18
specgroup	_specgroup.1	7001	7001	199.199.67.19
test123	_test123.0	7001	7001	199.199.67.18
test123	_test123.1	7001	7001	199.199.67.19

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 344.

▼ 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 -s server=192.168.89.1,192.168.89.2 ftpgroup
# ilbadm add-server -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 23–3 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 -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 server=_sg1.2 sg1
   ```

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 23–4 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 server=_sg1.2 sg1
```

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 websg.1
   # ilbadm disable-server websg.1
   ```

Example 23–5 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 websg.1
# ilbadm disable-server 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 344.

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 delete-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 show-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 344.

▼ 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 23–6 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 show-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 344.

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-statistics
 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 23–7 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 23–8 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 24

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 Oracle Solaris to provide high availability. Oracle Solaris 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 378.

This chapter includes the following sections:

- “VRRP Terminology” on page 378
- “VRRP Architectural Overview” on page 378
- “VRRP Limitations” on page 381

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 Oracle Solaris, 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 card (NIC). A physical interface can have more than one VNIC. VNICs are essential components of network virtualization. For more information, see Part III, " Network Virtualization and Resource Management," in *Oracle Solaris Administration: 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

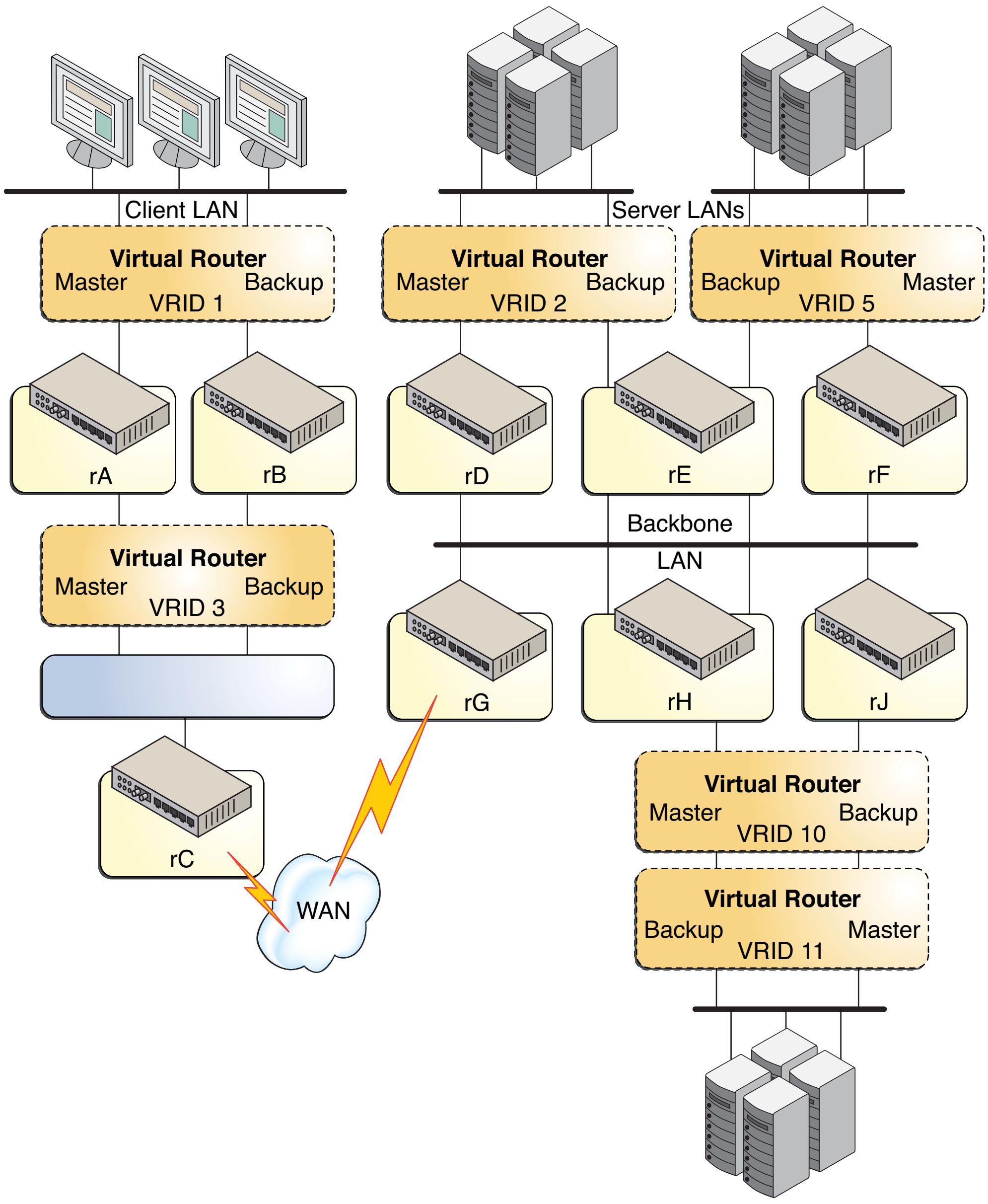
VRRP runs on each VRRP router and manages the state of the router. A host can have multiple VRRP routers configured, where each VRRP router belongs to a different virtual router.

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.


Client LAN
Server LANs
Virtual Router
Master
Backup
VRID 1
Virtual Router
Master
Backup
VRID 2
Virtual Router
Backup
Master
VRID 5
rA
rB
rD
rE
rF
Backbone
LAN
Virtual Router
Master
Backup
VRID 3
rG
rH
rJ
rC
WAN
Virtual Router
Master
Backup
VRID 10
Virtual Router
Backup
Master
VRID 11

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 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 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 25

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:

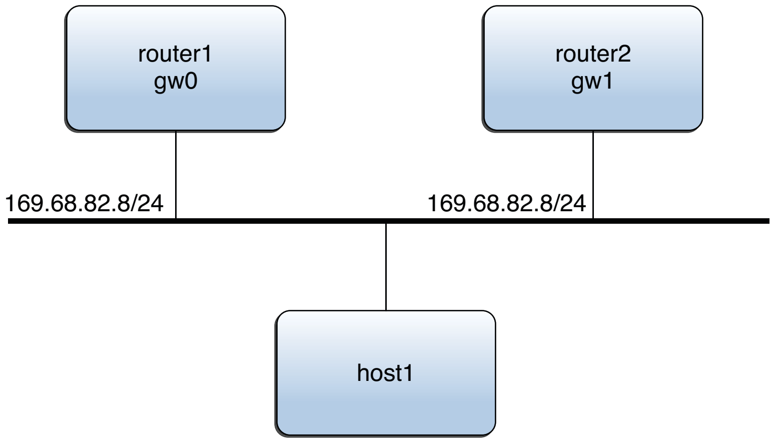
```
# vrrpadm show-router vrrp1
NAME VRID LINK AF PRIO ADV_INTV MODE STATE VNIC
vrrp1 1 bge1 IPv4 100 1000 e-pa- BACK vnic1

