System Administration Guide: IP Services

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

Part No: 806–4075–10
May 2002

011210@2870

Contents

Preface

System Administration Guide, IP Services is part of a seven-volume set that covers a significant part of the Solaris™ system administration information. This book assumes that you have already installed the SunOS™ 5.9 operating system, and you have set up any networking software that you plan to use. The SunOS 5.9 operating system is part of the Solaris product family, which also includes the Solaris Common Desktop Environment (CDE). The SunOS 5.9 operating system is compliant with AT&T's System V, Release 4 operating system.

Note – The Solaris operating environment runs on two types of hardware, or platforms—SPARC™ and IA. The Solaris operating environment runs on both 64-bit and 32-bit address spaces. The information in this document pertains to both platforms and address spaces unless called out in a special chapter, section, note, bullet, figure, table, example, or code example.

Who Should Use This Book

This book is intended for anyone responsible for administering one or more systems that run the Solaris 9 release. To use this book, you should have one to two years of UNIX® system administration experience. Attending UNIX system administration training courses might be helpful.

How This Book Is Organized

Chapter 2 provides an overview of TCP/IP and its components. This chapter also provides an introduction to the Internet Protocol Suite.

Chapter 3 describes how to plan your TCP/IP network.

Chapter 4 describes how to administer TCP/IP within the Solaris operating environment on your network.

Chapter 5 provides reference information about the implementation of TCP/IP on your network.

Chapter 7 provides an overview of DHCP and its components. This chapter also describes how DHCP works within the Solaris operating environment on your network.

Chapter 8 describes what you need to do before setting up the DHCP service on your network.

Chapter 9 describes how you configure the DHCP service on your network. This chapter also provides instructions on how to use the DHCP Manager.

Chapter 10 describes tasks that enable you to administer the Solaris DHCP service on your network.

Chapter 11 provides information to help you solve problems that you might encounter when you configure a DHCP server or client.

Chapter 12 provides reference information about the commands and files that are used by the DHCP service on your network.

Chapter 14 provides an overview of the new Internet Protocol that is known as IPv6.

Chapter 15 provides procedures for enabling IPv6 and IPv6 routers, configuring IPv6 addresses for DNS, NIS, and NIS+, creating tunnels between routers, running IPv6 additions to commands for diagnostics, and displaying IPv6 name service information.

Chapter 16 describes the concepts that are associated with the Solaris implementation of IPv6.

Chapter 17 provides an overview of the approach and the standardized solutions to transitioning from IPv4 to IPv6.

Chapter 19 provides an overview of the new IP Security Architecture that provides protection for IP datagrams.

Chapter 20 provides procedures for implementing IPsec on your network.

Chapter 21 provides an overview and procedures for implementing IKE for use with IPsec.

Chapter 23 provides an overview of the new Mobile IP service that enables the transfer of information to and from mobile computers, such as laptops and wireless communications.

Chapter 24 provides procedures for modifying, adding, deleting, and displaying parameters in the Mobile IP configuration file. This chapter also shows you how to display mobility agent status.

Chapter 25 describes the components that are provided with the Solaris implementation of Mobile IP.

Chapter 27 provides an overview of the new IP Network Multipathing service that allows both load spreading and failover when you have multiple network interface cards connected to the same IP link (for example, Ethernet).

Chapter 28 provides procedures for creating and working with an interface group, configuring test addresses, configuring the `hostname` file, and configuring the multipathing configuration file.

The Glossary provides definitions of key IP Services terms.

How the System Administration Volumes Are Organized

This section provides a list of the topics that are covered by the other six volumes of the *System Administration Guide* set.

System Administration Guide: Basic Administration

- "Solaris Administration Tool Roadmap" in *System Administration Guide: Basic Administration*
- "Working With the Solaris Management Console Tools (Tasks)" in *System Administration Guide: Basic Administration*

- "Managing Users and Groups Topics" in *System Administration Guide: Basic Administration*
- "Managing Server and Client Support Topics" in *System Administration Guide: Basic Administration*
- "Shutting Down and Booting a System Topics" in *System Administration Guide: Basic Administration*
- "Managing Removable Media Topics" in *System Administration Guide: Basic Administration*
- "Managing Software Topics" in *System Administration Guide: Basic Administration*
- "Managing Devices Topics" in *System Administration Guide: Basic Administration*
- "Managing Disks Topics" in *System Administration Guide: Basic Administration*
- "Managing File Systems Topics" in *System Administration Guide: Basic Administration*
- "Backing Up and Restoring Data Topics" in *System Administration Guide: Basic Administration*

System Administration Guide: Advanced Administration

- "Managing Printing Services Topics" in *System Administration Guide: Advanced Administration*
- "Managing Terminals and Modems Topics" in *System Administration Guide: Advanced Administration*
- "Managing System Resources Topics" in *System Administration Guide: Advanced Administration*
- "Managing System Performance Topics" in *System Administration Guide: Advanced Administration*
- "Troubleshooting Solaris Software Topics" in *System Administration Guide: Advanced Administration*

System Administration Guide: Resource Management and Network Services

- "Resource Management and Network Service Overview" in *System Administration Guide: Resource Management and Network Services*
- "Managing Web Cache Servers" in *System Administration Guide: Resource Management and Network Services*

- "Time Related Services" in *System Administration Guide: Resource Management and Network Services*
- "Solaris 9 Resource Manager Topics" in *System Administration Guide: Resource Management and Network Services*
- "Accessing Remote File Systems Topics" in *System Administration Guide: Resource Management and Network Services*
- "SLP Topics" in *System Administration Guide: Resource Management and Network Services*
- "Mail Services Topics" in *System Administration Guide: Resource Management and Network Services*
- "Modem–Related Network Services Topics" in *System Administration Guide: Resource Management and Network Services*
- "Working With Remote Systems Topics" in *System Administration Guide: Resource Management and Network Services*
- "Monitoring Network Services Topics" in *System Administration Guide: Resource Management and Network Services*

System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)

- "Naming and Directory Services (Overview)" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*
- "The Name Service Switch" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*
- "Introduction to DNS (Overview)" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*
- "Network Information Service (NIS): An Overview" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*
- "Introduction to LDAP (Overview/Reference)" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*

System Administration Guide: Security Services

- "Security Services Overview" in *System Administration Guide: Security Services*
- "Using Authentication Services (Tasks)" in *System Administration Guide: Security Services*
- "Using Secure Shell (Tasks)" in *System Administration Guide: Security Services*
- "Secure Shell Administration" in *System Administration Guide: Security Services*

- "Introduction to SEAM" in *System Administration Guide: Security Services*
- "Managing System Security Topics" in *System Administration Guide: Security Services*
- "Role-Based Access Control (Overview)" in *System Administration Guide: Security Services*
- "Using Automated Security Enhancement Tool (Tasks)" in *System Administration Guide: Security Services*
- "Auditing Topics" in *System Administration Guide: Security Services*
-

System Administration Guide: Naming and Directory Services: (FNS and NIS+)

- "The Name Service Switch" in *System Administration Guide: Naming and Directory Services (FNS and NIS+)*
- "NIS+: An Introduction" in *System Administration Guide: Naming and Directory Services (FNS and NIS+)*
- "Federated Naming Service (FNS)" in *System Administration Guide: Naming and Directory Services (FNS and NIS+)*
- "Transitioning from NIS to NIS+" in *System Administration Guide: Naming and Directory Services (FNS and NIS+)*
- "Transitioning From NIS+ to LDAP" in *System Administration Guide: Naming and Directory Services (FNS and NIS+)*
- "Error Messages" in *System Administration Guide: Naming and Directory Services (FNS and NIS+)*

Related Books

The following documentation is referred to in this book.

- Krol, Ed. *The Whole Internet User's Guide and Catalog*. O' Reilly & Associates, Inc., 1993.
- Stevens, W. Richard. *TCP/IP Illustrated, Volume 1, The Protocols*. Addison Wesley, 1994.
- Perkins, Charles E. *Mobile IP Design Principles and Practices*. Massachusetts, 1998, Addison-Wesley Publishing Company.

- *RFC 2002* from the Internet Engineering Task Force (IETF). Available online at [http://ietf.org/rfc.html].
- Solomon, James D. *Mobile IP: The Internet Unplugged*. New Jersey, 1998, Prentice-Hall, Inc.

Accessing Sun Documentation Online

The docs.sun.com^{SM} Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

Typographic Conventions

The following table describes the typographic changes used in this book.

TABLE P–1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type **`rm`** *filename*.
AaBbCc123	Book titles, new words, or terms, or words to be emphasized.	Read Chapter 6 in *User's Guide*. These are called *class* options. You must be *root* to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

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

CHAPTER 1

TCP/IP Topics

Chapter 2	Provides overview information for TCP/IP
Chapter 3	Provides instructions for planning a TCP/IP network
Chapter 4	Provides step-by-step instructions for setting up and troubleshooting a TCP/IP network
Chapter 5	Provides reference information for TCP/IP

CHAPTER 2

TCP/IP (Overview)

This chapter introduces the Solaris implementation of the TCP/IP network protocol suite. The information is intended for network administrators who are unfamiliar with TCP/IP. If you are an experienced TCP/IP network administrator, consider reading chapters that cover the tasks that you want to perform.

This chapter contains the following information:

Introducing the Internet Protocol Suite

This section presents an in-depth introduction to the protocols that compose TCP/IP. Although the information is conceptual, you should learn the names of the protocols and what each does.

TCP/IP is the abbreviation that is commonly used for the set of network protocols that compose the *Internet Protocol suite*. Many texts use the term "Internet" to describe both the protocol suite and the global wide area network. In this book, the "TCP/IP" refers specifically to the Internet protocol suite. "Internet" refers to the wide area network and the bodies that govern the Internet.

To interconnect your TCP/IP network with other networks, you must obtain a unique IP network number. At the time of this writing, the InterNIC organization assigns IP network numbers.

If hosts on your network are to participate in the Internet domain name system (DNS), you must obtain and register a unique domain name. The InterNIC also handles the

registration of domain names under certain top-level domains such as .com (commercial), .edu (education), and .gov (government). Chapter 3 contains more information about the InterNIC. For more information on DNS, refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

Protocol Layers and the OSI Model

Most network protocol suites are structured as a series of layers, sometimes referred to collectively as a *protocol stack*. Each layer is designed for a specific purpose. Each layer exists on both the sending and receiving hosts. A specific layer on one machine sends or receives exactly the same object that another machine's *peer process* sends or receives. These activities occur independently from activities in layers above or below the layer under consideration. Effectively, each layer on a host acts independently of other layers on the same machine. Each layer acts in parallel with the same layer on other hosts.

OSI Reference Model

Most network protocol suites are viewed as structured in layers. The International Organization for Standardization (ISO) designed the Open Systems Interconnection (OSI) Reference Model that uses structured layers. The OSI model describes a structure with seven layers for network activities. Each layer associates one or more protocols with the layer. The layers represent data transfer operations common to all types of data transfers among cooperating networks.

The OSI Reference Model lists the protocol layers from the top (layer 7) to the bottom (layer 1). The following table shows the model.

TABLE 2–1 Open Systems Interconnection Reference Model

Layer No.	Layer Name	Description
7	Application	Consists of standard communication services and applications that everyone can use.
6	Presentation	Ensures that information is delivered to the receiving machine in a form that the machine can understand.
5	Session	Manages the connections and terminations between cooperating computers.
4	Transport	Manages the transfer of data. Also assures that the received data are identical to the transmitted data.
3	Network	Manages data addressing and delivery between networks.
2	Data Link	Handles the transfer of data across the network media.

TABLE 2–1 Open Systems Interconnection Reference Model *(Continued)*

Layer No.	Layer Name	Description
1	Physical	Defines the characteristics of the network hardware.

The OSI model defines conceptual operations that are not unique to any particular network protocol suite. For example, the OSI network protocol suite implements all seven layers of the OSI Reference Model. TCP/IP uses some of OSI model layers. TCP/IP also combines other layers. Other network protocols, such as SNA, add an eighth layer.

TCP/IP Protocol Architecture Model

The OSI model describes an idealized network communications with a family of protocols. TCP/IP does not correspond to this model directly. TCP/IP either combines several OSI layers into a single layer, or does not use certain layers at all. The following table shows the layers of the Solaris implementation of TCP/IP. The table lists the layers from the topmost layer (application) to the lowest (physical network).

TABLE 2–2 TCP/IP Protocol Stack

OSI Ref. Layer No.	OSI Layer Equivalent	TCP/IP Layer	TCP/IP Protocol Examples
5,6,7	Application, session, presentation	Application	NFS, NIS+, DNS, `telnet`, `ftp`, `rlogin`, `rsh`, `rcp`, RIP, RDISC, SNMP, and others
4	Transport	Transport	TCP, UDP
3	Network	Internet	IP, ARP, ICMP
2	Data link	Data link	PPP, IEEE 802.2
1	Physical	Physical network	Ethernet (IEEE 802.3) Token Ring, RS-232, others

The table shows the TCP/IP protocol layers. Also shown are the OSI Model equivalents with examples of the protocols that are available at each level of the TCP/IP protocol stack. Each host that is involved in a communication transaction runs a unique implementation of the protocol stack.

Physical Network Layer

The physical network layer specifies the characteristics of the hardware to be used for the network. For example, physical network layer specifies the physical characteristics

of the communications media. The physical layer of TCP/IP describes hardware standards such as IEEE 802.3, the specification for Ethernet network media, and RS-232, the specification for standard pin connectors.

Data-Link Layer

The data-link layer identifies the network protocol type of the packet, in this instance TCP/IP. The data-link layer also provides error control and "framing." Examples of data-link layer protocols are Ethernet IEEE 802.2 framing and Point-to-Point Protocol (PPP) framing.

Internet Layer

This layer, also known as the network layer, accepts and delivers packets for the network. This layer includes the powerful Internet Protocol (IP), the Address Resolution Protocol (ARP), and the Internet Control Message Protocol (ICMP).

IP Protocol

The IP protocol and its associated routing protocols are possibly the most significant of the entire TCP/IP suite. IP is responsible for the following:

- *IP addressing* – The IP addressing conventions are part of the IP protocol. Chapter 3 describes IPv4 addressing in detail and Chapter 14 describes IPv6 addressing in detail.
- *Host-to-host communications* – IP determines the path a packet must take, based on the receiving host's IP address.
- *Packet formatting* – IP assembles packets into units that are known as *IP datagrams*. Datagrams are fully described in "Internet Layer" on page 42.
- *Fragmentation* – If a packet is too large for transmission over the network media, IP on the sending host breaks the packet into smaller fragments. IP on the receiving host then reconstructs the fragments into the original packet.

Previous releases of the Solaris operating environment implement version 4 of the Internet Protocol, which is abbreviated as IPv4. However, because of the rapid growth of the Internet, a new Internet Protocol was created. The new protocol increases address space. This new version, known as version 6, is abbreviated as IPv6. The Solaris operating environment supports both versions, which are described in this book. To avoid confusion when addressing the Internet Protocol, one of the following conventions is used:

- When the term IP is used in a description, the description applies to both IPv4 and IPv6.

- When the term IPv4 is used in a description, the description applies only to IPv4.
- When the term IPv6 is used in a description, the description applies only to IPv6.

ARP Protocol

The Address Resolution Protocol (ARP) conceptually exists between the data-link and Internet layers. ARP assists IP in directing datagrams to the appropriate receiving host by mapping Ethernet addresses (48 bits long) to known IP addresses (32 bits long).

ICMP Protocol

Internet Control Message Protocol (ICMP) detects and reports network error conditions. ICMP reports on the following:

- Dropped packets – Packets that arrive too fast to be processed
- Connectivity failure – A destination host that cannot be reached)
- Redirection – Redirecting a sending host to use another router

The "`ping` Command" on page 75 contains more information on the operating system commands that use ICMP for error detection.

Transport Layer

The TCP/IP transport layer protocols ensure that packets arrive in sequence and without error, by swapping acknowledgments of data reception, and retransmitting lost packets. This type of communication is known as "end-to-end." Transport layer protocols at this level are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

TCP Protocol

TCP enables applications to communicate with each other as though connected by a physical circuit. TCP sends data in a form that appears to be transmitted in a character-by-character fashion, rather than as discrete packets. This transmission consists of a starting point, which opens the connection, the entire transmission in byte order, and an ending point, which closes the connection.

TCP attaches a header onto the transmitted data. This header contains a large number of parameters that help processes on the sending machine connect to peer processes on the receiving machine.

TCP confirms that a packet has reached its destination by establishing an end-to-end connection between sending and receiving hosts. TCP is therefore considered a "reliable, connection-oriented" protocol.

UDP Protocol

UDP, the other transport layer protocol, provides datagram delivery service. UDP does not verify connections between receiving and sending hosts. Because UDP eliminates the processes of establishing and verifying connections, applications that send small amounts of data use UDP rather than TCP.

Application Layer

The application layer defines standard Internet services and network applications that anyone can use. These services work with the transport layer to send and receive data. Many application layer protocols exist. The following list shows examples of application layer protocols:

- Standard TCP/IP services such as the `ftp`, `tftp`, and `telnet` commands
- UNIX "r" commands, such as `rlogin` and `rsh`
- Name services, such as NIS+ and domain name system (DNS)
- File services, such as the NFS service
- Simple Network Management Protocol (SNMP), which enables network management
- RIP and RDISC routing protocols

Standard TCP/IP Services

- *FTP and Anonymous FTP* – The File Transfer Protocol (FTP) transfers files to and from a remote network. The protocol includes the `ftp` command (local machine) and the `in.ftpd` daemon (remote machine). FTP enables a user to specify the name of the remote host and file transfer command options on the local host's command line. The `in.ftpd` daemon on the remote host then handles the requests from the local host. Unlike `rcp`, `ftp` works even when the remote computer does not run a UNIX-based operating system. A user must log in to the remote computer to make an `ftp` connection unless the remote computer has been configured to allow anonymous FTP.

  You can now obtain an enormous amount of materials from *anonymous FTP servers* that are connected to the Internet. Universities and other institutions set up these servers to offer software, research papers, and other information to the public domain. When you log in to this type of server, you use the login name `anonymous`, hence the term "anonymous FTP servers."

Using anonymous FTP and setting up anonymous FTP servers is outside the scope of this manual. However, many books, such as *The Whole Internet User's Guide & Catalog*, discuss anonymous FTP in detail. Instructions for using FTP to reach standard machines are in *System Administration Guide: Resource Management and Network Services*. The `ftp`(1) man page describes all `ftp` command options that are invoked through the command interpreter. The `ftpd`(1M) man page describes the services that are provided by the daemon `in.ftpd`.

- *Telnet* – The Telnet protocol enables terminals and terminal-oriented processes to communicate on a network that runs TCP/IP. This protocol is implemented as the program `telnet` (on local machines) and the daemon `in.telnetd` (on remote machines). Telnet provides a user interface through which two hosts can communicate on a character-by-character or line-by-line basis. The application includes a set of commands that are fully documented in the `telnet`(1) man page.
- *TFTP* – The Trivial File Transfer Protocol (`tftp`) provides functions that are similar to `ftp`, but the protocol does not establish `ftp`'s interactive connection. As a result, users cannot list the contents of a directory or change directories. A user must know the full name of the file to be copied. The `telnet`(1) man page describes the `tftp` command set.