# 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 25–1 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 25–1 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 26

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

The following table lists and describes the supported algorithms.

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

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 value 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 an algorithm or the `-=` modifier to remove an algorithm.

▼ How to Implement TCP and SCTP Network Congestion Control

1 Become an administrator.

For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: Security Services*.

2 Display the current values of the TCP 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 properties are displayed.

The command displays both the current values as well as the possible algorithms that you can assign to the properties.

3 Set the congestion control properties of the TCP 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 you add or remove algorithms. You can remove an algorithm before adding other algorithms to a property. However, the `cong_default` property must always have a defined algorithm.

5 **(Optional) Display the new values of the congestion control properties.**

```
# ipadm show-prop -p cong_enabled,cong_default tcp
```

Example 26–1 Setting Algorithms for Congestion Control

This example changes the default algorithm for 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
```

P A R T 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 27

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" on page 395
- "Providing Quality of Service With IPQoS" on page 398
- "Improving Network Efficiency With IPQoS" on page 399
- "Differentiated Services Model" on page 400
- "Traffic Forwarding on an IPQoS-Enabled Network" on page 405

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 413.

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 469.

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 469.

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 415.

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 418.

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 419.

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 422.

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

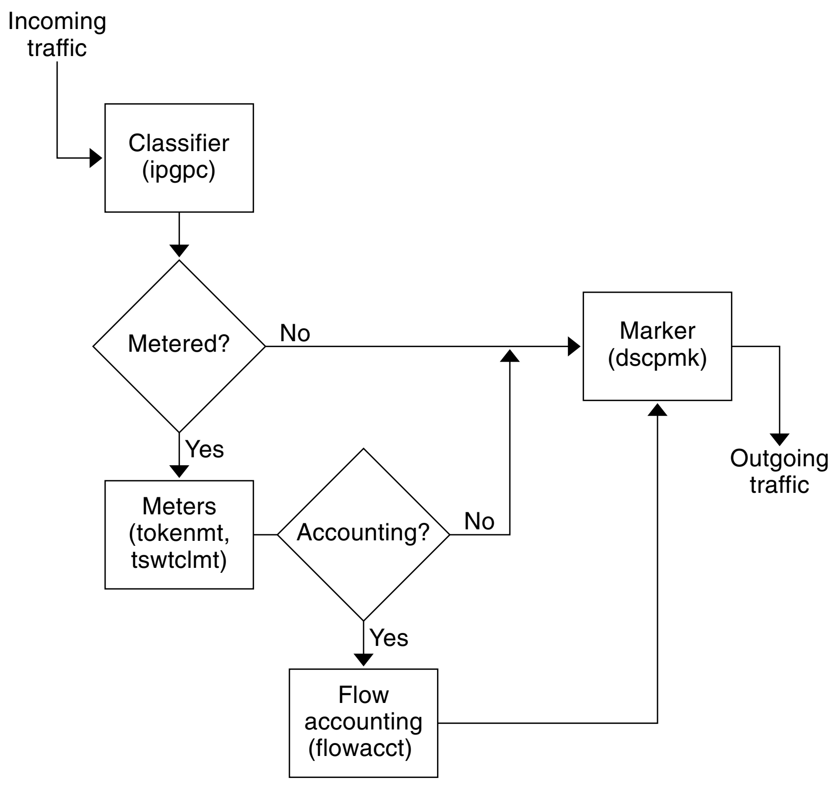
You can also gather statistics on other attributes, as described in "Recording Information About Traffic Flows" on page 463, and in the `flowacct`(7ipp) and `acctadm`(1M) man pages.

For information on planning a flow-accounting strategy, see "How to Plan for Flow Accounting" on page 424.