UNIX "r" Commands

The UNIX "r" commands enable users to issue commands on their local machines that run on the remote host. These commands include the following:

- `rcp`
- `rlogin`
- `rsh`

Instructions for using these commands are in `rcp`(1), `rlogin`(1), and `rsh`(1) man pages.

Name Services

The Solaris operating environment provides the following naming services:

- *DNS* – The domain name system (DNS) is the naming service provided by the Internet for TCP/IP networks. DNS provides host names to the IP address service. DNS also serves as a database for mail administration. For a complete description of this service, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*. See also the `resolver`(3RESOLV) man page.
- `/etc` *files* – The original host-based UNIX™ naming system was developed for standalone UNIX™ machines and then adapted for network use. Many old UNIX™ operating systems and machines still use this system, but it is not well suited for large complex networks.

- *NIS* – Network Information Service (NIS) was developed independently of DNS and has a slightly different focus. Whereas DNS focuses on making communication simpler by using machine names instead of numerical IP addresses, NIS focuses on making network administration more manageable by providing centralized control over a variety of network information. NIS stores information about machine names and addresses, users, the network itself, and network services. NIS namespace information is stored in NIS maps. For more information on NIS Architecture and NIS Administration, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.
- *NIS+* – NIS+ provides centralized control over network administration services, such as mapping host names to IP and Ethernet addresses, verifying passwords, and so on. See *System Administration Guide: Naming and Directory Services (FNS and NIS+)*.
- *FNS* – Federated Naming Service (FNS), supports the use of different autonomous naming systems in a single Solaris operating environment. FNS allows you to use a single, simple naming system interface for all of the different name services on your network. FNS conforms to the X/Open federated naming (XFN) specification. FNS is not a replacement for NIS+, NIS, DNS, or `/etc` files. Rather, FNS is implemented on top of these services and allows you to use a set of common names with desktop applications. See *System Administration Guide: Naming and Directory Services (FNS and NIS+)*.

Directory Service

The Solaris operating environment supports LDAP (Lightweight Directory Access Protocol) in conjunction with the iPlanet Directory Server 5.x, as well as other LDAP Directory Servers. The distinction between a Naming Service and a Directory Service is in the differing extent of functionality. A directory service provides the same functionality of a naming service, but provides additional functionalities as well. See *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

File Services

The NFS application layer protocol provides file services for the Solaris operating environment. You can find complete information about the NFS service in *System Administration Guide: Resource Management and Network Services*.

Network Administration

The Simple Network Management Protocol (SNMP) enables you to view the layout of your network and view the status of key machines. SNMP also enables you to obtain complex network statistics from software that is based on a graphical user interface.

Many companies offer network management packages that implement SNMP. SunNet Manager™ software is an example.

Routing Protocols

The Routing Information Protocol (RIP) and the Router Discovery Protocol (RDISC) are two routing protocols for TCP/IP networks. They are described in "Routing Protocols" on page 106.

How the TCP/IP Protocols Handle Data Communications

When a user issues a command that uses a TCP/IP application layer protocol, a series of events is initiated. The user's command or message passes through the TCP/IP protocol stack on the local machine. Then the command or message passes across the network media to the protocols on the recipient. The protocols at each layer on the sending host add information to the original data.

Protocols on each layer of the sending host also interact with their peers on the receiving host. Figure 2–1 shows this interaction.

Data Encapsulation and the TCP/IP Protocol Stack

The packet is the basic unit of information that is transferred across a network. The packet consists, at a minimum, of a header with the sending and receiving hosts' addresses, and a body with the data to be transferred. As the packet travels through the TCP/IP protocol stack, the protocols at each layer either add or remove fields from the basic header. When a protocol on the sending host adds data to the packet header, the process is called *data encapsulation*. Moreover, each layer has a different term for the altered packet, as shown in the following figure.

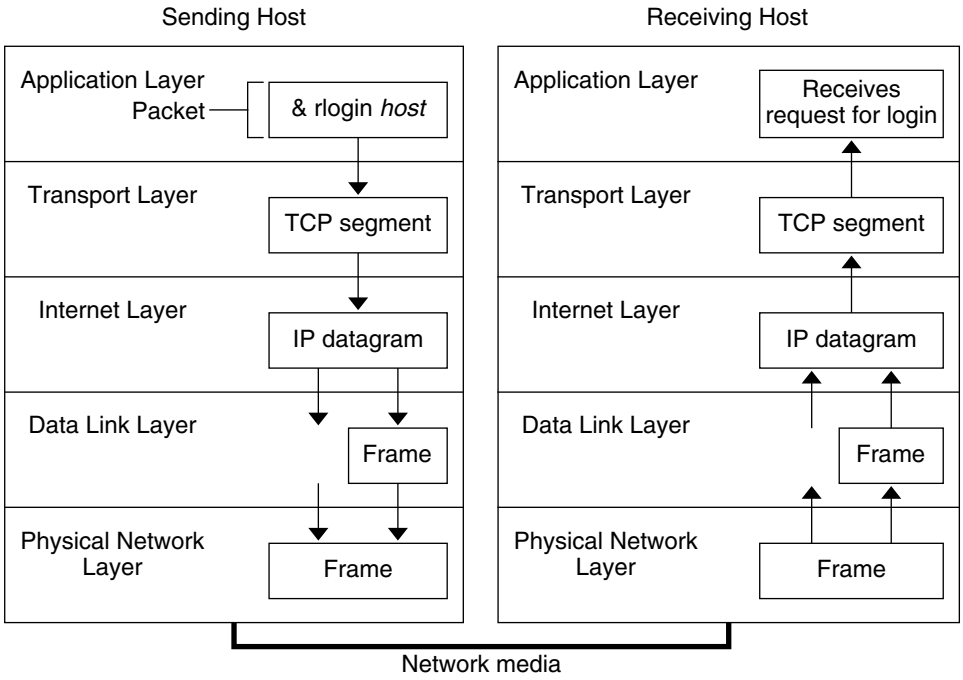


FIGURE 2–1 How a Packet Travels Through the TCP/IP Stack

This section summarizes the life cycle of a packet. The life cycle starts when you issue a command or send a message. The life cycle finishes when the appropriate application on the receiving host receives the packet.

Application Layer—User Initiates Communication

The packet's history begins when a user on one host sends a message or issues a command that must access a remote host. The application protocol formats the packet so that the appropriate transport layer protocol, TCP or UDP, can handle the packet.

Suppose the user issues an `rlogin` command to log in to the remote host, as shown in Figure 2–1. The `rlogin` command uses the TCP transport layer protocol. TCP expects to receive data in the form of a stream of bytes that contain the information in the command. Therefore, `rlogin` sends this data as a TCP stream.

Not all application layer protocols use TCP, however. Suppose a user wants to mount a file system on a remote host, thus initiating the NIS+ application layer protocol. NIS+ uses the UDP transport layer protocol. Therefore, the packet that contains the command must be formatted in a manner that UDP expects. This type of packet is referred to as a *message*.

Transport Layer—Data Encapsulation Begins

When the data arrives at the transport layer, the protocols at the layer start the process of data encapsulation. The end result depends on whether TCP or UDP handles the information.

TCP Segmentation

TCP is often called a "connection-oriented" protocol because TCP ensures the successful delivery of data to the receiving host. Figure 2–1 shows how the TCP protocol receives the stream from the `rlogin` command. TCP divides the data that is received from the application layer into segments and attaches a header to each segment.

Segment headers contain sender and recipient ports, segment ordering information, and a data field that is known as a *checksum*. The TCP protocols on both hosts use the checksum data to determine if the data transfers without error.

Establishing a TCP Connection

TCP uses segments to determine whether the receiving host is ready to receive the data. When the sending TCP wants to establish connections, TCP sends a segment that is called a SYN to the TCP protocol on the receiving host. The receiving TCP returns a segment that is called an ACK to acknowledge the successful receipt of the segment. The sending TCP sends another ACK segment, then proceeds to send the data. This exchange of control information is referred to as a *three-way handshake*.

UDP Packets

UDP is a "connectionless" protocol. Unlike TCP, UDP does not check that data arrived at the receiving host. Instead, UDP formats the message that is received from the application layer into *UDP packets*. UDP attaches a header to each packet. The header contains the sending and receiving host ports, a field with the length of the packet, and a checksum.

The sending UDP process attempts to send the packet to its peer UDP process on the receiving host. The application layer determines whether the receiving UDP process acknowledges the reception of the packet. UDP requires no notification of receipt. UDP does not use the three-way handshake.

Internet Layer

As shown in Figure 2–1, both TCP and UDP pass their segments and packets down to the Internet layer, where the IP protocol handles the segments and packets. IP prepares them for delivery by formatting them into units called IP datagrams. IP then determines the IP addresses for the datagrams, so that they can be delivered effectively to the receiving host.

IP Datagrams

IP attaches an *IP header* to the segment or packet's header in addition to the information that is added by TCP or UDP. Information in the IP header includes the IP addresses of the sending and receiving hosts, datagram length, and datagram sequence order. This information is provided if the datagram exceeds the allowable byte size for network packets and must be fragmented.

Data-Link Layer—Framing Takes Place

Data-link layer protocols, such as PPP, format the IP datagram into a *frame*. These protocols attach a third header and a footer to "frame" the datagram. The frame header includes a *cyclic redundancy check* (CRC) field that checks for errors as the frame travels over the network media. Then the data-link layer passes the frame to the physical layer.

Physical Network Layer—Preparing the Frame for Transmission

The physical network layer on the sending host receives the frames and converts the IP addresses into the hardware addresses appropriate to the network media. The physical network layer then sends the frame out over the network media.

How the Receiving Host Handles the Packet

When the packet arrives on the receiving host, the packet travels through the TCP/IP protocol stack in the reverse order from that which the packet travels on the sender. Figure 2–1 illustrates this path. Moreover, each protocol on the receiving host strips off header information that is attached to the packet by its peer on the sending host. The following process occurs:

1. The physical network layer receives the packet in its frame form. The physical network layer computes the CRC of the packet, then sends the frame to the data link layer.

2. The data-link layer verifies that the CRC for the frame is correct and strips off the frame header and CRC. Finally, the data link protocol sends the frame to the Internet layer.
3. The Internet layer reads information in the header to identify the transmission. Then Internet layer determines if the packet is a fragment. If the transmission is fragmented, IP reassembles the fragments into the original datagram. IP then strips off the IP header and passes the datagram on to transport layer protocols.
4. The transport layer (TCP and UDP) reads the header to determine which application layer protocol must receive the data. Then TCP or UDP strips off its related header. TCP or UDP sends the message or stream up to the receiving application.
5. The application layer receives the message. The application layer then performs the operation that the sending host requested.

TCP/IP Internal Trace Support

TCP/IP provides internal trace support by logging TCP communication when an RST packet terminates a connection. When an RST packet is transmitted or received, information on as many as 10 packets, which were just transmitted is logged with the connection information.

Finding Out More About TCP/IP and the Internet

Information about TCP/IP and the Internet is widely available. If you require specific information that is not covered in this text, you can probably find what you need in the sources cited next.

Computer Books

Many books about TCP/IP and the Internet are available from your local library or computer bookstore. The following three books are recommended:

- Craig Hunt. *TCP/IP Network Administration* – This book contains some theory and much practical information for managing a heterogeneous TCP/IP network.
- W. Richard Stevens. *TCP/IP Illustrated, Volume I* – This book is an in-depth explanation of the TCP/IP protocols. This book is ideal for network administrators

who require a technical background in TCP/IP and for network programmers.

- Ed Krol. *The Whole Internet User's Guide & Catalog* – This book is ideal for anyone interested in using the many tools that are available for retrieving information over the Internet.

RFCs and FYIs

The Internet Architecture Board (IAB) must approve all RFCs before they are placed in the public domain. Typically, the information in RFCs is designed for developers and other highly technical readers.

Generally, for your information (FYI) documents appear as a subset of the RFCs. FYIs contain information that does not deal with Internet standards. FYIs contain Internet information of a more general nature. For example, FYI documents include a bibliography that list introductory TCP/IP books and papers. FYI documents provide an exhaustive compendium of Internet-related software tools. Finally, FYI documents include a glossary of Internet and general networking terms.

You'll find references to relevant RFCs throughout this guide and other books in the Solaris System Administrator set.

CHAPTER 3

Planning Your TCP/IP Network (Task)

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

This chapter contains the following information:

Designing the Network

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

- Number of host machines your network can support
- Type of network media to use: Ethernet, token ring, FDDI, and so on
- Network topology, the layout and connections of the network hardware
- Types of hosts the network supports: standalone and dataless

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

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

Network Planning Tasks

After you complete your hardware plan, you are ready to begin network planning, from the software perspective.

As part of the planning process, you must do the following:

1. Obtain a network number and, if applicable, register your network domain with the InterNIC.
2. Devise an IP addressing scheme for your hosts, after you receive your IP network number.
3. Create a list that contains the IP addresses and host names of all machines on your network. Use the list to build network databases.
4. Determine which name service to use on your network: NIS, NIS+, DNS, or the network databases in the local `/etc` directory.
5. Establish administrative subdivisions, if appropriate for your network.
6. Determine if your network is large enough to require routers, and, if appropriate, create a network topology that supports them.
7. Set up subnets, if appropriate, for your network.

The remainder of this chapter explains how to plan your network.

Setting Up an IP Address Scheme

The number of machines that you expect to support affects how you configure your network. Your organization might require a small network of several dozen standalone machines that are located on one floor of a single building. Alternatively, you might need to set up a network with more than 1000 hosts in several buildings. This arrangement can require you to further divide your network into subdivisions that are called *subnets*. The size of your prospective network affects the following factors:

- Network class that you apply for
- Network number that you receive
- IP addressing scheme that you use for your network

Administering Network Numbers

If your organization has been assigned more than one network number, or uses subnets, appoint a centralized authority within your organization to assign network numbers. That authority should maintain control of a pool of assigned network numbers, and assign network, subnet, and host numbers as required. To prevent problems, ensure that duplicate or random network numbers do not exist in your organization. If you are planning to transition to IPv6, see Chapter 17.

Designing Your IPv4 Addressing Scheme

After you receive your network number, you can then plan how to assign the host parts of the IPv4 address.

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

TABLE 3–1 Division of IPv4 Address Space

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

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

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

TABLE 3–2 Range of Available Numbers

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

How IP Addresses Apply to Network Interfaces

In order to connect to the network, a computer must have at least one network interface. Each network interface must have its own unique IP address. The IP address that you give to a host is assigned to its network interface, sometimes referred to as the *primary network interface*. If you add a second network interface to a machine, the machine must have its own unique IP number. When you add a second network interface, the machine changes to a router. See "Configuring Routers" on page 68 for an explanation. If you add a second network interface to a host and you disable routing, the host is then considered a multihomed host.

Each network interface has a device name, device driver, and an associated device file in the `/devices` directory. The network interface might have a device name, such as `le0` or `smc0`, device names for two commonly used Ethernet interfaces.

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

Naming Entities on Your Network

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

The TCP/IP protocols locate a machine on a network by using its IP address. However, if you use a recognizable name, then you can identify the machine easily.

Therefore, the TCP/IP protocols (and the Solaris operating environment) require both the IP address and the host name to uniquely identify a machine.

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 name identifies a *domain*.

Administering Host Names

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

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

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

Selecting a Name Service

The Solaris operating environment gives you the option of using four types of name services: local files, NIS, 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. The Solaris operating environment also gives you the option of using the LDAP directory service.

Network Databases

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

The configuration of the network databases is a critical. Therefore, you need to decide which name service to use as part of the network planning process. Moreover, the decision to use name services also affects whether you organize your network into an

administrative domain. “Network Databases and `nsswitch.conf` File” on page 97 has detailed information on the set of network databases.

Using NIS, NIS+, or DNS for Name Service

The NIS, NIS+, or DNS name services maintain network databases on several servers on the network. *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* and *System Administration Guide: Naming and Directory Services (FNS and NIS+)* describe these name services. These guides also explain how to configure the databases. In addition, the guides explain the “namespace” and “administrative domain” concepts in detail.

Using Local Files for Name Service

If you do not implement NIS, NIS+, or DNS, the network uses *local files* to provide 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 NIS, NIS+, or the DNS name services, 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 should register your domain name with the InterNIC.

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

The following examples show top-level domains:

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

The name that identifies your organization is one that you select, with the provision that the name is unique.

Administrative Subdivisions

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

- How large is the network?

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

- Do users on the network have similar needs?

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

Registering Your Network

Before you assign IP addresses to the machines on your Solaris network, you must obtain a network number from the InterNIC. Moreover, if you are using administrative domains, you should register them with the InterNIC.

InterNIC and InterNIC Registration Services

The InterNIC was created in 1993 to act as a central body for Internet information, such as:

- The Internet's policies
- Accessing the Internet, including training services

- Resources available to Internet users, such as anonymous FTP servers, Usenet user groups, and so on

The InterNIC also includes the InterNIC Registration Services, the organization with which you register your TCP/IP network. The InterNIC Registration Services provide templates for obtaining a network number and for registering your domain. When you register, remember the following points:

- The InterNIC assigns network numbers.

  Note – Do not arbitrarily assign network numbers to your network, even if you are not attaching the network to other TCP/IP networks.

  InterNIC does not assign subnet numbers. Rather, subnet numbers are composed partly of the assigned network number and numbers that you define, as explained in "What Is Subnetting?" on page 94.
- You—not InterNIC—determine the domain name for your network and then register the domain name with the InterNIC.

How to Contact the InterNIC

You can reach the InterNIC Registration Services by the following forms of communication:

- Mail

  Write to the following address:

  ```
  Network Solutions
  Attn: InterNIC Registration Services
  505 Huntmar Park Drive
  Herndon, Virginia 22070
  ```

- Telephone

  The phone number is 1–703-742-4777. Phone service is available from 7 a.m. to 7 p.m. Eastern Standard Time. The domestic toll free phone number is 1–800–779–1710.

Adding Routers

Recall that in TCP/IP, two types of entities exist on a network: hosts and routers. All networks must have hosts, while not all networks require routers. The physical

topology of the network determines if you need routers. This section introduces the concepts of network topology and routing, important when you decide to add another network to your existing network environment.

Network Topology

Network topology describes how networks fit together. Routers are the entities that connect networks to each other. From a TCP/IP perspective, a router is any machine that has two or more network interfaces. However, the machine cannot function as a router until properly configured, as described in "Configuring Routers" on page 68.

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

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

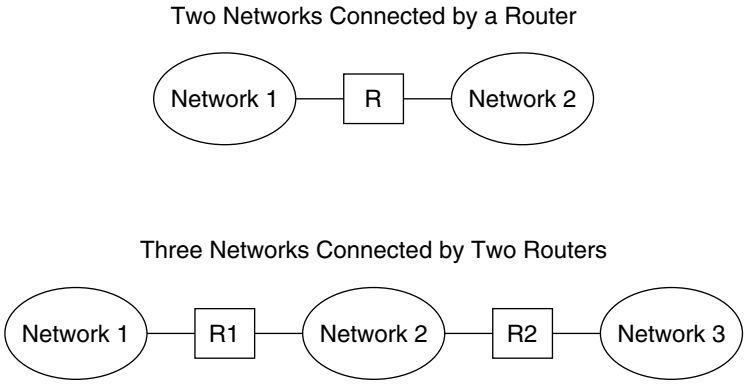


FIGURE 3–1 Basic Network Topology

Routers join networks into internetworks. Routers also route packets between networks that are based on the addresses of the destination network. As internetworks grow more complex, each router must make more and more decisions about the packet destinations.

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

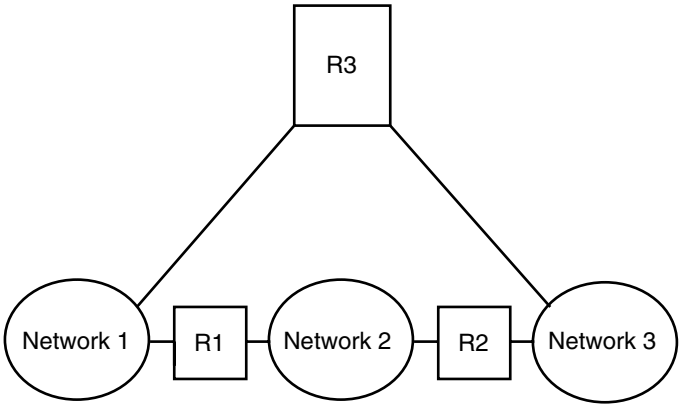


FIGURE 3–2 Providing an Additional Path Between Networks

How Routers Transfer Packets

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

Routers maintain routing information in *routing tables*. These tables contain the IP address of the hosts and routers on the networks to which the router is connected. The tables also contain pointers to these networks. When a router receives a packet, the router consults its routing table to see if the table lists the destination address in the header. If the table does not contain the destination address, the router forwards the packet to another router that is listed in its routing table. Refer to "Configuring Routers" on page 68 for detailed information on routers.

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

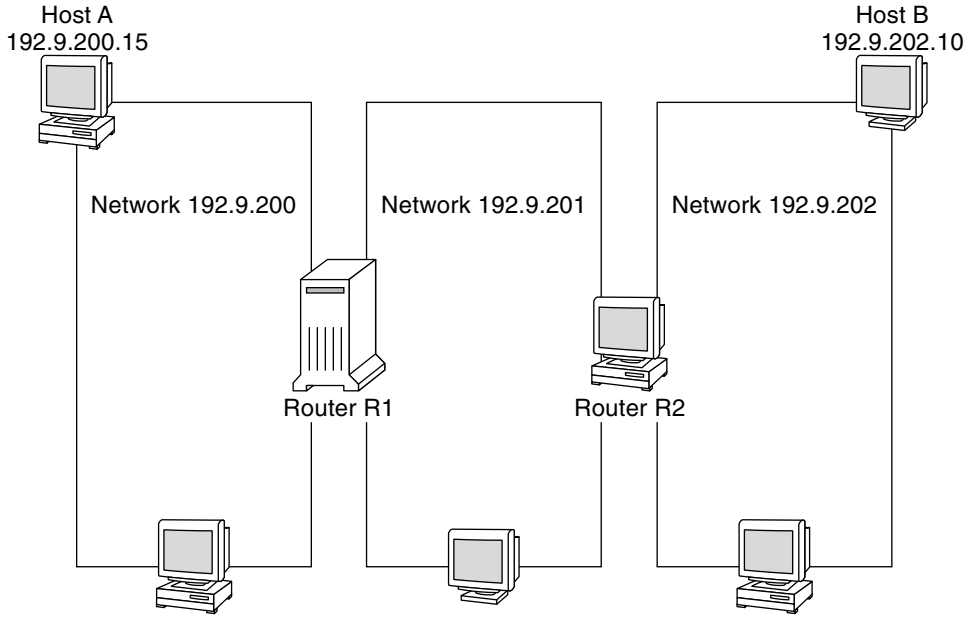


FIGURE 3–3 Three Interconnected Networks

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

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

CHAPTER 4

Administering TCP/IP (Task)

TCP/IP administration involves the procedures that you use to configure your network. First you assemble the hardware. Then you configure TCP/IP. This chapter explains how to configure TCP/IP. This chapter also addresses how to troubleshoot TCP/IP problems.

This chapter contains the following information:

Before You Configure TCP/IP

Before you configure TCP/IP, complete the tasks that are listed in the following table.

TABLE 4–1 Before You Configure TCP/IP Task Map

Description	For Instructions, Go To ...
Design the network topology.	See "Network Topology" on page 53.
Obtain a network number from your Internet addressing authority.	See "Designing Your IPv4 Addressing Scheme" on page 47.
Assemble the network hardware depending on the network topology. Assure that the hardware is functioning properly.	See the hardware manuals and "Network Topology" on page 53.
Run configuration software that is required by network interfaces and routers, if applicable.	See "Adding Routers" on page 52 and "Configuring Routers" on page 68 for information on routers.
Plan the IP addressing scheme for the network. If applicable, include subnet addressing.	See "Designing Your IPv4 Addressing Scheme" on page 47 and "IPv6 Addressing" on page 282.
Assign IP numbers and host names to all machines in the network.	See "Designing Your IPv4 Addressing Scheme" on page 47 and "IPv6 Addressing" on page 282.
Determine which name service your network uses: NIS, NIS+, DNS, or local files.	See *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* and *System Administration Guide: Naming and Directory Services (FNS and NIS+)*.
Select domain names for your network, if applicable.	See *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* and *System Administration Guide: Naming and Directory Services (FNS and NIS+)*.
Install the operating system on at least one machine on the network.	See *Solaris 9 Installation Guide*.

Determining Host Configuration Modes

As a network administrator, you configure TCP/IP to run on hosts and routers (if applicable). You can configure these machines to obtain configuration information

from files on the local machine or from files that are located on other machines on the network. You need the following configuration information:

- Host name of a machine
- IP address of the machine
- Domain name to which the machine belongs
- Default router
- Netmask in use on the machine's network (if applicable)

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

Machines That Should Run in *Local Files* Mode

To run in *local files* mode, a machine must have local copies of the TCP/IP configuration files. "TCP/IP Configuration Files" on page 87 describes the files. The machine should have its own disk, though this recommendation is not strictly necessary.

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

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

Additionally, routers should run in *local files* mode.

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

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

Network Configuration Servers

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

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

Network configuration servers can also function as NFS file servers.

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

Machines That Are Network Clients

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

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

You can configure *network client* mode on all types of computers. For example, you can configure *network client* mode on fully standalone systems or dataless machines.

Mixed Configurations

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

Sample Network

The following figure shows the hosts of a fictitious network with the network number 192.9.200. The network has one network configuration server, the machine `sahara`. Machines `tenere` and `nubian` have their own disks and run in *local files* mode. Machine `faiyum` also has a disk, but this machine operates in *network client* mode.

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

Most examples in this chapter use the network that is shown in the following figure.

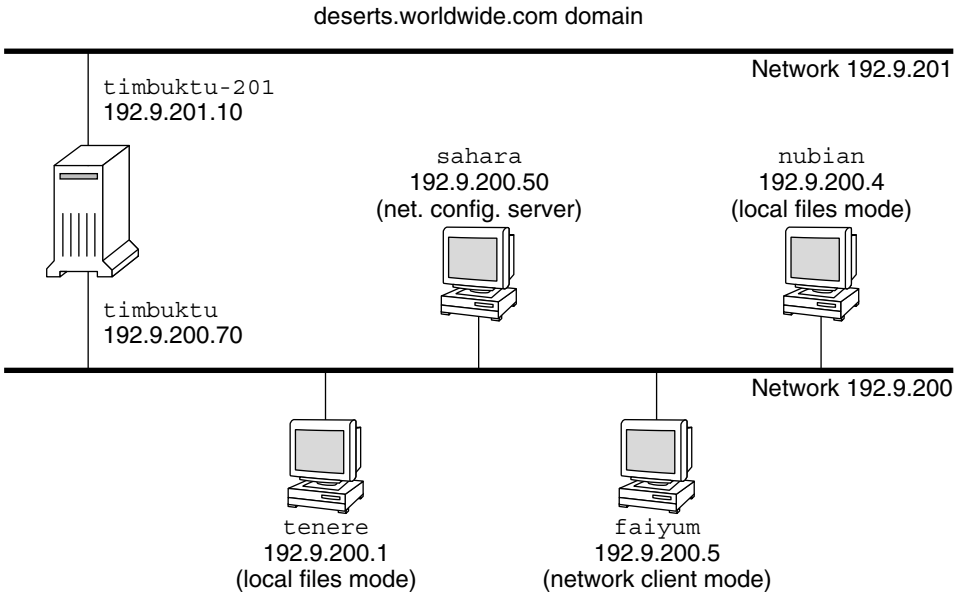


FIGURE 4–1 Hosts in a Sample Network

Adding a Subnet to a Network Task Map

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

TABLE 4–2 Adding a Subnet to a Network Task Map

Description	For Instructions, Go To ...
1. Decide on the new subnet topology, including considerations for routers and locations of hosts on the subnets.	"Adding Routers" on page 52, "What Is Subnetting?" on page 94, and "Network Classes" on page 109.
2. Assign all subnet and host addresses.	"Setting Up an IP Address Scheme" on page 46 and "Parts of the IPv4 Address" on page 108.
3. Modify the `/etc/inet/netmasks` file, if you are manually configuring TCP/IP, or supply the netmask to the Solaris installation program.	"`netmasks` Database" on page 93 and "Creating the Network Mask for IPv4 Addresses" on page 94.
4. Modify the `/etc/inet/hosts` and `/etc/inet/ipnodes` files on all hosts to reflect the new host addresses.	"`hosts` Database" on page 90 and "`ipnodes` Database" on page 93.
5. Reboot all machines.	

Network Configuration Procedures

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

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

The parameters that are supplied during network configuration follow:

- IP address of each network interface on every machine.
- Host names of each machine on the network. You can type the host name in a local file or a name service database.
- NIS, NIS+, or DNS domain name in which the machine resides, if applicable.
- Default router addresses. You supply this information if you have a simple network topology with only one router attached to each network. You also supply this information if your routers do not run routing protocols such as the Router Discovery Server Protocol (RDISC) or the Router Information Protocol (RIP). See "Routing Protocols" on page 106 for more information about these protocols.
- Subnet mask (required only for networks with subnets).

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

Network Configuration Task Map

TABLE 4–3 Network Configuration Task Map

Task	Description	For Instructions, Go To ...
Configure a host for local files mode	Involves editing the `nodename`, `hostname`, `hosts`, `defaultdomain`, `defaultrouter`, and `netmasks` files	"How to Configure a Host for Local Files Mode" on page 63
Set up a network configuration server	Involves turning on the `in.tftp` daemon, and editing the `inetd.conf`, `hosts`, `ethers`, and `bootparams` files	"How to Set Up a Network Configuration Server" on page 64
Configure a host for network client mode	Involves creating the `hostname` file, editing the `hosts` file, and deleting the `nodename` and `defaultdomain` files, if they exist	"How to Configure Hosts for Network Client Mode" on page 65
Specify a router for the network client	Involves editing the `defaultrouter` and `hosts` files	"How to Specify a Router for the Network Client" on page 66

▼ How to Configure a Host for Local Files Mode

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

1. **Become superuser and change directories to `/etc`.**
2. **Type the host name of the machine in the file `/etc/nodename`.**
   For example, if the name of the host is `tenere`, type `tenere` in the file.
3. **Create a file that is named `/etc/hostname.`*interface* for each network interface.**
   The Solaris installation program automatically creates this file for the primary network interface. Refer to "`/etc/hostname.`*interface* File" on page 88 for details. If you are using IPv6, see "IPv6 Network Interface Configuration File" on page 324.
4. **Type either the interface IP address or the interface name in each `/etc/hostname.`*interface* file.**

For example, create a file that is named `hostname.ie1`, and type either the IP address of the host's interface or the host's name.

5. **Edit the `/etc/inet/hosts` file to add the following:**

   a. **IP addresses that you have assigned to any additional network interfaces in the local machine, along with the corresponding host name for each interface.**

      The Solaris installation program has already created entries for the primary network interface and loopback address.

   b. **IP address or addresses of the file server, if the `/usr` file system is NFS mounted.**

Note – The Solaris installation program creates the default `/etc/inet/hosts` for the local machine. If the file does not exist, create the file as shown in "`hosts` Database" on page 90. Also, if you are using IPv6, see "`/etc/inet/ipnodes` File" on page 336.

---

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

   For example, suppose host `tenere` was part of the domain `deserts.worldwide.com`. Therefore, you would type `deserts.worldwide.com` in `/etc/defaultdomain`. See "`/etc/defaultdomain` File" on page 89 for more information.

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

   See "`/etc/defaultrouter` File" on page 90 for information about this file.

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

   Additional routing options are available. Refer to the discussion on routing options in "How to Configure Hosts for Network Client Mode" on page 65. You can apply these options to a local files mode configuration.

9. **If your network is subnetted, type the network number and the netmask in the file `/etc/inet/netmasks`.**

   If you have set up an NIS or NIS+ server, you can type `netmask` information in the appropriate database on the server if server and clients are on the same network.

10. **Reboot each machine on the network.**

▼ How to Set Up a Network Configuration Server

1. **Become superuser and change to the root directory of the prospective network configuration server.**

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

```
# mkdir /tftpboot
```

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

3. **Create a symbolic link to the directory.**

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

4. **Enable the `tftp` line in `inetd.conf`.**

Check that the `/etc/inetd.conf` entry reads as follows:

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

This line prevents `inettftpd()` from retrieving any file other than the file that is located in `/tftpboot`.

5. **Edit the `hosts` database, and add the host names and IP addresses for every client on the network.**

6. **Edit the `ethers` database, and create entries for every host on the network to run in network client mode.**

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

See "`bootparams` Database" on page 101. Use the wildcard entry or create an entry for every host that runs in network client mode.

8. **On a command line, type the following command.**

```
# pkill -HUP inetd
```

Information for setting up install servers and boot servers is found in *Solaris 9 Installation Guide*.

Configuring Network Clients

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

▼ How to Configure Hosts for Network Client Mode

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

1. **Become superuser.**

2. **Check the directory for the existence of an `/etc/nodename` file. If such a file exists, delete it.**

Eliminating `/etc/nodename` causes the system to use the `hostconfig` program to obtain the host name, domain name, and router addresses from the network configuration server. See "Network Configuration Procedures" on page 62.

3. **Create the file `/etc/hostname.`*interface*, if this file does not exist.**

   Ensure that the file is empty. An empty `/etc/hostname.`*interface* file causes the system to acquire the IP address from the network configuration server. If you are using IPv6, see "IPv6 Network Interface Configuration File" on page 324.

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

   For more information, see "Loopback Address" on page 91. The file should not contain the IP address and host name for the local machine (primary network interface). If you are using IPv6, see "`/etc/inet/ipnodes` File" on page 336.

5. **Check for the existence of an `/etc/defaultdomain` file. If such a file exists, delete it.**

   The `hostconfig` program sets the domain name automatically. If you are overriding the domain name that is set by `hostconfig`, type the substitute domain name in the file `/etc/defaultdomain`.

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

▼ How to Specify a Router for the Network Client

1. **If only one router is on the network and the network configuration server is to specify its name automatically, ensure that no `/etc/defaultrouter` file exists on the network client.**

2. **To override the name of the default router that is provided by the network configuration server, do the following:**

   a. **Create `/etc/defaultrouter` on the network client.**

   b. **Type the host name and IP address of the machine you have designated as the default router.**

   c. **Add the host name and IP address of the designated default router to the network client's `/etc/inet/hosts`.**

3. **If you have multiple routers on the network, create `/etc/defaultrouter` on the network client, but leave this file empty.**

   Creating `/etc/defaultrouter` and leaving this file empty causes one of the two dynamic routing protocols to run: ICMP Router Discovery Protocol (RDISC), or

Routing Information Protocol (RIP). The system first runs the program `in.rdisc`, which looks for routers that are running the router discovery protocol. If `in.rdisc` finds one such router, `in.rdisc` continues to run and monitors the routers that are running the RDISC protocol.

If the system discovers that routers are not responding to the RDISC protocol, the system uses RIP and runs the `in.routed` daemon to monitor the routers.

Configuring Standard TCP/IP Services

Services such as `telnet`, `ftp`, and `rlogin` are started by the `inetd` daemon, which runs automatically at boot time. Refer to the `inetd`(1M) and `inetd.conf`(4) man pages.

In addition to the service definitions in the `/etc/inetd.conf` file, you can configure `inetd` by using the `/etc/default/inetd` file. For example, you can configure the logging of all incoming connections. You can also configure the use of the TCP Wrappers facility for access control.

▼ How to Log the IP Addresses of All Incoming TCP Connections

1. **Become superuser.**
2. **Turn logging on by editing the `/etc/default/inetd` file by adding the following line:**

   ```
   ENABLE_CONNECTION_LOGGING=YES
   ```

   Note – If the previous line already exists with a comment symbol, then you can just delete the comment symbol.

3. **Kill the `inetd` daemon.**
4. **Restart the `inetd` daemon.**

   See *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* and *System Administration Guide: Naming and Directory Services (FNS and NIS+)* for further information on name services.

▼ How to Use TCP Wrappers to Control Access to TCP Services

1. **Become superuser.**
2. **Turn TCP Wrappers on by editing the** `/etc/default/inetd` **file by adding the following line.**

   ```
   ENABLE_TCPWRAPPERS=YES
   ```

   Note – If the previous line already exists with a comment symbol, then you can just delete the comment symbol.

3. **Kill the** `inetd` **daemon.**
4. **Restart the** `inetd` **daemon.**
5. **Configure the TCP Wrappers access control policy as described in the** `hosts_access`**(4) man page.**

Configuring Routers

TCP/IP's first requirement for a router is that the machine must have at least two network interfaces installed. If one of the network interfaces is not disabled, the router automatically "talks" to the RDISC and RIP protocols. These protocols monitor routers on the network and advertise the router to the hosts on the network.

After the router is physically installed on the network, configure the router to operate in *local files* mode, as described in "How to Configure a Host for Local Files Mode" on page 63. This configuration ensures that routers boot if the network configuration server is down. Remember that, unlike a host, a router has a minimum of two interfaces to configure.

Configuring Routers Task Map

TABLE 4–4 Configuring Routers Task Map

Task	Description	For Instructions, Go To ...
Configure a machine as a router	Involves creating `hostname` and `hosts` file and adding addresses	"How to Configure a Machine as a Router" on page 69
Select static routing on a host that is a network client	Involves adding an entry into the `defaultrouter` file	"How to Select Static Routing on a Host That Is a Network Client" on page 70
Select dynamic routing on a host that is a network client	Involves editing entries in the `defaultrouter` file	"How to Select Dynamic Routing on a Host That Is a Network Client" on page 70
Force a machine to be a router	Involves creating a `gateways` file	"How to Force a Machine to Be a Router" on page 71

Configuring Both Router Network Interfaces

Because a router provides the interface between two or more networks, you must assign a unique name and IP address to each of the router's network interface cards. Thus, each router has a host name and an IP address that are associated with its primary network interface, plus a minimum of one more unique name and an IP address for each additional network interface.