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 27–1 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 474.

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 475.

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

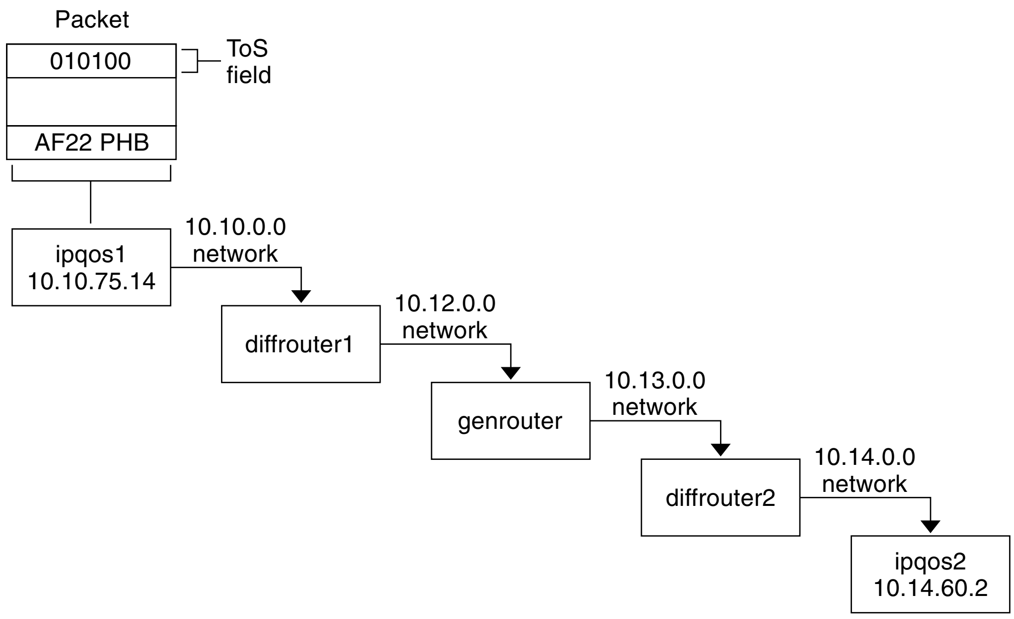
The *assured forwarding (AF)* per-hop behavior provides four different forwarding classes that you can assign to a packet. Every forwarding class provides three drop precedences, as shown in Table 32–2.

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 27–2 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 28

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)" on page 409
- "Planning the Diffserv Network Topology" on page 410
- "Planning the Quality-of-Service Policy" on page 413
- "QoS Policy Planning (Task Map)" on page 414
- "Introducing the IPQoS Configuration Example" on page 425

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 410.
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 413.

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 413.
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™ servers from Oracle. 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.

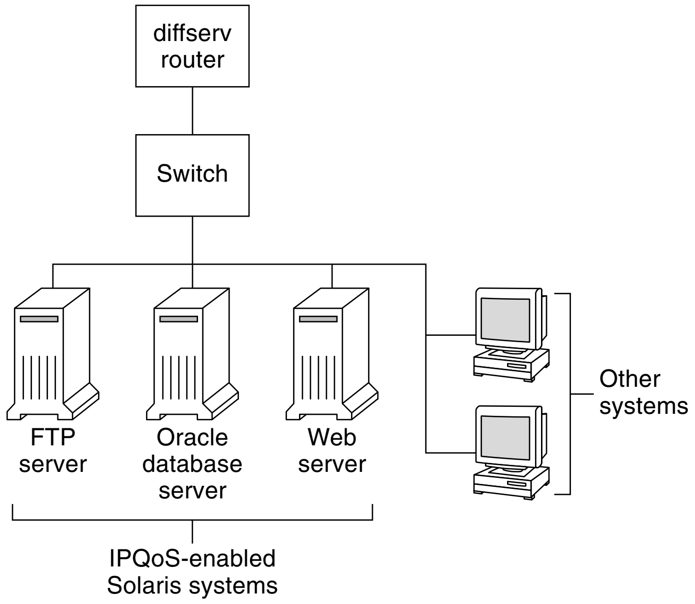
IPQoS Network Topologies

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 28–1 IPQoS Systems on a Network Segment



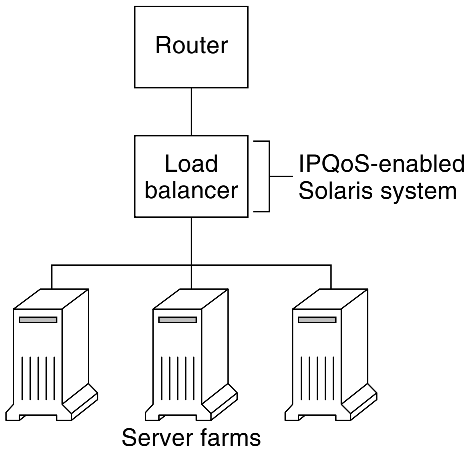
This network is but one segment of a corporate intranet. By enabling IPQoS on the application servers and web servers, you can control the rate at which each IPQoS system releases outgoing traffic. If you make the router Diffserv aware, you can further control incoming and outgoing traffic.

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 28–4.

IPQoS on a Network of Server Farms

The following figure shows a network with several heterogeneous server farms.

FIGURE 28–2 Network of IPQoS-Enabled Server Farms



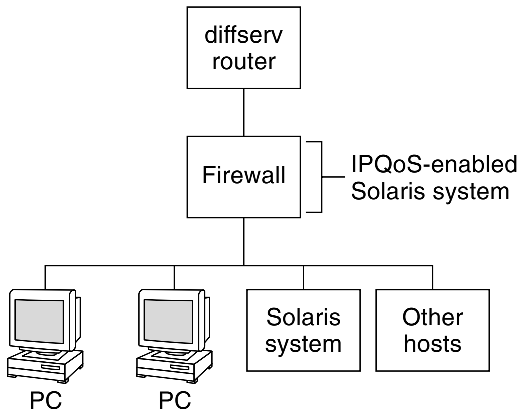
In such a topology, the router is Diffserv aware, and therefore able to queue and rate both incoming and outgoing traffic. The load balancer is also Diffserv-aware, and the server farms are IPQoS enabled. The load balancer can provide additional filtering beyond the router by using selectors such as 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 28–3 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 28–1 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 28–1 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 415
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 415
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 418
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 419
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 422
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 424