▼ How to Configure a Machine as a Router

1. **Become superuser on the machine to be configured as a router.**
2. **Create an `/etc/hostname.`*interface* file for each network interface that is installed.**

   For example, create `hostname.ie0` and `hostname.ie1`. See "`/etc/hostname.`*interface* File" on page 88 for more information. If you are using IPv6, see "IPv6 Network Interface Configuration File" on page 324.
3. **In each file, type the host name you have selected for that interface.**

   For example, you could type the name `timbuktu` in the file `hostname.ie0`, then type the name `timbuktu-201` in the file `hostname.ie1`. Both interfaces would be located on the same machine.
4. **Type the host name and IP address of each interface into `/etc/inet/hosts`.**

   For example:

```
192.9.200.20     timbuktu       #interface for network 192.9.200
192.9.201.20     timbuktu-201   #interface for network 192.9.201
192.9.200.9      gobi
192.9.200.10     mojave
192.9.200.110    saltlake
192.9.200.12     chilean
```

The interfaces `timbuktu` and `timbuktu-201` are on the same machine. Notice that the network address for `timbuktu-201` is different from that of `timbuktu`. The difference exists because the medium for network 192.9.201 is connected to the `timbuktu-201` network interface while the media for network 192.9.200 is connected to the `timbuktu` interface. If you are using IPv6, see "`/etc/inet/ipnodes` File" on page 336.

5. **If the router is connected to any subnetted network, edit `/etc/inet/netmasks` and type the local network number (129.9.0.0, for example) and associated netmask number (255.255.255.0, for example).**

   The startup script determines whether to start up a routing protocol (RIP or RDISC) on the machine or use static routing.

▼ How to Select Static Routing on a Host That Is a Network Client

1. **Become superuser on the host.**
2. **Add an entry for a router on the network into the `/etc/defaultrouter` file.**

   See "`/etc/defaultrouter` File" on page 90. A single static default route is then installed in the routing table. Under this condition, the host does not run any dynamic routing protocol (such as RIP and RDISC).

▼ How to Select Dynamic Routing on a Host That Is a Network Client

1. **Become superuser on the host.**
2. **Ensure that the `/etc/defaultrouter` file is empty.**
   If this file is empty, a network client is forced to select a dynamic routing protocol.

   The type of dynamic routing used is selected by using the following criteria:

   - If the `/usr/sbin/in.rdisc` program exists, the startup script starts `in.rdisc`. Any router on the network that is running RDISC then responds to any RDISC queries from the host. If at least one router responds, the host selects RDISC as its

routing protocol.

- If the network router is not running RDISC or fails to respond to the RDISC queries, then `in.rdisc` on the host exits. The host then starts `in.routed`, which runs RIP.

▼ How to Force a Machine to Be a Router

You can force a machine that has only one `/etc/hostname.`*interface* file (by default a host) to be a router.

1. **Become superuser on the machine.**
2. **Create a file that is named `/etc/gateways` and leave this file empty.**

   This procedure is important if you decide to configure PPP links, as explained in *System Administration Guide: Resource Management and Network Services*.

Creating a Multihomed Host

By default, TCP/IP considers any machine with multiple network interfaces to be a router. However, you can change a router into a *multihomed host*—a machine with more than one network interface that does not run routing protocols or forward IP packets. You typically configure the following types of machines as multihomed hosts:

- NFS servers, particularly 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 for the same reason as NFS servers—to provide resources to a large pool of users.
- Firewall gateways are machines 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. However, the host can still provide standard TCP/IP services, such as `ftp` or `rlogin`, to authorized users.

Because TCP/IP considers any machine with multiple network interfaces to be a router, you need to perform a few operations to turn the machine into a multihomed host.

▼ How to Create a Multihomed Host

1. **Become superuser on the prospective multihomed host.**
2. **Create an** `/etc/hostname.`*interface* **file for each additional network interface that is installed in the machine.**
3. **Type the following:**

   % **`touch /etc/notrouter`**

   This command creates an empty file that is called `/etc/notrouter`.
4. **Reboot the machine.**

   When the machine reboots, the startup script checks for the presence of the `/etc/notrouter` file. If the file exists, the startup script does not run `in.routed -s` or `in.rdisc -r`. The file also does not turn on IP forwarding on all interfaces that are configured "up" by `ifconfig`. This process happens regardless of whether an `/etc/gateways` file exists. Thus the machine is now a multihomed host.

Turning On Space-Saving Mode

Space-saving mode provides the host with a table that contains only the default routes. On a host, `in.routed` runs with space-saving mode turned off by default.

If the host is not to have a full routing table (which provides increased protection against misconfigured routers), turn space-saving mode on.

▼ How to Turn On Space-Saving Mode

1. **Become superuser on the host.**
2. **Edit the** `/etc/rc2.d/S69inet` **startup script by adding to the line** `/usr/sbin/in.routed -q` **the** `-s` **option:**

   ```
   /usr/sbin/in.routed -q -s
   ```

   to

   ```
   /usr/sbin/in.routed -q -S
   ```

Turning Off ICMP Router Discovery

For reasons that involve router reliability, you might not want your hosts to use RDISC. If the automatic selection of RIP rather than RDISC by a host is to work reliably, the routers in the network (particularly those that run RDISC) must also work reliably.

If your routers are not running RDISC and you install a single Solaris router, by default all hosts that are connected to that router rely on that router alone. To have the hosts on that network use the other routers as well, turn off RDISC on the new router.

Turning Off ICMP Router Discovery Task Map

TABLE 4–5 Turning Off ICMP Router Discovery Task Map

Task	Description	For Instructions, Go To ...
Turn off ICMP router discovery on the host	Involves changing the name of the host's `in.rdisc` file	"`netmasks` Database" on page 93
Turn off ICMP router discovery on the router	Involves changing the name of the router's `in.rdisc` file	"What Is Subnetting?" on page 94

▼ How to Turn Off ICMP Router Discovery on the Host

1. **Become superuser on the host.**
2. **Change the name of the host's** `/usr/sbin/in.rdisc` **to some other name, such as** `/usr/sbin/in.rdisc.saved`**.**
3. **Reboot the host.**

▼ How to Turn Off ICMP Router Discovery on the Router

1. **Become superuser on the router.**
2. **Change the name of the router's** `/usr/bin/in.rdisc` **file to some other file name.**

3. **Reboot the router.**

General Troubleshooting Tips

One of the first signs of trouble on the network is a loss of communications by one or more hosts. If a host refuses 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 `ifconfig` program to obtain information on network interfaces and `netstat` to display routing tables and protocol statistics. Third-party network diagnostic programs provide a number of troubleshooting utilities. 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 Software Checks

If the network has problems, diagnose and fix software-related problems by acting in one of the following ways:

- Use the `netstat` command to display network information.
- Check the `hosts` database (and `ipnodes` if you are using IPv6) to ensure that the entries are correct and current.
- If you are running RARP, check the Ethernet addresses in the `ethers` database to ensure that the entries are correct and current.
- Try to connect to the local host by using `telnet`.
- Ensure that the network daemon `inetd` is running. Log in as superuser and type the following:

  ```
  # ps -ef | grep inetd
  ```

The following example shows the output when the `inetd` daemon is running:

```
root 57 1 0 Apr 04 ? 3:19 /usr/sbin/inetd -s
root 4218 4198 0 17:57:23 pts/3 0:00 grep inetd
```

ping Command

Use the ping command to find out whether an IP connection exists for a particular host. The basic syntax is:

/usr/sbin/ping *host [timeout]*

In this syntax, *host* is the host name of the machine in question. The optional *timeout* argument indicates the time in seconds for ping to continue trying to reach the machine—20 seconds by default. The ping(1M) man page describes additional syntaxes and options.

When you run ping, the ICMP protocol sends a datagram to the host you specify, asking for a response. ICMP is the protocol responsible for error handling on a TCP/IP network. See "ICMP Protocol" on page 35 for details.

ping Command Task Map

TABLE 4–6 ping Command Task Map

Task	Description	For Instructions, Go To ...
Determine if a host is running	Involves pinging the hostname	"Network Databases and nsswitch.conf File" on page 97
Determine if a host is losing packets	Involves using the -s option of the ping command	"How Name Services Affect Network Databases" on page 97

▼ How to Determine if a Host Is Running

- **On the command line, type the following command.**

  % **ping** *hostname*

  If host *hostname* is up, 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 `ping`:

`no answer from` *hostname*

▼ How to Determine if a Host Is Losing Packets

If you suspect that a machine might be losing packets even though the machine is running, you can use the `s` option of `ping` to try to detect the problem.

- **On the command line, type the following command.**

  `% ping -s` *hostname*

  `ping` continually sends packets to *hostname* until you send an interrupt character or a timeout occurs. The responses on your screen resemble the following:

```
PING elvis: 56 data bytes
64 bytes from 129.144.50.21: icmp_seq=0. time=80. ms
64 bytes from 129.144.50.21: icmp_seq=1. time=0. ms
64 bytes from 129.144.50.21: icmp_seq=2. time=0. ms
64 bytes from 129.144.50.21: icmp_seq=3. time=0. ms
.
.
.
----elvis PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms) min/avg/max = 0/20/80
```

  The packet-loss statistic indicates whether the host has dropped packets.

  If `ping` fails, check the status of the network that is reported by `ifconfig` and `netstat`, as described in "`ifconfig` Command" on page 76 and "`netstat` Command" on page 78.

`ifconfig` Command

The `ifconfig` command displays information about the configuration of an interface that you specify. Refer to the `ifconfig`(1M) man page for details. The syntax of `ifconfig` follows:

`ifconfig` *interface-name [protocol_family]*

ifconfig Command Task Map

TABLE 4–7 ifconfig Command Task Map

Task	Description	For Instructions, Go To ...
Get information about a specific interface	Involves using the ifconfig command	"How to Get Information About a Specific Interface" on page 77
Get information about all interfaces on a network	Involves using the -a option of the ifconfig command	"nsswitch.conf File — Specifying Which Name Service to Use" on page 99

▼ How to Get Information About a Specific Interface

1. **Become superuser.**
2. **On the command line, type the following command.**

   # **ifconfig** *interface*

   For an le0 interface, your output resembles the following:

```
le0: flags=863<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
     inet 129.144.44.140 netmask ffffff00 broadcast 129.144.44.255
ether 8:0:20:8:el:fd
```

   The previous flags section shows that the interface is configured "up," capable of broadcasting, and not using "trailer" link-level encapsulation. The mtu field tells you that this interface has a maximum transfer size of 1500 octets. Information on the second line includes the IP address of the host you are using, the netmask being currently used, and the IP broadcast address of the interface. The third line gives the machine address (Ethernet, in this instance) of the host.

▼ How to Get Information About All Interfaces on a Network

A useful ifconfig option is -a, which provides information on all interfaces on your network.

1. **Become superuser.**
2. **On the command line, type the following command.**

   # **ifconfig -a** *interface*

   This command produces, for example:

```
le0:  flags=49<UP,LOOPBACK,RUNNING> mtu 8232
     inet 127.144.44.140 netmask ff000000
le0:flags=863<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
     inet 129.144.44.140 netmask ffffff00 broadcast 129.144.44.255
ether 8:0:20:8:el:fd
```

Output that indicates an interface is not running might mean a problem with that interface. In this instance, see the `ifconfig`(1M) man page.

`netstat` Command

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

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

`netstat [-m] [-n] [-s] [-i | -r] [-f` *address_family*`]`

The most frequently used options for determining network status are `s`, `r`, and `i`. See the `netstat`(1M) man page for a description of the options.

`netstat` Command Task Map

TABLE 4–8 `netstat` Command Task Map

Task	Description	For Instructions, Go To ...
Display statistics by protocol	Involves using the `-s` option of the `netstat` command	"How to Display Statistics by Protocol" on page 79
Display network interface status	Involves using the `-i` option of the `netstat` command	"How to Display Network Interface Status" on page 80
Display routing table status	Involves using the `-r` option of the `netstat` command	"How to Display Routing Table Status" on page 80

▼ How to Display Statistics by Protocol

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

- **On the command line, type the following command.**

  ```
  % netstat -s
  ```

  The result resembles the display that is shown in the following example. (Parts of the output have been truncated.) The information can indicate areas where a protocol is having problems. For example, statistical information from ICMP can indicate where this protocol has found errors.

```
UDP

      udpInDatagrams      =  39228     udpOutDatagrams     =  2455
      udpInErrors         =     0

TCP

      tcpRtoAlgorithm     =     4      tcpMaxConn          =    -1
      tcpRtoMax           = 60000      tcpPassiveOpens     =     2
      tcpActiveOpens      =     4      tcpEstabResets      =     1
      tcpAttemptFails     =     3      tcpOutSegs          =   315
            .
            .
IP

      ipForwarding        =     2      ipDefaultTTL        =   255
      ipInReceives        =  4518      ipInHdrErrors       =     0
            .
            .
ICMP

      icmpInMsgs          =     0      icmpInErrors        =     0
      icmpInCksumErrs     =     0      icmpInUnknowns      =     0
            .
            .

IGMP:

0 messages received
0 messages received with too few bytes
0 messages received with bad checksum
0 membership queries received
0 membership queries received with invalid field(s)
0 membership reports received
0 membership reports received with invalid field(s)
0 membership reports received for groups to which we belong
0 membership reports sent
```

▼ How to Display Network Interface Status

The `i` option of `netstat` shows the state of the network interfaces that are configured with the machine where you ran the command.

- **On the command line, type the following command:**

```
% netstat -i
```

`netstat -i` produced the following sample display:

```
Name Mtu  Net/Dest     Address   Ipkts    Ierrs Opkts    Oerrs  Collis  Queue
le0  1500 b5-spd-2f-cm tatra     14093893 8492  10174659 1119   2314178   0
lo0  8232 loopback     localhost 92997622 5442  12451748 0      775125    0
```

Using this display, you can determine the number of packets a machine transmits and receives on each network. For example, the input packet count (`Ipkts`) that are 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, but does not realize that the server is supposed to respond to them. This confusion might be caused by an incorrect address in the `hosts`, `ipnodes`, 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.

▼ How to Display Routing Table Status

The `-r` option of `netstat` displays the IP routing table.

- **On the command line, type the following command.**

```
% netstat -r
```

`netstat -r` produces the following sample display on machine `tenere`:

```
Routing tables
Destination   Gateway Flags Refcnt Use   Interface
temp8milptp   elvis   UGH   0      0
irmcpeb1-ptp0 elvis   UGH   0      0
route93-ptp0  speed   UGH   0      0
mtvb9-ptp0    speed   UGH   0      0
                  .
mtnside       speed   UG    1      567
ray-net       speed   UG    0      0
mtnside-eng   speed   UG    0      36
mtnside-eng   speed   UG    0      558
mtnside-eng   tenere  U     33     190248  le0
```

The first column shows the destination network, the second the router through which packets are forwarded. The `U` flag indicates that the route is up. The `G` flag indicates that the route is to a gateway. The `H` flag indicates that the destination is a fully qualified host address, rather than a network.

The `Refcnt` column shows the number of active uses per route, and the `Use` column shows the number of packets sent per route. Finally, the `Interface` column shows the network interface that the route uses.

Logging Network Problems

If you suspect a routing daemon malfunction, you can log its actions, including all packet transfers when you start up the `routed` daemon.

▼ How to Log Network Problems

1. **Become superuser.**
2. **Create a log file of routing daemon actions by typing the following command at a command-line prompt.**

   # **/usr/sbin/in.routed /var/***logfilename*

Caution – On a busy network, this command can generate almost continuous output.

Displaying Packet Contents

You can use `snoop` to capture network packets and display their contents. 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. For information about using the `snoop` command, refer to the `snoop`(1M) man page.

The snoop command must be run by root (#) to capture packets to and from the default interface in promiscuous mode. In summary form, only the data that pertains to the highest-level protocol is displayed. 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.

The snoop capture file format is described in RFC 1761.

snoop server client rpc rstatd collects all RPC traffic between a client and server, and filters the traffic for rstatd.

Displaying Packet Contents Task Map

TABLE 4–9 Displaying Packet Contents Task Map

Task	Description	For Instructions, Go To ...
Check all packets from your system	Involves using the netstat and snoop commands and interpreting the results	"How to Check All Packets From Your System" on page 82
Capture snoop results to a file	Involves using the -o option of the snoop command	"How to Capture snoop Results to a File" on page 83
Check packets between server and client	Involves saving the results of the snoop command to a file and inspecting the results	"How to Check Packets Between Server and Client" on page 84

▼ How to Check All Packets From Your System

1. **Become superuser.**
2. **Type the following command at the command-line prompt to find the interfaces that are attached to the system.**

   ```
   # netstat -i
   ```

   snoop normally uses the first non-loopback device (le0).
3. **Type** snoop**.**

   Use Control-C to halt the process.

   ```
   # snoop
   Using device /dev/le (promiscuous mode)
        maupiti -> atlantic-82  NFS C GETATTR FH=0343
    atlantic-82 -> maupiti      NFS R GETATTR OK
        maupiti -> atlantic-82  NFS C GETATTR FH=D360
    atlantic-82 -> maupiti      NFS R GETATTR OK
   ```

```
    maupiti -> atlantic-82  NFS C GETATTR FH=1A18
atlantic-82 -> maupiti      NFS R GETATTR OK
    maupiti -> (broadcast)  ARP C Who is 120.146.82.36, npmpk17a-82 ?
```

4. **Interpret the results.**

   In the example, client `maupiti` transmits to server `atlantic-82` by using NFS file handle 0343. `atlantic-82` acknowledges with OK. The conversation continues until `maupiti` broadcasts an ARP request that asks who is `120.146.82.36`?

   This example demonstrates the format of `snoop`. The next step is to filter `snoop` to capture packets to a file.

   Interpret the capture file by using details that are described in RFC 1761.

▼ How to Capture `snoop` Results to a File

1. **Become superuser.**

2. **On the command line, type the following command.**

   # **snoop -o** *filename*

   For example:

   ```
   # snoop -o /tmp/cap
   Using device /dev/le (promiscuous mode)
   30 snoop: 30 packets captured
   ```

   By using this command, you have captured 30 packets in a file `/tmp/cap`. The file can be anywhere 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 (see the next section).

3. **On the command line, type the following command to inspect the file.**

   # **snoop -i** *filename*

   For example:

```
# snoop -i /tmp/cap

1  0.00000 frmpk17b-082 -> 224.0.0.2    IP  D=224.0.0.2 S=129.146.82.1 LEN=32, ID=0
2  0.56104        scout -> (broadcast)  ARP C Who is 129.146.82.63, grail ?
3  0.16742  atlantic-82 -> (broadcast)  ARP C Who is 129.146.82.76, honeybea ?
4  0.77247        scout -> (broadcast)  ARP C Who is 129.146.82.63, grail ?
5  0.80532 frmpk17b-082 -> (broadcast)  ARP C Who is 129.146.82.92, holmes ?
6  0.13462        scout -> (broadcast)  ARP C Who is 129.146.82.63, grail ?
7  0.94003        scout -> (broadcast)  ARP C Who is 129.146.82.63, grail ?
8  0.93992        scout -> (broadcast)  ARP C Who is 129.146.82.63, grail ?
9  0.60887        towel -> (broadcast)  ARP C Who is 129.146.82.35, udmpk17b-82 ?
10 0.86691  nimpk17a-82 -> 129.146.82.255 RIP R (1 destinations)
```

Refer to specific protocol documentation for detailed analysis and recommended parameters for ARP, IP, RIP and so forth. The Web contains a number of requests for comments.

▼ How to Check Packets Between Server and Client

1. **Establish a snoop system off a hub that is connected to either the client or 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. **Become superuser.**
3. **On the command line, type `snoop` with options and save to a file.**
4. **Inspect and interpret results.**
   Look at RFC 1761 for details of the snoop capture file.

   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 YP. For details on using `snoop` and its options, refer to the `snoop`(1M) man page.

Displaying Routing Information

Use the `traceroute` utility to trace the route an IP packet follows to some Internet host. The `traceroute` utility utilizes the IP protocol (time-to-live) `ttl` field and attempts to elicit an ICMP `TIME_EXCEEDED` response from each gateway along the path. This utility also attempts to elicit the response `PORT_UNREACHABLE` (or `ECHO_REPLY`) from the destination host. The `traceroute` utility sends probes with a `ttl` of one and increases by one until the intended host is found or has incremented beyond the maximum number of intermediate hosts.

The `traceroute` utility is especially useful for determining routing misconfiguration and routing path failures. If a particular host is unreachable, you can use the `traceroute` utility to see what path the packet follows to the intended host and where possible failures might occur.

The `traceroute` utility 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 Run the Traceroute Utility

- **On the command line, type the following command.**

  `% traceroute` *destination-hostname*

  For details of the `traceroute` utility, see the `traceroute`(1M) man page.

Example—`traceroute` Utility

The following sample of the `traceroute` command shows the seven–hop path a packet follows from the host `istanbul` to the host `sanfrancisco`, along with the times for a packet to traverse each hop.

```
istanbul% traceroute sanfrancisco
    traceroute: Warning: Multiple interfaces found; using 172.31.86.247 @ le0
    traceroute to sanfrancisco (172.29.64.39), 30 hops max, 40 byte packets
     1  frbldg7c-86 (172.31.86.1)  1.516 ms  1.283 ms  1.362 ms
     2  bldg1a-001 (172.31.1.211)  2.277 ms  1.773 ms  2.186 ms
     3  bldg4-bldg1 (172.30.4.42)  1.978 ms  1.986 ms  13.996 ms
     4  bldg6-bldg4 (172.30.4.49)  2.655 ms  3.042 ms  2.344 ms
     5  ferbldg11a-001 (172.29.1.236)  2.636 ms  3.432 ms  3.830 ms
     6  frbldg12b-153 (172.29.153.72)  3.452 ms  3.146 ms  2.962 ms
     7  sanfrancisco (172.29.64.39)  3.430 ms  3.312 ms  3.451 ms
```

CHAPTER 5

TCP/IP (Reference)

This chapter provides TCP/IP network reference information about TCP/IP configuration files, including the types, their purpose, and the format of the file entries. The existing network databases are also described in detail.

The chapter also shows how the structure of IPv4 addresses are derived, based on defined network classifications and subnet numbers.

For additional information about Internet Transmission Control Protocol (TCP), see the `tcp`(7P) man page.

This chapter contains the following information:

TCP/IP Configuration Files

Each machine on the network obtains its TCP/IP configuration information from the following TCP/IP configuration files and network databases:

- `/etc/hostname.`*interface* file
- `/etc/nodename` file
- `/etc/defaultdomain` file
- `/etc/defaultrouter` file (optional)

- `hosts` database
- `ipnodes` database
- `netmasks` database (optional)

The Solaris installation program creates these files as part of the installation process. You can also edit the files manually, as explained in this section. The `hosts` and `netmasks` databases are two of the network databases read by the name services available on Solaris networks. "Network Databases and `nsswitch.conf` File" on page 97 describes the concept of network databases in detail. For information on the `ipnodes` file, see "`/etc/inet/ipnodes` File" on page 336.

`/etc/hostname.`*interface* File

This file defines the network interfaces on the local host for IPv4. A minimum of one `/etc/hostname.`*interface* file should exist on the local machine. The Solaris installation program creates this file for you. In the file name, *interface* is replaced by the device name of the primary network interface.

Note – If you add a new network interface to your system after the initial Solaris software installation, you must create an `/etc/hostname.`*interface* file for that interface, add the interface's IP address to the `/etc/inet/hosts` file, and reboot the system with the `-r` option. See substeps within "How to Configure a Host for Local Files Mode" on page 63 for instructions. Also, in order for the Solaris software to recognize and use the new network interface, you need to load the interface's device driver into the appropriate directory. Refer to the documentation that comes with the new network interface for the appropriate *interface* name and device driver instructions.

The file contains only one entry: the host name or IPv4 address that is associated with the network interface. For example, suppose `smc0` is the primary network interface for a machine that is called `tenere`. The `/etc/hostname.`*interface* file would have the name `/etc/hostname.smc0`. The file would contain the entry `tenere`.

Files for Multiple Network Interfaces

If a machine contains more than one network interface, you must create additional `/etc/hostname.`*interface* files for the additional network interfaces. You must create these files with a text editor. The Solaris installation program does not create them for you.

For example, consider the machine `timbuktu`, which is shown in Figure 4–1. This machine has two network interfaces and functions as a router. The primary network interface `le0` is connected to network 192.9.200. The IP address is 192.9.200.70, and its

host name is `timbuktu`. The Solaris installation program creates the file `/etc/hostname.le0` for the primary network interface and enters the host name `timbuktu` in the file.

The second network interface is `le1`. This interface is connected to network 192.9.201. Although this interface is physically installed on machine `timbuktu`, the interface must have a separate IPv4 address. Therefore, you have to manually create the `/etc/hostname.le1` file for this interface. The entry in the file would be the router's name, `timbuktu-201`.

`/etc/hostname6.`*interface* File

IPv6 uses the file `/etc/hostname6.`*interface* at start up to automatically define network interfaces in the same way IPv4 uses `/etc/hostname.`*interface*. A minimum of one `/etc/hostname.` or `/etc/hostname6.` file should exist on the local machine. The Solaris installation program creates these files for you. In the file name, replace *interface* with the device name of the primary network interface. For more information about the `/etc/hostname6.`*interface* file, see "IPv6 Network Interface Configuration File" on page 324.

`/etc/nodename` File

This file should contain one entry: the host name of the local machine. For example, on machine `timbuktu`, the file `/etc/nodename` would contain the entry `timbuktu`.

`/etc/defaultdomain` File

This file should contain one entry, the fully qualified domain name of the administrative domain to which the local host's network belongs. You can supply this name to the Solaris installation program or edit the file at a later date.

In Figure 4–1, the networks are part of the domain `deserts.worldwide`, which was classified as a `.com` domain. Therefore, `/etc/defaultdomain` should contain the entry `deserts.worldwide.com`. For more information on network domains, refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

/etc/defaultrouter File

This file should contain an entry for each router that is directly connected to the network. The entry should be the name for the network interface that functions as a router between networks.

In Figure 4–1, the network interface `le1` connects machine `timbuktu` with network 192.9.201. This interface has the unique name `timbuktu-201`. Thus, the machines on network 192.9.200 that are configured in local files mode have the name `timbuktu-201` as the entry in `/etc/defaultrouter`.

hosts Database

The `hosts` database contains the IPv4 addresses and host names of machines on your network. If you use the NIS, NIS+, or DNS name services (or LDAP as a name service), the `hosts` database is maintained in a database that is designated for host information. For example, on a network that runs NIS+, the `hosts` database is maintained in the host table.

If you use local files for the name service, the `hosts` database is maintained in the `/etc/inet/hosts` file. This file contains the host names and IPv4 addresses of the primary network interface, other network interfaces that are attached to the machine, and any other network addresses that the machine must check for.

Note – For compatibility with BSD-based operating systems, the file `/etc/hosts` is a symbolic link to `/etc/inet/hosts`.

/etc/inet/hosts File Format

The `/etc/inet/hosts` file uses the basic syntax that follows. Refer to the `hosts`(4) man page for complete syntax information.

IPv4-address hostname [nicknames] [#comment]

IPv4-address contains the IPv4 address for each interface that the local host must recognize.

hostname contains the host name that is assigned to the machine at setup, plus the host names that are assigned to additional network interfaces that the local host must recognize.

[nickname] is an optional field that contains a nickname for the host.

[# comment] is an optional field for a comment.

Initial /etc/inet/hosts File

When you run the Solaris installation program on a machine, the program configures the initial /etc/inet/hosts file. This file contains the minimum entries that the local host requires. The entries include the loopback address, the host IPv4 address, and the host name.

For example, the Solaris installation program might create the following /etc/inet/hosts file for machine tenere shown in Figure 4–1:

EXAMPLE 5–1 /etc/inet/hosts File for Machine ahaggar

```
127.0.0.1     localhost         loghost    #loopback address
192.9.200.3   tenere                       #host name
```

Loopback Address

In Example 5–1, the IPv4 address 127.0.0.1 is the *loopback address*. The loopback address is the reserved network interface that is used by the local machine to allow interprocess communication. This enables the host to send packets to itself. The ifconfig command uses the loopback address for configuration and testing, as explained in "ifconfig Command" on page 76. Every machine on a TCP/IP network must use the IP address 127.0.0.1 for the local host.

Host Name

The IPv4 address 192.9.200.1 and the name tenere are the address and host name of the local machine. They are assigned to the machine's primary network interface.

Multiple Network Interfaces

Some machines have more than one network interface, because they are either routers or multihomed hosts. Each additional network interface that is attached to the machine requires its own IPv4 address and associated name. When you configure a router or multihomed host, you must add this information manually to the router's /etc/inet/hosts file. See "Configuring Routers" on page 68 for more information on configuring routers and multihomed hosts.

Example 5–2 is the /etc/inet/hosts file for machine timbuktu that is shown in Figure 4–1.

EXAMPLE 5–2 /etc/inet/hosts File for Machine timbuktu

```
127.0.0.1      localhost     loghost
192.9.200.70   timbuktu      #This is the local host name
192.9.201.10   timbuktu-201  #Interface to network 192.9.201
```

With these two interfaces, `timbuktu` connects networks 192.9.200 and 192.9.201 as a router.

How Name Services Affect the `hosts` Database

The NIS, NIS+, and DNS name services (or LDAP as a name service) maintain host names and addresses on one or more servers. These servers maintain `hosts` databases that contain information for every host and router (if applicable) on the servers' network. Refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* and *System Administration Guide: Naming and Directory Services (FNS and NIS+)* for more information about these services.

When Local Files Provide Name Service

On a network that uses local files for name service, machines that run in local files mode consult their individual `/etc/inet/hosts` files for IPv4 addresses and host names of other machines on the network. Therefore, these machine's `/etc/inet/hosts` files must contain the following:

- Loopback address
- IPv4 address and host name of the local machine (primary network interface)
- IPv4 address and host name of additional network interfaces that are attached to this machine, if applicable
- IPv4 addresses and host names of all hosts on the local network
- IPv4 addresses and host names of any routers that this machine must know about, if applicable
- IPv4 address of any machine your machine wants to refer to by its host name

The figure below shows the `/etc/inet/hosts` file for machine `tenere`. This machine runs in local files mode. Notice that the file contains the IPv4 addresses and host names for every machine on the 192.9.200 network. The file also contains the IPv4 address and interface name `timbuktu-201`. This interface connects the 192.9.200 network to the 192.9.201 network.

A machine that is configured as a network client uses the local `/etc/inet/hosts` file for its loopback address and IPv4 address.

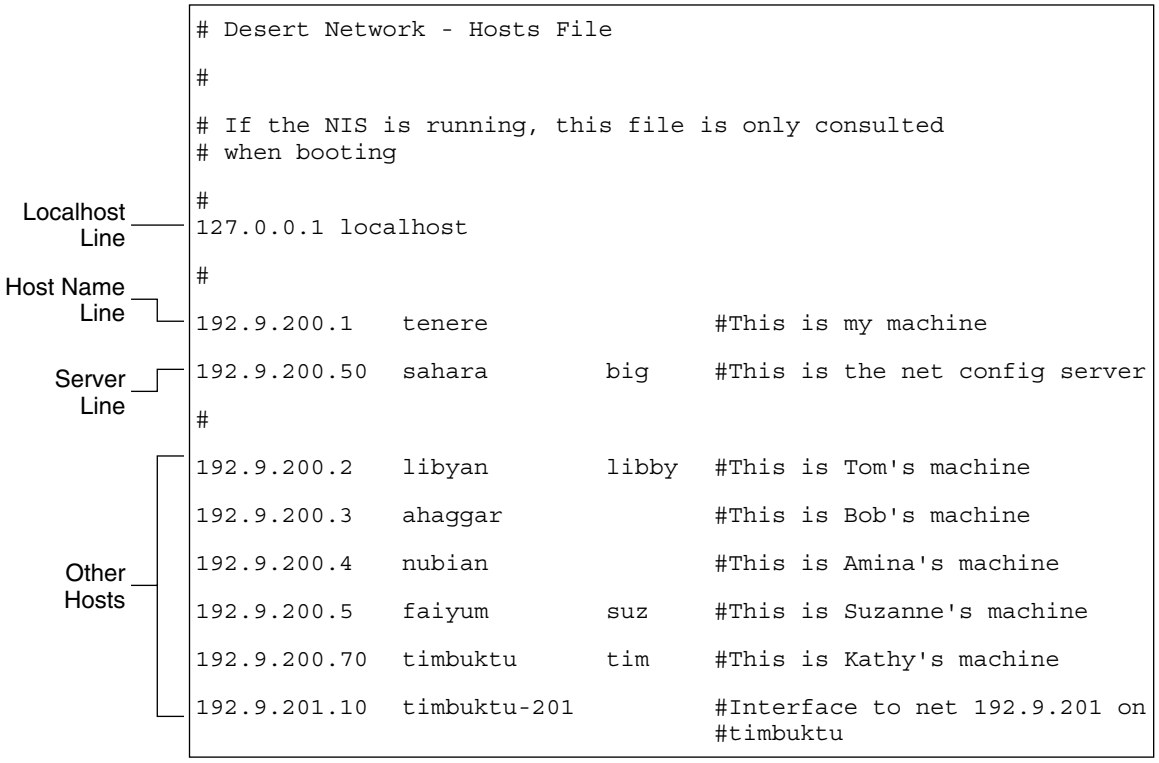


FIGURE 5–1 `/etc/inet/hosts` File for Machine Running in Local Files Mode

`ipnodes` Database

The `ipnodes` database contains the IPv6 addresses and host names of machines on your network. If you use the NIS, NIS+, or DNS name services (or LDAP as a name service), the `ipnodes` database is maintained in a database that is designated for host information. For example, on a network that runs NIS+, the `ipnodes` database is maintained in the host table. For more information about the `ipnodes` database, see "`/etc/inet/ipnodes` File" on page 336.

`netmasks` Database

You need to edit the `netmasks` database as part of network configuration *only* if you have set up subnetting on your network. The `netmasks` database consists of a list of networks and their associated subnet masks.

Note – When you create subnets, each new network must be a separate physical network. You cannot apply subnetting to a single physical network.

What Is Subnetting?

Subnetting is a method for maximizing the limited 32-bit IPv4 addressing space and reducing the size of the routing tables in a large internetwork. With any address class, subnetting provides a means of allocating a part of the host address space to network addresses, which lets you have more networks. The part of the host address space that is allocated to new network addresses is known as the *subnet* number.

In addition to making more efficient use of the IPv4 address space, subnetting has several administrative benefits. Routing can become very complicated as the number of networks grows. A small organization, for example, might give each local network a class C number. As the organization grows, the administration of a number of different network numbers could become complicated. A better idea is to allocate a few class B network numbers to each major division in an organization. For instance, you could allocate one to Engineering, one to Operations, and so on. Then, you could divide each class B network into additional networks, using the additional network numbers gained by subnetting. This division can also reduce the amount of routing information that must be communicated among routers.

Creating the Network Mask for IPv4 Addresses

As part of the subnetting process, you need to select a network-wide netmask. The netmask determines how many and which bits in the host address space represent the subnet number and how many bits and which represent the host number. Recall that the complete IPv4 address consists of 32 bits. Depending on the address class, as many as 24 bits and as few as 8 bits can be available for representing the host address space. The netmask is specified in the `netmasks` database.

If you plan to use subnets, you must determine your netmask before you configure TCP/IP. If you plan to install the operating system as part of network configuration, the Solaris installation program requests the netmask for your network.

As described in “Administering Network Numbers” on page 47, 32-bit IP addresses consist of a network part and a host part. The 32 bits are divided into 4 bytes. Each byte is assigned to either the network number or the host number, depending on the network class.

For example, in a class B IPv4 address, the 2 bytes on the left are assigned to the network number, and the 2 bytes on the right are assigned to the host number. In the class B IPv4 address 129.144.41.10, you can assign the 2 bytes on the right to hosts.

If you are to implement subnetting, you need to use some of the bits in the bytes that are assigned to the host number to apply to subnet addresses. For example, a 16-bit host address space provides addressing for 65,534 hosts. If you apply the third byte to subnet addresses and the fourth to host addresses, you can address up to 254 networks, with up to 254 hosts on each network.

The bits in the host address bytes that are applied to subnet addresses and those applied to host addresses are determined by a subnet mask. Subnet masks are used to select bits from either byte for use as subnet addresses. Although netmask bits must be contiguous, they need not align on byte boundaries.

The netmask can be applied to an IPv4 address by using the bitwise logical AND operator. This operation selects out the network number and subnet number positions of the address.

Netmasks can be explained in terms of their binary representation. You can use a calculator for binary-to-decimal conversion. The following examples show both the decimal and binary forms of the netmask.

If a netmask 255.255.255.0 is applied to the IPv4 address 129.144.41.101, the result is the IPv4 address of 129.144.41.0.

129.144.41.101 & 255.255.255.0 = 129.144.41.0

In binary form, the operation is as follows:

10000001.10010000.00101001.01100101 (IPv4 address)

ANDed with

11111111.11111111.11111111.00000000 (netmask)

Now the system looks for a network number of 129.144.41 instead of a network number of 129.144. If your network has the number 129.144.41, that number is what the system checks for and finds. Because you can assign up to 254 values to the third byte of the IPv4 address space, subnetting lets you create address space for 254 networks, where previously space was available for only one.

If you are providing address space for only two additional networks, you can use the following subnet mask:

255.255.192.0

This netmask provides the following result:

11111111.11111111.1100000.00000000

This result still leaves 14 bits available for host addresses. Because all 0s and 1s are reserved, a minimum of 2 bits must be reserved for the host number.