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 410.

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 401.

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 415.

1. **Create a QoS planning table for organizing the QoS policy information.**

   For suggestions, refer to Table 28–1.

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 418.**

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 422.

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 401.

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 32–1.

TABLE 28–2 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 415.
`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 415.

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 415.**

   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 28–1 to fill in filters for the classes you defined.

Example 28–1 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 419.
- To define forwarding behaviors for flows as the flows return to the network stream, refer to "How to Plan Forwarding Behavior" on page 422.
- To plan for flow accounting of certain types of traffic, refer to "How to Plan for Flow Accounting" on page 424.
- To add more classes to the QoS policy, refer to "How to Define the Classes for Your QoS Policy" on page 415.
- To add more filters to the QoS policy, refer to "How to Define Filters in the QoS Policy" on page 418.

▼ 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 402.

The next procedure assumes that you have defined filters and selectors, as described in "How to Define Filters in the QoS Policy" on page 418.

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 415.**

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 473. 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 474 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 472.

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 28–2 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`
			`sport imap`
			`direction LOCAL_OUT`
				meter=`tokenmt`
				committed rate=5000000
				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 422.

- To plan for flow accounting of certain types of traffic, refer to "How to Plan for Flow Accounting" on page 424.
- To add more classes to the QoS policy, refer to "How to Define the Classes for Your QoS Policy" on page 415.
- To add more filters to the QoS policy, refer to "How to Define Filters in the QoS Policy" on page 418.
- To define another flow-control scheme, refer to "How to Plan Flow Control" on page 419.
- To create an IPQoS configuration file, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 433.

▼ 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 477.

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 405.