/etc/inet/netmasks File

If your network runs NIS, NIS+, or LDAP, the servers for these name services maintain `netmasks` databases. For networks that use local files for name service, this information is maintained in the `/etc/inet/netmasks` file.

Note – For compatibility with BSD-based operating systems, the file `/etc/netmasks` is a symbolic link to `/etc/inet/netmasks`.

The following example shows the `/etc/inet/netmasks` file for a class B network.

EXAMPLE 5–3 `/etc/inet/netmasks` File for a Class B Network

```
## The netmasks file associates Internet Protocol (IPv4) address
# masks with IPv4 network numbers.
#
#     network-number    netmask
#
# Both the network-number and the netmasks are specified in
# "decimal dot" notation, e.g:
#
#        128.32.0.0   255.255.255.0
129.144.0.0  255.255.255.0
```

If the file does not exist, create it. Use the following syntax:

network-number netmask-number

Refer to the `netmasks`(4) man page for complete details.

When creating netmask numbers, type the network number that is assigned by the InterNIC (not the subnet number) and netmask number in `/etc/inet/netmasks`. Each subnet mask should be on a separate line.

For example:

```
128.78.0.0        255.255.248.0
```

You can also type symbolic names for network numbers in the `/etc/inet/hosts` file. You can then use these network names instead of the network numbers as parameters to commands.

Network Databases and `nsswitch.conf` File

The network databases are files that provide information that is needed to configure the network. The network databases follow:

- `hosts`
- `ipnodes`
- `netmasks`
- `ethers`
- `bootparams`
- `protocols`
- `services`
- `networks`

As part of the configuration process, you edit the `hosts` database and the `netmasks` database, if your network is subnetted. Two network databases, `bootparams` and `ethers`, are used to configure machines as network clients. The remaining databases are used by the operating system and seldom require editing.

Although `nsswitch.conf` file is not a network database, you need to configure this file along with the relevant network databases. `nsswitch.conf` specifies which name service to use for a particular machine: local files, NIS, NIS+, DNS, or LDAP.

How Name Services Affect Network Databases

The form of your network database depends on the type of name service you select for your network. For example, the `hosts` database contains, at minimum, the host name and IPv4 address of the local machine and any network interfaces that are directly connected to the local machine. 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+ tables.
- NIS uses databases that are called NIS maps.
- DNS uses records with host information.

Note – DNS boot and data files do not correspond directly to the network databases.

The following figure shows the forms of the `hosts` database that is used by these name services.

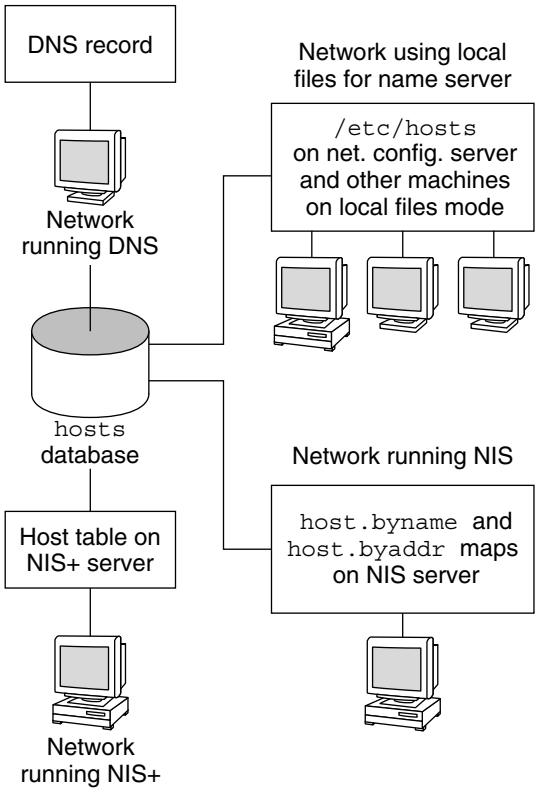


FIGURE 5–2 Forms of the `hosts` Database Used by Name Services

The following table lists the network databases and how they are used by local files, NIS+, and NIS.

TABLE 5–1 Network Databases and Corresponding Name Service Files

Network Database	Local Files	NIS+ Tables	NIS Maps
`hosts`	`/etc/inet/hosts`	`hosts.org_dir`	`hosts.byaddr` `hosts.byname`

TABLE 5–1 Network Databases and Corresponding Name Service Files *(Continued)*

Network Database	Local Files	NIS+ Tables	NIS Maps
ipnodes	/etc/inet/ipnodes	ipnodes.org_dir	ipnodes.byaddr ipnodes.byname
netmasks	/etc/inet/netmasks	netmasks.org_dir	netmasks.byaddr
ethers	/etc/ethers	ethers.org_dir	ethers.byname ethers.byaddr
bootparams	/etc/bootparams	bootparams.org_dir	bootparams
protocols	/etc/inet/protocols	protocols.org_dir	protocols.byname protocols.bynumber
services	/etc/inet/services	services.org_dir	services.byname
networks	/etc/inet/networks	networks.org_dir	networks.byaddr networks.byname

This book discusses network databases as viewed by networks that use local files for name services. Information about the `hosts` database is in "`hosts` Database" on page 90. Information about the `ipnodes` database is in "`/etc/inet/ipnodes` File" on page 336. Information about the `netmasks` database is in "`netmasks` Database" on page 93. Refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* and *System Administration Guide: Naming and Directory Services (FNS and NIS+)* for information on network databases correspondences in NIS, NIS+, DNS, and LDAP.

`nsswitch.conf` File — Specifying Which Name Service to Use

The `/etc/nsswitch.conf` file defines the search order of the network databases. The Solaris installation program creates a default `/etc/nsswitch.conf` file for the local machine, based on the name service you indicate during the installation process. If you selected the "None" option, indicating local files for name service, the resulting `nsswitch.conf` file resembles the following example.

EXAMPLE 5–4 `nsswitch.conf` for Networks Using Files for Name Service

```
# /etc/nsswitch.files:
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it does not use any naming service.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file contains "switch.so" as a
# nametoaddr library for "inet" transports.
```

EXAMPLE 5–4 `nsswitch.conf` for Networks Using Files for Name Service *(Continued)*

```
passwd:          files
group:           files
hosts:           files
networks:        files
protocols:       files
rpc:             files
ethers:          files
netmasks:        files
bootparams:      files
publickey:       files
# At present there isn't a 'files' backend for netgroup; the
# system will figure it out pretty quickly,
# and won't use netgroups at all.
netgroup:        files
automount:       files
aliases:         files
services:        files
sendmailvars:    files
```

The `nsswitch.conf`(4) man page describes the file in detail. The file's basic syntax is:

database name-service-to-search

The *database* field can list one of many types of databases that are searched by the operating system. For example, the field could indicate a database that affects users, such as `passwd` or `aliases`, or a network database. The parameter *name-service-to-search* can have the values `files`, `nis`, or `nis+` for the network databases. The `hosts` database can also have `dns` as a name service to search. You can also list more than one name service, such as `nis+` and `files`.

In Example 5–4, the only search option that is indicated is `files`. Therefore, the local machine obtains security and automounting information, in addition to network database information, from files that are located in its `/etc` and `/etc/inet` directories.

Changing `nsswitch.conf`

The `/etc` directory contains the `nsswitch.conf` file that is created by the Solaris installation program. This directory also contains template files for the following name services:

- `nsswitch.files`
- `nsswitch.nis`
- `nsswitch.nis+`

If you want to change from one name service to another, you can copy the appropriate template to `nsswitch.conf`. You can also selectively edit the `nsswitch.conf` file, and change the default name service to search for individual databases.

For example, on a network that runs NIS, you might have to change the `nsswitch.conf` file on network clients. The search path for the `bootparams` and `ethers` databases must list `files` as the first option, and `nis`. The following example shows the correct search paths.

EXAMPLE 5–5 `nsswitch.conf` for a Client on a Network Running NIS

```
## /etc/nsswitch.conf:#
.
.
passwd:        files nis
group:         file nis

# consult /etc "files" only if nis is down.
hosts:         nis    [NOTFOUND=return] files
networks:      nis    [NOTFOUND=return] files
protocols:     nis    [NOTFOUND=return] files
rpc:           nis    [NOTFOUND=return] files
ethers:        files  [NOTFOUND=return] nis
netmasks:      nis    [NOTFOUND=return] files
bootparams:    files  [NOTFOUND=return] nis
publickey:     nis
netgroup:      nis

automount:     files nis
aliases:       files nis

# for efficient getservbyname() avoid nis
services:      files nis
sendmailvars:  files
```

For complete details on the name service switch, refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* and *System Administration Guide: Naming and Directory Services (FNS and NIS+)*.

`bootparams` Database

The `bootparams` database contains information that is used by machines that are configured to boot in the network client mode. You need to edit this database if your network has network clients. See “Configuring Network Clients” on page 65 for procedures. The database is built from information that is entered into the `/etc/bootparams` file.

The bootparams(4) man page contains complete syntax for this database. The man page's basic syntax is shown in the following example:

machine-name file-key-server-name:pathname

For each network client machine, the entry might contain the following information: the name of the client, a list of keys, the names of servers, and path names.

The first item of each entry is the name of the client machine. Next is a list of keys, names of servers, and path names, separated by tab characters. All items but the first are optional. An example follows.

EXAMPLE 5–6 bootparams Database

```
myclient   root=myserver : /nfsroot/myclient  \
swap=myserver : /nfsswap//myclient \
dump=myserver : /nfsdump/myclient
```

In this example, the term dump=: tells client hosts not to look for a dump file.

Wildcard Entry for bootparams

In most instances, use the wildcard entry when editing the bootparams database to support clients. This entry follows:

```
*  root=server:/path dump=:
```

The asterisk (*) wildcard indicates that this entry applies to all clients that are not specifically named within the bootparams database.

ethers Database

The ethers database is built from information that is entered into the /etc/ethers file. This database associates host names to their Ethernet addresses. You need to create an ethers database only if you are running the RARP daemon. That is, you need to create this database if you are configuring network clients.

RARP uses the file to map Ethernet addresses to IP addresses. If you are running the RARP daemon in.rarpd, you need to set up the ethers file and maintain this file on all hosts that are running the daemon to reflect changes to the network.

The ethers(4) man page contains complete syntax information for this database. The man page's basic format follows:

Ethernet-address hostname #comment

Ethernet-address is the Ethernet address of the host.

hostname is the official name of the host.

#comment is any note that you want to append to an entry in the file.

The equipment manufacturer provides the Ethernet address. If a machine does not display the Ethernet address when you power up, see your hardware manuals for assistance.

When adding entries to the `ethers` database, ensure that host names correspond to the primary names in the `hosts` and `ipnodes` databases, not to the nicknames, as follows.

EXAMPLE 5–7 Entries in the `ethers` Database

```
8:0:20:1:40:16  fayoum
8:0:20:1:40:15  nubian
8:0:20:1:40:7   sahara    # This is a comment
8:0:20:1:40:14  tenere
```

Other Network Databases

The remaining network databases seldom need to be edited.

`networks` database

The `networks` database associates network names with network numbers, enabling some applications to use and display names rather than numbers. The `networks` database is based on information in the `/etc/inet/networks` file. This file contains the names of all networks to which your network connects through routers.

The Solaris installation program configures the initial `networks` database. However, if you add a new network to your existing network topology, you must update this database.

The `networks`(4) man page contains full syntax information for `/etc/inet/networks`. The man page's basic format follows:

network-name network-number nickname(s) #comment

network-name is the official name for the network.

network-number is the number assigned by the InterNIC.

nickname is any other name by which the network is known.

#comment is any note that you want to append to an entry in the file.

You must maintain the `networks` file. The `netstat` program uses the information in this database to produce status tables.

A sample `/etc/networks` file follows.

EXAMPLE 5–8 `/etc/networks` File

```
#ident    "@(#)networks    1.4    92/07/14 SMI"    /* SVr4.0 1.1    */
#
# The networks file associates Internet Protocol (IP) network
# numbers with network names. The format of this file is:
#
#     network-name              network-number              nicnames . . .

# The loopback network is used only for intra-machine communication
loopback              127

#
# Internet networks
#
arpanet     10       arpa  # Historical
ucb-ether   46       ucbether
#
# local networks

eng   193.9.0  #engineering
acc   193.9.1  #accounting
prog  193.9.2  #programming
```

`protocols` Database

The `protocols` database lists the TCP/IP protocols that are installed on your system and their numbers. The Solaris installation program automatically creates the database. This file seldom requires any administration.

The `protocols` database contains the names of the TCP/IP protocols that are installed on the system. The `protocols`(4) man page describes the syntax of this database. An example of the `/etc/inet/protocols` file follows.

EXAMPLE 5–9 `/etc/inet/protocols` File

```
#
# Internet (IP) protocols
#
ip    0   IP    # internet protocol, pseudo protocol number
icmp  1   ICMP  # internet control message protocol
tcp   6   TCP   # transmission control protocol
udp  17   UDP   # user datagram protocol
```

services Database

The services database lists the names of TCP and UDP services and their well-known port numbers. This database is used by programs that call network services. The Solaris installation automatically creates the services database. Generally, this database does not require any administration.

The services(4) man page contains complete syntax information. An excerpt from a typical /etc/inet/services file follows.

EXAMPLE 5–10 /etc/inet/services File

```
#
# Network services
#
echo      7/udp
echo      7/tcp
discard   9/udp     sink null
discard   11/tcp
daytime   13/udp
daytime   13/tcp
netstat   15/tcp
ftp-data  20/tcp
ftp       21/tcp
telnet    23/tcp
time      37/tcp    timeserver
time      37/udp    timeserver
name      42/udp    nameserver
whois     43/tcp    nickname
```

Booting Processes

Note – The names of startup scripts might change from one release to another.

1. You start the operating system on a host.
2. The kernel runs /sbin/init, as part of the booting process.
3. /sbin/init runs the /etc/rcS.d/S30rootusr.sh. startup script.
4. The script runs a number of system startup tasks, including the establishment of the minimum host and network configurations for diskless and dataless operations. /etc/rcS.d/S30rootusr.sh also mounts the /usr file system.
   a. If the local database files contain the required configuration information (host name and IP address), the script uses it.

b. If the information is not available in local host configuration files, `/etc/rcS.d/S30rootusr.sh` uses RARP to acquire the host's IP address.

5. If the local files contain domain name, host name, and default router address, the machine uses them. If the configuration information is not in local files, then the system uses the Bootparams protocol to acquire the host name, domain name, and default router address. Note that the required information must be available on a network configuration server that is located on the same network as the host. This requirement is necessary because no internetwork communications exist at this point.
6. After `/etc/rcS/S30rootusr.sh` completes its tasks and several other boot procedures have executed, `/etc/rc2.d/S69inet` runs. This script executes startup tasks that must be completed before the name services (NIS, NIS+, or DNS) can start. These tasks include configuring the IP routing and setting the domain name.
7. At completion of the `S69inet` tasks, `/etc/rc2.d/S71rpc` runs. This script starts the NIS, NIS+, or DNS name service.
8. After `/etc/rc2.d/S71` runs, `/etc/rc2.d/S72inetsvc` runs. This script starts up services that depend on the presence of the name services. `S72inetsvc` also starts the daemon `inetd`, which manages user services such as `telnet`.

See *System Administration Guide: Basic Administration* for a complete description of the booting process.

Routing Protocols

Solaris system software supports two routing protocols: Routing Information Protocol (RIP) and ICMP Router Discovery (RDISC). RIP and RDISC are both standard TCP/IP protocols.

Routing Information Protocol (RIP)

RIP is implemented by `in.routed`, the routing daemon, which automatically starts when the machine 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.rdisc`, which should run on both routers and hosts. Normally, when `in.rdisc` runs on a host, `in.rdisc` enters a default route for each router that is also running `in.rdisc`. A host that is running `in.rdisc` cannot discover routers that are running only RIP. Furthermore, when routers are running `in.rdisc` (rather than `in.routed`), they can be configured to have a different preference, which causes hosts to select a better router. See the `rdisc`(1M) man page.

How a Machine Determines if It Is a Router

The `/etc/rc2.d/S69inet` startup script, which runs when the machine boots, determines whether a machine is a router or a host. This decision also determines whether the routing protocols (RIP and RDISC) should run in router mode or host mode.

The `/etc/rc2.d/S69inet` script concludes that a machine is a router if the following two conditions exist:

- More than one `/etc/hostname.`*interface* file exists.
- More than one interface was configured "up" by the `ifconfig` command. See the `ifconfig`(1M) man page.

If only one interface is found, the script concludes that the machine is a host. See "Configuring Both Router Network Interfaces" on page 69. An interface that is configured by any means other than an `/etc/hostname.`*interface* file is not considered.

Parts of the IPv4 Address

Each network that runs TCP/IP must have a unique network number. Every machine on the network must have a unique IP address. You must understand how IP addresses are constructed before you register your network and obtain its network number. This section describes IPv4 addresses. For information on IPv6 addresses, see "IPv6 Addressing" on page 282.

The IPv4 address is a 32-bit number that uniquely identifies a network interface on a machine. An IPv4 address is typically written in decimal digits, formatted as four 8-bit fields that are separated by periods. Each 8-bit field represents a byte of the IPv4 address. This form of representing the bytes of an IPv4 address is often referred to as the *dotted-decimal format*.

The bytes of the IPv4 address are further classified into two parts: the network part and the host part. The following figure shows the component parts of a typical IPv4 address, 129.144.50.56.

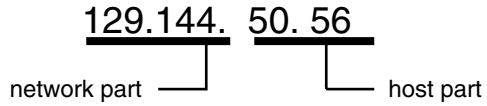


FIGURE 5–3 Parts of an IPv4 Address

Network Part

The network part specifies the unique number that is assigned to your network. The network part also identifies the class of network that is assigned. In Figure 5–3, the network part occupies two bytes of the IPv4 address.

Host Part

This is the part of the IPv4 address that you assign to each host. The host part uniquely identifies this machine on your network. Note that for each host on your network, the network part of the address is the same, but the host part must be different.

Subnet Number (Optional)

Local networks with large numbers of hosts are sometimes divided into subnets. If you choose to divide your network into subnets, you need to assign a *subnet number* for the subnet. You can maximize the efficiency of the IPv4 address space by using some of the bits from the host number part of the IPv4 address as a network identifier. When used as a network identifier, the specified part of the address becomes the subnet number. You create a subnet number by using a netmask, which is a bitmask that selects the network and subnet parts of an IPv4 address. Refer to "Creating the Network Mask for IPv4 Addresses" on page 94 for details.

Network Classes

The first step in planning for IPv4 addressing on your network is to determine which network class is appropriate for your network. After you have completed this step, you can move to the crucial second step: obtain the network number from the InterNIC addressing authority.

Currently there are three classes of TCP/IP networks. Each class uses the 32-bit IPv4 address space differently, providing more or fewer bits for the network part of the address. These classes are class A, class B, and class C.

Class A Network Numbers

A class A network number uses the first 8 bits of the IPv4 address as its "network part." The remaining 24 bits compose the host part of the IPv4 address, as the following figure illustrates.