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 475.

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 28–3 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 32–2.

See Also

- To plan for flow accounting of certain types of traffic, refer to "How to Plan for Flow Accounting" on page 424.
- To add more classes to the QoS policy, refer to "How to Define the Classes for Your QoS Policy" on page 415.
- To add more filters to the QoS policy, refer to "How to Define Filters in the QoS Policy" on page 418.
- To define a flow-control scheme, refer to "How to Plan Flow Control" on page 419.
- To define additional forwarding behaviors for flows as the packets return to the network stream, refer to "How to Plan Forwarding Behavior" on page 422.
- To create an IPQoS configuration file, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 433.

▼ 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 415.
- To add more filters to the QoS policy, refer to "How to Define Filters in the QoS Policy" on page 418.

- To define a flow-control scheme, refer to "How to Plan Flow Control" on page 419.
- To define forwarding behaviors for flows as the packets return to the network stream, refer to "How to Plan Forwarding Behavior" on page 422.
- To plan for additional flow accounting of certain types of traffic, refer to "How to Plan for Flow Accounting" on page 424.
- To create the IPQoS configuration file, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 433.

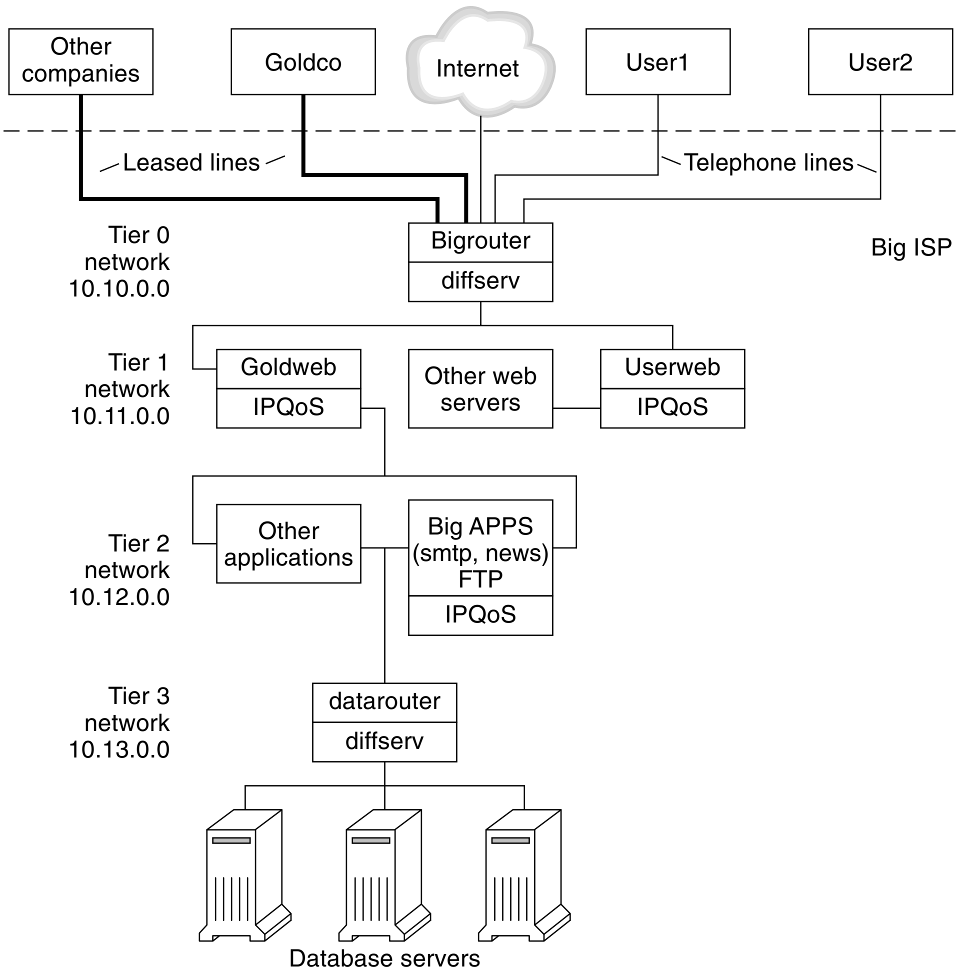
Introducing the IPQoS Configuration Example

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 28–4 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 29

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 413.

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 415
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 413

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 433
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 435
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 441
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 450
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 437
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 440
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 456
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 453

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 456.

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 413. 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 32–3 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 28–4.

- `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 28–4.

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 29–1 Sample IPQoS Configuration File for a Premium Web Server

```
fmt_version 1.0

action {
    module ipgpc
    name ipgpc.classify
    params {
        global_stats TRUE
```

EXAMPLE 29–1 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 29–2 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 29–1.

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 29–1.

---

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 482 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 484 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 483 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 435.
- To create another class clause for the configuration file, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 433.

▼ 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 433.

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 29–1.

---

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 484.

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 437.
- To define flow-control parameters for the metering modules, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 450.
- To activate the IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 456.
- To define additional filters, refer to "How to Define Filters in the IPQoS Configuration File" on page 435.
- To create classes for traffic flows from applications, refer to "How to Configure the IPQoS Configuration File for an Application Server" on page 446.

▼ 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 29–1.

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 477.

---

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 32–2.

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 475.

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 453.

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 440.
- To define forwarding behaviors for the marker modules, refer to "How to Define Traffic Forwarding in the IPQoS Configuration File" on page 437.
- To define flow-control parameters for the metering modules, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 450.

- To activate the IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 456.
- To define additional filters, refer to "How to Define Filters in the IPQoS Configuration File" on page 435.
- To create classes for traffic flows from applications, refer to "How to Configure the IPQoS Configuration File for an Application Server" on page 446.

▼ 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 433. 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 29–1.

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 453.
- To activate the IPQoS configuration file, refer to “How to Apply a New Configuration to the IPQoS Kernel Modules” on page 456.
- To create classes for traffic flows from applications, refer to “How to Configure the IPQoS Configuration File for an Application Server” on page 446.

▼ 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 29–2.

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 433.

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 433.

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 435.

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 446.

- To configure per-hop behaviors on a router, refer to "How to Configure a Router on an IPQoS-Enabled Network" on page 453.
- To activate your IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 456.

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 28–4.

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 29–3 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 29–3 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 433.

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 433.

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 435.
- To define forwarding behaviors for application traffic, refer to "How to Configure Forwarding for Application Traffic in the IPQoS Configuration File" on page 448.
- To configure flow control by using the metering modules, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 450.
- To configure flow accounting, refer to "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 440.

▼ 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 29–3.

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 32–2.

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 433.
- To configure flow control by using the metering modules, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 450.
- To configure flow accounting, refer to "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 440.
- To configure forwarding behaviors on a router, refer to "How to Configure a Router on an IPQoS-Enabled Network" on page 453.
- To activate the IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 456.

▼ 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 29–3. 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 473.

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 472.

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 32–2.

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 32–2.

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 456.
- To add configuration information for web servers, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 433.
- To configure flow accounting, refer to "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 440.
- To configure forwarding behaviors on a router, refer to "How to Configure a Router on an IPQoS-Enabled Network" on page 453.

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 410. 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 28–4.

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 30

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)" on page 455
- "Applying an IPQoS Configuration" on page 456
- "Enabling `syslog` Logging for IPQoS Messages" on page 457
- "Troubleshooting with IPQoS Error Messages" on page 458

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 429.

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 456
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 457.
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 457.
4. Fix any IPQoS problems that arise.	Troubleshoot IPQoS problems by using error messages.	Refer to the error messages in Table 30–1.

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 431. 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 466.
- To log `ipqosconf` messages, refer to "Enabling `syslog` Logging for IPQoS Messages" on page 457.
- 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 457.

▼ 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 456.

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 30–1 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 30–1.

Troubleshooting with IPQoS Error Messages

This section contains a table of error messages that are generated by IPQoS and their possible solutions.

TABLE 30–1 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 433.

TABLE 30–1 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 beginning with the Solaris 9 9/02 release 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 433.
`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 456.
`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 32–5.
`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 30–1 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 469. 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 469. 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 469. 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 469. Alternatively, you can refer to the `ipqosconf`(1M) man page.

TABLE 30–1 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 469. 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 469. 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 31

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)" on page 463
- "Recording Information About Traffic Flows" on page 463
- "Gathering Statistical Information" on page 466

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 464
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 440

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 32–4.

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 450.
- To learn how `flowacct` works, refer to "Classifier Module" on page 469.
- 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 32–4. 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 29–1.