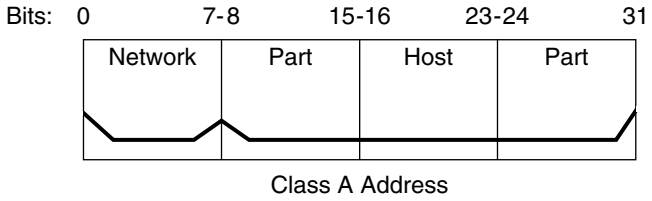


FIGURE 5–4 Byte Assignment in a Class A Address

The values that are assigned to the first byte of class A network numbers fall within the range 0–127. Consider the IPv4 address 75.4.10.4. The value 75 in the first byte

indicates that the host is on a class A network. The remaining bytes, 4.10.4, establish the host address. The InterNIC assigns only the first byte of a class A number. Use of the remaining three bytes is left to the discretion of the owner of the network number. Only 127 class A networks can exist. Each one of these numbers can accommodate a maximum of 16,777,214 hosts.

Class B Network Numbers

A class B network number uses 16 bits for the network number and 16 bits for host numbers. The first byte of a class B network number is in the range 128–191. In the number 129.144.50.56, the first two bytes, 129.144, are assigned by the InterNIC, and compose the network address. The last two bytes, 50.56, compose the host address, and are assigned at the discretion of the owner of the network number. The following figure graphically illustrates a class B address.

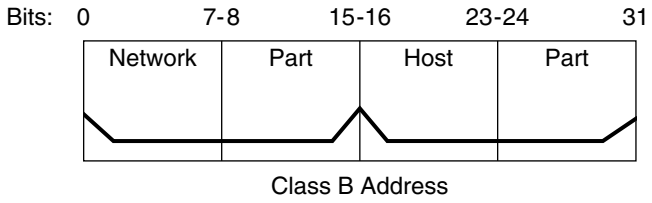


FIGURE 5–5 Byte Assignment in a Class B Address

Class B is typically assigned to organizations with many hosts on their networks.

Class C Network Numbers

Class C network numbers use 24 bits for the network number and 8 bits for host numbers. Class C network numbers are appropriate for networks with few hosts—the maximum being 254. A class C network number occupies the first three bytes of an IPv4 address. Only the fourth byte is assigned at the discretion of the network owners. The following figure graphically represents the bytes in a class C address.

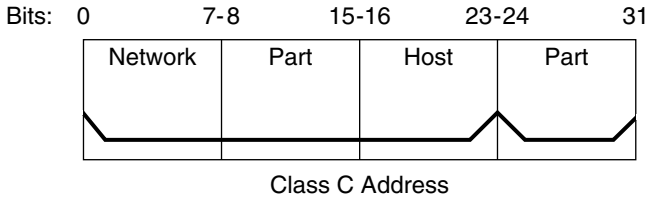


FIGURE 5–6 Byte Assignment in a Class C Address

The first byte of a class C network number covers the range 192–223. The second and third bytes each cover the range 1– 255. A typical class C address might be 192.5.2.5. The first three bytes, 192.5.2, form the network number. The final byte in this example, 5, is the host number.

CHAPTER 6

DHCP Topics

Chapter 7	Provides overview information for DHCP
Chapter 8	Provides instructions for planning for to use DHCP
Chapter 9	Provides step-by-step instructions for configuring DHCP
Chapter 10	Provides step-by-step instructions for administering DHCP
Chapter 11	Provides troubleshooting instructions for DHCP
Chapter 12	Provides background information for DHCP

CHAPTER 7

About Solaris DHCP (Overview)

This chapter introduces the Dynamic Host Configuration Protocol (DHCP), explains the concepts underlying the protocol, and describes the advantages of using it 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 they 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 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 client and sever, but includes more information in the messages. The additional information is the network configuration data for the client.

A primary benefit of DHCP is its ability to manage IP address assignments through leasing, which allows IP addresses to be reclaimed when not in use and reassigned to other clients. This enables a site to use a smaller pool of IP address than would be needed if all clients were assigned a permanent address.

Advantages of Using Solaris DHCP

DHCP relieves the system or network administrator of some of the time-consuming tasks involved in setting up a TCP/IP network and the daily management of that network. Note that Solaris DHCP works only with IPv4.

Solaris DHCP offers the following advantages:

- **IP address management** – A primary advantage of DHCP is easier management of IP addresses. In a network without DHCP, an administrator must manually assign IP addresses, being careful to assign unique IP addresses to each client and configure each client individually. If a client moves to a different network, the administrator 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** – A network administrator can create a tailored configuration for certain clients, or certain types of clients, and keep the information in one place, the DHCP data store. The administrator does not need to log in to a client to change its configuration. The administrator 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 and remote clients** – BOOTP provides for the relaying of messages from one network to another. 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 a server that is not on the client's network. DHCP requests can be relayed in the same manner because, to the router, they are indistinguishable from BOOTP requests. The Solaris 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 needed to boot from a server on the network, instead of using RARP (Reverse Address Resolution Protocol) and `bootparams`. The DHCP server can give a client all the information it needs to function, including IP address, boot server, and network configuration information. Because DHCP network boot 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 has a boot server.
- **Large network support** - Networks with millions of DHCP clients can use Solaris DHCP. The DHCP server uses multithreading to process many client requests

simultaneously and supports data stores optimized to handle large amounts of data. Data store access is handled by separate processing modules, and sites can add support for any database they want to use for their DHCP data.

How DHCP Works

The DHCP server must first be installed and configured by a system administrator. During configuration, the administrator enters information about the network that clients will 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.

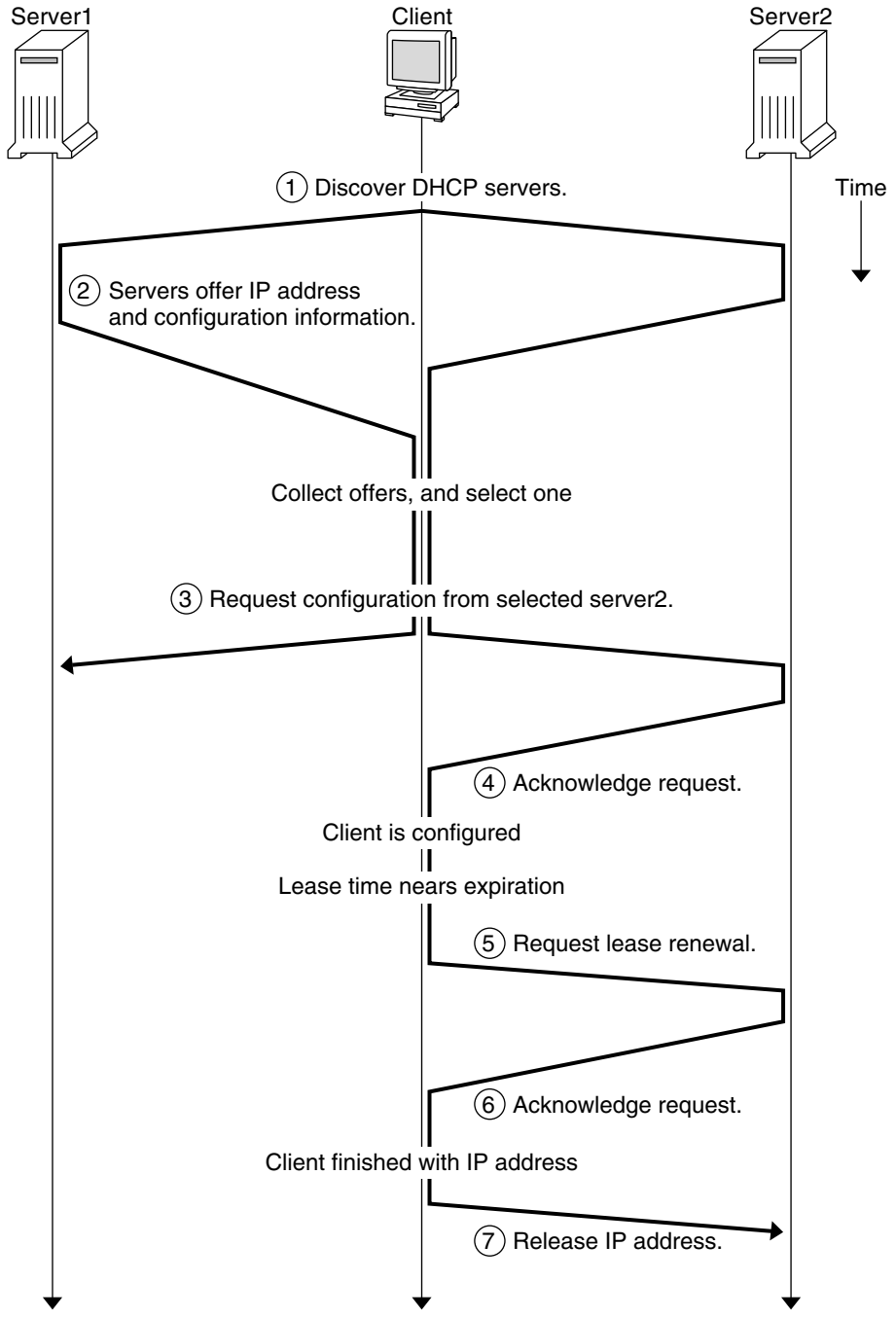


FIGURE 7–1 Sequence of Events for DHCP Service

LEGEND:

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 Solaris DHCP implementation, 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? This tells the server that the client is either 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, it 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 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 select an IP address appropriate for the client. (See `netmasks`(4) for information on `netmasks`.)
2. After they determine the client's network, DHCP 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 that 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 it can determine if the client will use it.
3. The client selects the best offer (based on the number and type of services offered) and 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 the client has chosen a server, and those servers not chosen can cancel the reservations for the IP addresses they had offered.
4. The selected server allocates the IP address for the client, stores the information in the DHCP data store, and sends an acknowledgement (ACK) to the client. The acknowledgement message contains the network configuration parameters for the client. The client uses `ping` to test the IP address to make sure no other system is using it, then continues booting to join the network.
5. The client monitors the lease time, and when a set period of time has elapsed, the client sends a new message to the chosen server to increase its lease time.

6. The DHCP server that receives the request extends the lease time if it 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, it notifies the server that it is releasing the IP address. This can happen during an orderly shutdown and can also be done manually.

Solaris DHCP Server

The Solaris DHCP server runs as a daemon in the Solaris operating environment on a host system. The server has two basic functions:

- **Managing IP addresses** – The server controls a range of IP addresses, and allocates them to clients, either permanently or for a defined period of time. The DHCP server uses a lease mechanism to determine how long a client can use a nonpermanent address. When the address is no longer in use, it is returned to the pool and can be reassigned. The server maintains information about the binding of IP addresses to clients in its DHCP network tables, ensuring that no address is used by more than one client.
- **Providing network configuration for clients** – The server assigns an IP address and provides other information for network configuration, such as a hostname, broadcast address, network subnet mask, default gateway, name service, and potentially much more information. The network configuration information is obtained from the server's `dhcptab` database.

The Solaris DHCP server can also be configured to perform the following additional functions:

- **Responding to BOOTP client requests** – The server listens for broadcasts from BOOTP clients discovering a BOOTP server and provides them with an IP address and boot parameters. The information must have been configured statically by an administrator. The DHCP server can perform as a BOOTP server and DHCP server simultaneously.
- **Relaying requests** – The server relays BOOTP and DHCP requests to appropriate servers on other subnets. The server cannot provide DHCP or BOOTP service when configured as a BOOTP relay agent.
- **Providing network booting support for DHCP clients** – The server can provide DHCP clients with information needed to boot over the network: IP address, boot parameters, and network configuration information.
- **Updating DNS tables for clients that supply a host name** – For clients that provide a Hostname option and value in their requests for DHCP service, the

server can attempt DNS updates on their behalf.

DHCP Server Management

As superuser, you can start, stop, and configure the DHCP server with the DHCP Manager, or with command-line utilities described in "DHCP Command-Line Utilities" on page 123. Generally, the DHCP server is configured to start automatically when the system boots, and stop when the system is shut down. You should not need to start and stop the server manually under normal conditions.

DHCP Data Store

All the data used by the Solaris DHCP server is maintained in a data store, which might be stored as plain text files, NIS+ tables, or binary-format files. While configuring the DHCP service, the administrator chooses the type of data store to be used. The section "Choosing the Data Store" on page 142 describes the differences between the data stores. Data stores can be converted from one format to another using DHCP Manager or the `dhcpconfig` command.

You can also move data from one DHCP server's data store to another with export and import utilities that work with the data stores, even if the servers are using different data store formats. The entire content of a data store, or just some of the data within it, can be exported and imported using DHCP Manager or the `dhcpconfig` command.

Note – Any database or file format can be used for DHCP data storage if you want to develop your own code module to provide an interface between Solaris DHCP (server and management tools) and the database. *Solaris DHCP Service Developer's Guide* contains information for doing this.

Within the Solaris DHCP data store are two types of tables, the contents of which you can view and manage by using either the DHCP Manager or command-line utilities. The data tables are:

- `dhcptab` table – Table of configuration information that can be passed to clients.
- **DHCP network tables** – Tables that contain information about the DHCP and BOOTP clients that reside on the network specified in the table name. For example, the network 134.20.0.0 would have a table whose name includes `134_20_0_0`.

The dhcptab Table

The dhcptab table contains all the information that clients can obtain from the DHCP server. The DHCP server scans the dhcptab each time it starts. The file name of the dhcptab varies according to the data store used. For example, the dhcptab created by the NIS+ data store SUNWnisplus is SUNWnisplus1_dhcptab.

The DHCP protocol defines a number of standard items of information that can be passed to clients. These items are referred to as parameters, symbols, or options. Options are defined in the DHCP protocol by numeric codes and text labels, but without values. Some commonly used standard options are shown in the following table.

TABLE 7–1 Sample DHCP Standard Options

Code	Label	Description
1	Subnet	Subnet mask IP address
3	Router	IP address for router
6	DNSserv	IP address for DNS server
12	Hostname	Text string for client host name
15	DNSdmain	DNS domain name

Some options are automatically assigned values when the administrator provides information during server configuration. The administrator can also explicitly assign values to other options at a later time. Options and their values are passed to the client to provide configuration information. For example, the option/value pair, DNSdmain=Georgia.Peach.COM, sets the client's DNS domain name to Georgia.Peach.COM.

Options can be grouped with other options in containers known as macros, which makes it easier to pass information to a client. Some macros are created automatically during server configuration, and contain options that were assigned values during configuration. Macros can also contain other macros.

The format of the dhcptab table is described in dhcptab(4) man page. In DHCP Manager, all the information shown in the Options and Macros tabs comes from the dhcptab table. See "About Options" on page 126 for more information about options, and "About Macros" on page 127 for more information about macros.

Note that the dhcptab table should not be edited manually. You should use either the dhtadm command or DHCP Manager to create, delete, or modify options and macros.

DHCP Network Tables

A DHCP network table maps client identifiers to IP addresses and the configuration parameters associated with each address. The format of the network tables is described in the `dhcp_network`(4) man page. In DHCP Manager, all the information shown in the Addresses tab is acquired from the network tables.

DHCP Manager

DHCP Manager is a graphical tool you can use to perform all management duties associated with DHCP services, and you must be root when you run it. You can use it to manage the server itself as well as the data the server uses. You can use DHCP Manager with the server in the following ways:

- Configure and unconfigure the DHCP server
- Start, stop, and restart the DHCP server
- Disable and enable DHCP service
- Customize server settings

DHCP Manager allows you to manage the IP addresses, network configuration macros, and network configuration options in the following ways:

- Add and delete networks under DHCP management
- View, add, modify, delete, and release IP addresses under DHCP management
- View, add, modify, and delete network configuration macros
- View, add, modify, and delete nonstandard network configuration options

DHCP Manager allows you to manage the DHCP data stores in the following ways:

- Convert data to a new data store format
- Move DHCP data from one DHCP server to another by exporting it from the first server and importing it on the second server

DHCP Manager includes extensive online help for procedures you can perform with the tool.

DHCP Command-Line Utilities

All DHCP management functions can be performed using command-line utilities. You can run them if you are logged in as root, or as a user assigned to the DHCP Management profile. See “Setting Up User Access to DHCP Commands” on page 169.

The following table lists the utilities and describes the purpose of each utility.

TABLE 7–2 DHCP Command-Line Utilities

Command	Description and Purpose
`in.dhcpd`	The DHCP service daemon. It provides command-line arguments that allow you to set several runtime options.
`dhcpconfig`	Used to configure and unconfigure a DHCP server. This utility enables you to perform many of the functions of DHCP Manager from the command line. It is primarily intended for use in scripts for sites that want to automate some configuration functions. `dhcpconfig` collects information from the server system's network topology files to create useful information for the initial configuration.
`dhtadm`	Used to add, delete, and modify configuration options and macros for DHCP clients. This utility lets you edit the `dhcptab` indirectly, which ensures the correct format of the `dhcptab`. You should not directly edit the `dhcptab`.
`pntadm`	Used to manage the DHCP network tables. You can use this utility to add and remove IP addresses and networks under DHCP management, modify the network configuration for specified IP addresses, and display information about IP addresses and networks under DHCP management.

Role-Based Access Control for DHCP Commands

Security for the `dhcpconfig`, `dhtadm`, and `pntadm` commands is determined by role-based access control (RBAC) settings. By default, the commands can be run only by root. If you want to be able to use the commands under another user name, you must assign the user name to the DHCP Management profile as described in "Setting Up User Access to DHCP Commands" on page 169.

DHCP Server Configuration

You configure the DHCP server the first time you run DHCP Manager on the system where you want to run the DHCP server. DHCP Manager server configuration dialogs prompt you for essential information needed to enable and run the DHCP server on one network. Some default values are obtained from existing system files. If you have not configured the system for the network, there will be no default values. DHCP Manager prompts for the following information:

- Role of the server, either DHCP server or BOOTP relay agent

- Data store type (files, binary files, NIS+, or something specific to your site)
- Data store configuration parameters, which vary according to the data store type you selected
- Naming service to use to update host records, if any (`/etc/hosts`, NIS+, or DNS)
- Length of lease time and whether clients should be able to renew leases
- DNS domain name and IP addresses of DNS servers
- Network address and subnet mask for the first network you want to be configured for DHCP service
- Network type, either LAN or point-to-point
- Router discovery or the IP address of a particular router
- NIS domain name and IP address of NIS servers
- NIS+ domain name and IP address of NIS+ servers

You can also configure the DHCP server using the `dhcpconfig` command. This utility gathers information from existing system files automatically in order to provide a useful initial configuration. Therefore, you must ensure that the files are correct before running `dhcpconfig`. See the `dhcpconfig`(1M) man page for information about the files `dhcpconfig` uses to obtain information.

IP Address Allocation

The Solaris DHCP server supports the following types of IP address allocation:

- **Manual allocation** – The server provides a specific IP address chosen by the administrator for a specific DHCP client. The address cannot be reclaimed or assigned to any other client.
- **Automatic, or permanent, allocation** – The server provides an IP address that has no expiration time, making it permanently associated with the client until the administrator changes the assignment or the client releases the address.
- **Dynamic allocation** – The server provides an IP address to a requesting client, with a lease for a specific period of time. When the lease expires, the address is taken back by the server and can be assigned to another client. The period of time is determined by the lease time configured for the server.