```
# /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 Oracle 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 *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

See Also

- For detailed information on the extended accounting feature, refer to Chapter 4, "Extended Accounting (Overview)," in *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, 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 440.
- To print the data in the file that was created with `acctadm`, refer to "Perl Interface to libexacct" in *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, 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 32–5. 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 31–1 `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 31–1 `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 32

IPQoS in Detail (Reference)

This chapter contains reference materials that provide in-depth details about the following IPQoS topics:

- "IPQoS Architecture and the Diffserv Model" on page 469
- "IPQoS Configuration File" on page 481
- "`ipqosconf` Configuration Utility" on page 485

For an overview, refer to Chapter 27, "Introducing IPQoS (Overview)." For planning information, refer to Chapter 28, "Planning for an IPQoS-Enabled Network (Tasks)." For procedures for configuring IPQoS, refer to Chapter 29, "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 401. For information on invoking the classifier in the IPQoS configuration file, refer to "IPQoS Configuration File" on page 481.

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 32–1 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 32–1 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 29–3.

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 32–1 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 405.

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 405.

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 32–2 Assured Forwarding Codepoints

	Class 1	Class 2	Class 3	Class 4
Low-Drop Precedence	AF11 =	AF21 =	AF31 =	AF41 =
	10 (001010)	18 (010010)	26 (011010)	34 (100010)
Medium-Drop Precedence	AF12 =	AF22 =	AF32 =	AF42 =
	12 (001100)	20 (010100)	28 (011100)	36 (100100)
High-Drop Precedence	AF13 =	AF23 =	AF33 =	AF43 =
	14 (001110)	22 (010110)	30 (011110)	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 32–2 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 32–3 801.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 32–2 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 27–1. 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 463.

`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 450.

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 32–4 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 32–4.
`-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 29, "Creating the IPQoS Configuration File (Tasks)."

The syntax of the IPQoS configuration file is shown in Example 32–3. 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 32–3 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 32–3 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 469.

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 32–5.
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 32–5 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 482.

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 32–1.

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

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 algorithm 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 algorithm	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 from Oracle support DR. Some Sun platforms from Oracle 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 associations (SAs).
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 associations (SAs).
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 algorithm.
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 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