Network Configuration Information

The administrator determines what information to provide to DHCP clients. When you configure the DHCP server you provide essential information about the network. Later, you can add more information you want to provide to clients.

The DHCP server stores network configuration information in the `dhcptab` database, in the form of option/value pairs and macros. Options are keywords for network data you want to supply to clients. Values are assigned to options and passed to clients in DHCP messages. For example, the NIS server address is passed by way of an option called `NISservs` that has a value (a list of IP addresses) assigned by the DHCP server. Macros provide a convenient way to group together any number of options that you want to supply to clients. You can use the DHCP Manager to create macros to group options and assign values to the options. If you prefer a nongraphical tool, you can use `dhtadm`, the DHCP configuration table management utility, to work with options and macros.

About Options

In Solaris DHCP, an option is a piece of network information to be passed to a client. The DHCP literature also refers to options as symbols or tags. An option is defined by a numeric code and a text label. An option receives a value when it is used in the DHCP service.

The DHCP protocol defines a large number of standard options for commonly specified network data: `Subnet`, `Router`, `Broadcast`, `NIS+dom`, `Hostname`, and `LeaseTim` are a few examples. A complete list of standard options is shown in the `dhcp_inittab` man page. You cannot modify the standard option keywords in any way, but you can assign values to the options that are relevant to your network when you include the options in macros.

You can create new options for data that is not represented by the standard options. Options you create must be classified in one of three categories:

- **Extended** – Reserved for options that have become standard DHCP options, but are not yet included in the DHCP server implementation. You might use this if you know of a standard option that you want to use, but do not want to upgrade your DHCP server.
- **Site** – Reserved for options that are unique to your site. The system administrator creates these options.
- **Vendor** – Reserved for options that should apply only to clients of a particular class, such as hardware or vendor platform. The Solaris DHCP implementation includes a number of vendor options for Solaris clients. For example, the option `SrootIP4` is used to specify the IP address of a server that a client that boots from the network should use for its root file system.

Chapter 10 includes procedures for creating, modifying, and deleting options.

About Macros

In the Solaris DHCP service, a macro is a collection of network configuration options and the values assigned to them by the system administrator. Macros are created to group options together to be passed to specific clients or types of clients. For example, a macro intended for all clients of a particular subnet might contain option/value pairs for subnet mask, router IP address, broadcast address, NIS+ domain, and lease time.

Macro Processing by the DHCP Server

When the DHCP server processes a macro, it places the network options and values defined in the macro in a DHCP message to a client. The server processes some macros automatically for clients of a particular type.

In order for the server to process a macro automatically, the name of the macro must comply with one of the categories shown in the following table.

TABLE 7–3 Macro Categories for Automatic Processing

Macro Category	Description
Client class	The macro name matches a class of client, indicated by the client machine type and/or operating system. For example, if a server has a macro named `SUNW.Ultra-1`, any client whose hardware implementation is `SUNW,Ultra-1` automatically receives the values in the `SUNW.Ultra-1` macro.
Network address	The macro name matches a DHCP-managed network IP address. For example, if a server has a macro named `10.53.224.0`, any client connected to the 10.53.224.0 network automatically receives the values in the `10.53.224.0` macro.
Client ID	The macro name matches some unique identifier for the client, usually derived from an Ethernet or MAC address. For example, if a server has a macro named `08002011DF32`, the client with the client ID `08002011DF32` (derived from the Ethernet address `8:0:20:11:DF:32`) automatically receives the values in the macro named `08002011DF32`.

A macro with a name that does not use one of the categories listed in Table 7–3 can be processed only if one of the following is true:

- Macro is mapped to an IP address.
- Macro is included in another macro that is processed automatically.
- Macro is included in another macro that is mapped to an IP address.

Note – When you configure a server, a macro that is named to match the server's name is created by default. This server macro is *not* processed automatically for any client because it is not named with one of the name types that cause automatic processing. When you later create IP addresses on the server, the IP addresses are mapped to use the server macro by default.

Order of Macro Processing

When a DHCP client requests DHCP services, the DHCP server determines which macros match the client. The server processes the macros, using the macro categories to determine the order of processing, from the more general to the specific. The macros are processed in the following order:

1. Client class macros – the most general category
2. Network address macros – more specific than Client class
3. Macros mapped to IP addresses – more specific than Network address
4. Client ID macros – the most specific category, pertaining to one client

A macro that is included in another macro is processed as part of the containing macro.

If the same option is included in more than one macro, the value set for that option in the macro with the most specific category is used because it is processed last. For example, if a Network address macro contained the lease time option with a value of 24 hours, and a Client ID macro contained the lease time option with a value of 8 hours, the client would receive a lease time of 8 hours.

Solaris 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 here is a software entity. The Solaris DHCP client is a daemon (`dhcpagent`) that runs in the Solaris operating environment on a system that is configured to receive its network configuration from a DHCP server. DHCP clients from other vendors can also use the services of the Solaris DHCP server. However, this section describes only the Solaris DHCP client.

Notice that the description assumes one network interface. The section "DHCP Client Systems With Multiple Network Interfaces" on page 135 discusses issues important for hosts that have two or more network interfaces.

DHCP Client Installation

The Solaris DHCP client is installed and enabled on a system during installation of the Solaris operating environment when you specify that you want to use DHCP to configure network interfaces. You do not need to do anything else on the Solaris client to use DHCP.

If you want a system that is already running the Solaris operating environment to use DHCP to obtain network configuration information, see "Configuring and Unconfiguring a Solaris DHCP Client" on page 161.

DHCP Client Startup

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.

The presence of the file `/etc/dhcp.`*interface* (for example, `/etc/dhcp.hme0` on a Sun Enterprise Ultra™ system) indicates to the startup scripts that DHCP is to be used on the specified interface. Upon finding a `dhcp.`*interface* file, the startup scripts start the `dhcpagent` daemon.

After startup, `dhcpagent` waits until it receives instructions to configure a network interface. The startup scripts issue the `ifconfig` *interface* `dhcp start` command, which instructs `dhcpagent` to start DHCP as described in "How DHCP Works" on page 117. If commands are contained within the `dhcp.`*interface* file, they are appended to the `dhcp start` option of `ifconfig`. See the `ifconfig`(1M) man page for more information about options used with the `dhcp` option.

How Solaris DHCP Client Manages Network Configuration Information

After the information packet is obtained from a DHCP server, `dhcpagent` configures the network interface and brings it up, controlling the interface for the duration of the lease time for the IP address. The `dhcpagent` daemon maintains the configuration data in an internal table held in memory. The system startup scripts use the `dhcpinfo` command to extract configuration option values from the `dhcpagent` daemon's table. The values are used to configure the system and enable it to join the network.

The agent waits passively until a period of time elapses, usually half the lease time, and then requests an extension of the lease from a DHCP server. If the `dhcpagent` daemon finds that the interface is down or the IP address has changed, it does not

control the interface until it is instructed by the `ifconfig` command to do so. If the `dhcpagent` daemon finds that the interface is up and the IP address hasn't changed, it sends a request to the server for a lease renewal. If the lease cannot be renewed, the `dhcpagent` daemon takes down the interface at the end of the lease time.

DHCP Client Management

The Solaris DHCP client does not require management under normal system operation. It automatically starts when the system boots, renegotiates leases, and stops when the system shuts down. You cannot manually start and stop the `dhcpagent` daemon. However, you can use the `ifconfig` command as superuser on the client system to affect the client's management of the network interface if necessary.

`ifconfig` Command Options Used With DHCP Client

The `ifconfig` command enables you to:

- **Start the DHCP client** – The command `ifconfig` *interface* `dhcp start` initiates the interaction between the DHCP client and DHCP server to obtain an IP address and a new set of configuration options. This might be 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 `ifconfig` *interface* `dhcp inform` causes `dhcpagent` to issue a request for network configuration parameters, with the exception of the IP address. This is useful for situations where the network interface has a valid IP address, but the client system needs updated network options. For example, this might be useful if you do not use DHCP to manage IP addresses, but do use it to configure hosts on the network.
- **Request a lease extension** – The command `ifconfig` *interface* `dhcp extend` causes `dhcpagent` to issue a request to renew the lease. This happens automatically, but 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 `ifconfig` *interface* `dhcp release` causes `dhcpagent` to relinquish the IP address used by the network interface. This happens automatically when the lease expires. You might want to issue this command if the lease time is long and you need to take down the network interface for an extended period of time or you want to remove the system from the network.
- **Drop the IP address** – The command `ifconfig` *interface* `dhcp drop` causes `dhcpagent` to take down the network interface without informing the DHCP server that it is doing so. This enables the client to use the same IP address when it reboots.

- **Ping the network interface** – The command `ifconfig` *interface* `dhcp ping` lets you test to see if the interface is under the control of DHCP.
- **View DHCP configuration status of the network interface** – The command `ifconfig` *interface* `dhcp status` displays the current state of the DHCP client. The display indicates the following:
  - If an IP address has been bound to the client
  - Number of requests sent, received, and declined
  - If this is the primary interface
  - Times when the lease was obtained, when it expires, and when attempts to renew it will or did start For example:

```
# ifconfig hme0 dhcp status
Interface  State         Sent  Recv  Declined  Flags
hme0       BOUND            1     1         0   [PRIMARY]
(Began,Expires,Renew)=(08/16/2000 15:27, 08/18/2000 13:31, 08/17/2000 15:24)
```

DHCP Client Parameter File

The file `/etc/default/dhcpagent` on the client system contains tunable parameters for the `dhcpagent` daemon. You can use a text editor to change several parameters that affect client operation. The file is well documented so you should refer to the file for more information, as well as referring to the `dhcpagent` man page.

DHCP Client Shutdown

When the system running the DHCP client shuts down normally, the `dhcpagent` daemon writes the current configuration information to the file `/etc/dhcp/`*interface*`.dhc`. The lease is dropped rather than released, so the DHCP server does not know that the IP address is not in active use.

If the lease is still valid when the system reboots, the DHCP client sends an abbreviated request to use the same IP address and network configuration information it had used before the system rebooted. If the DHCP server permits this, the client 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, the client initiates the DHCP protocol sequence described previously and obtains new network configuration information.

DHCP Client Systems and Name Services

Solaris systems support the following name services: DNS, NIS, NIS+, and a local file store (`/etc/hosts`). Each name service requires some configuration before it is usable. The name service switch configuration file (see `nsswitch.conf`(4)) must also be set up appropriately to indicate the name services to be used.

Before a DHCP client system can use a name service, you must configure the system as a client of the name service.

The following table summarizes issues related to each name service and DHCP, and includes links to documentation that can help you set up clients for each name service.

TABLE 7–4 Name Service Client Setup Information for DHCP Client Systems

Name Service	Client Setup Notes

NIS	If you are *installing* the Solaris operating environment on a client system by using Solaris DHCP, you can use a configuration macro that contains the `NISservs` and `NISdmain` options to pass the IP addresses of NIS servers and the NIS domain name to the client. The client then automatically becomes a NIS client.
	
	If a DHCP client system is already running the Solaris operating environment, the NIS client is not automatically configured on that system when the DHCP server sends NIS information to the client.
	
	If the DHCP server is configured to send NIS information to the DHCP client system, you can see the values given to the client if you use the `dhcpinfo` command on the client as follows:
	
	`# /sbin/dhcpinfo NISdmain`
	
	`# /sbin/dhcpinfo NISservs`
	
	Use the values returned for the NIS domain name and NIS servers when you set up the system as a NIS client.
	
	You set up a NIS client for a Solaris DHCP client system in the standard way, as documented in "Setting Up and Configuring NIS Service" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.
	
	Note – You can write a script that uses `dhcpinfo` and `ypinit` to automate NIS client configuration on DHCP client systems.

TABLE 7–4 Name Service Client Setup Information for DHCP Client Systems *(Continued)*

Name Service	Client Setup Notes
NIS+	If the DHCP client system receives a nonreserved IP address (the address may not always be the same), you must set up the NIS+ client for a DHCP client system in a nonstandard way, which is documented in "Setting Up DHCP Clients as NIS+ Clients" on page 233. This procedure is necessary because NIS+ uses security measures to authenticate requests for service. The security measures depend upon the IP address. If the DHCP client system has been manually assigned an IP address (the client's address is always the same), you can set up the NIS+ client in the standard way, which is documented in "Setting Up NIS+ Client Machines" in *System Administration Guide: Naming and Directory Services (FNS and NIS+)*
`/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 add the host name manually 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 add the client's host name manually on the system.
DNS	If the DHCP client system receives the DNS domain name through DHCP, the client system's `/etc/resolv.conf` file is configured automatically. To actually use DNS on systems that use `/etc/inet/hosts` files, you must modify the `/etc/nsswitch.conf` file to add `dns` to the `hosts` line. See "Setting Up a DNS Client" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for more information about DNS clients. If the client system uses NIS or NIS+ for local name resolution, you should be aware of the following: ■ NIS – If the NIS server allows DNS forwarding (which it does by default), the NIS client system can also use DNS. No further DNS client setup is needed in this case. If the NIS server does not allow DNS forwarding, the client system can use DNS by becoming a DNS client as described in "Setting Up a DNS Client" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*. Note that if the client receives the DNS domain name from the DHCP server, the `/etc/resolv.conf` file needed for a DNS client is configured automatically, so you need only be concerned with the `nsswitch.conf` file. ■ NIS+ – The NIS+ client system can be configured to also use DNS if you edit the `nsswitch.conf`file to add `dns` to the `hosts` line.

Client Host Name Registration

If you let the DHCP server generate host names for the IP addresses you place in the DHCP service, the DHCP server can register those host names in NIS+, `/etc/inet/hosts`, or DNS name services. Host name registration cannot be done in NIS because NIS does not provide a protocol to allow programs to update and propagate NIS maps.

Note – The DHCP server can update DNS with generated host names only if the DNS server and DHCP server are running on the same system.

If a DHCP client provides its host name and the DNS server is configured to allow dynamic updates from the DHCP server, the DHCP server can update DNS on the client's behalf, even if the DNS and DHCP servers are running on different systems. See "Enabling Dynamic DNS Updates by DHCP Server" on page 177 for more information about enabling this feature.

The following table summarizes client host name registration for DHCP client systems with the various name services.

TABLE 7–5 Client Host Name Registration in Name Services

	Who Registers Host Name	
Name Service	**DHCP Generated Host Name**	**DHCP Client Supplied Host Name**
NIS	NIS Administrator	NIS Administrator
NIS+	DHCP tools	DHCP tools
`/etc/hosts`	DHCP tools	DHCP tools
DNS	DHCP tools, if the DNS server runs on the same system as the DHCP server. DNS Administrator, if the DNS sever runs on a different system.	DHCP server, if configured for dynamic DNS updates. DNS Administrator, if DHCP server is not so configured.

Note that Solaris DHCP clients can request particular host names in DHCP requests if configured to do so as described in "How to Enable a Solaris Client to Request Specific Host Name" on page 179. Please consult the documentation for non-Solaris clients to determine if the capability is supported.

DHCP Client Systems With Multiple Network Interfaces

The DHCP client daemon can manage several different interfaces on one system simultaneously, each with 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 and maintains a separate set of network configuration options for each interface. However, although the parameters are stored separately, some of the parameters are global in nature, applying to the system as a whole, rather than to a particular network interface.

Options such as hostname, NIS domain name, and timezone are global parameters and should have the same values for each interface, but these values may differ due to errors in the information entered by the DHCP administrator. To ensure that there is only one answer to a query for a global parameter, only the parameters for the primary network interface are requested. You can insert the word `primary` in the `/etc/dhcp.`*interface* file for the interface you want to be treated as the primary interface.

CHAPTER 8

Planning for DHCP Service (Task)

You can use DHCP services in a network you are creating or in a network that exists. If you are setting up a network, see Chapter 3 before you attempt to set up DHCP services. If the network exists, continue in this chapter.

This chapter describes what you need to do before you set up DHCP service on your network. The information is intended for use with DHCP Manager, although you can also use the command-line utility `dhcpconfig` to set up DHCP service.

This chapter contains the following information:

Preparing Your Network for DHCP Service (Task Map)

Before you set up your network to use DHCP, you must first collect information and make decisions about how you will configure the server(s). Use the following task map to identify the tasks for preparing your network for DHCP.

Task	Description	Instructions
Map your network topology	Determine what services are available on the network and where they are located.	"Mapping Your Network Topology" on page 138
Determine the number of DHCP servers you need	Use the expected number of DHCP clients as a basis for determining the number of DHCP servers you need.	"Determining the Number of DHCP Servers" on page 139
Update system files and `netmasks` table	Reflect the network topology accurately.	"Updating System Files and Netmask Tables" on page 140

Mapping Your Network Topology

If you have not already done so, you should map the physical structure or layout of your network. Indicate the location of routers and clients, and the location of servers that provide network services. This map of your network topology can help you determine which server to use for DHCP services, and what configuration information the DHCP server can provide to clients.

See "Planning Your TCP/IP Network" in *System Administration Guide, Volume 3* for more information about planning your network.

The DHCP configuration process can look up some network information from the server's system and network files. "Updating System Files and Netmask Tables" on page 140 discusses these files. However, you might want to give clients other service information, which you must enter into the server's macros. As you examine your network topology, record the IP addresses of any servers you want your clients to know about. The following are some examples of network services you may have on your network that the DHCP configuration does not discover:

- Time server
- Log server
- Print server
- Install server
- Boot server
- Swap server
- X Window font server
- TFTP server

Network Topology to Avoid

DHCP does not work well in network environments where more than one IP network shares the same network hardware media, either through the use of multiple network

hardware interfaces or multiple logical interfaces. When multiple IP networks run across the same physical LAN, a DHCP client's request arrives on all network hardware interfaces. This makes the client appear to be attached to all of the IP networks simultaneously.

DHCP must be able to determine the address of a client's network in order to assign an appropriate IP address to the client. If more than one network is present on the hardware media, the server cannot determine the client's network and cannot assign an IP address.

You can use DHCP on one of the networks, but not more than one. If this does not suit your needs, you must reconfigure the networks. Suggestions for reconfiguration include:

- Use variable length subnet masks (VLSM) to make better use of the IP address space you have, so you do not need to run multiple LANs on the same physical network. See RFC-1519 for more information on VLSM and Classless Inter-Domain Routing (CDIR).
- Configure the ports on your switches to assign devices to different physical LANs. This preserves the mapping of one LAN to one IP network required for Solaris DHCP. See the documentation for the switch for information about port configuration.

Determining the Number of DHCP Servers

The data store option you choose has a direct effect on the number of servers you must have to support your DHCP clients. The following table shows the maximum number of DHCP/BOOTP clients that can be supported by one DHCP server for each data store.

TABLE 8–1 Estimated Maximum Number of Clients

Data Store	Maximum Number of Clients
Text files	10,000
NIS+	40,000
Binary files	100,000

This maximum number is a general guideline, not an absolute number. A DHCP server's client capacity depends greatly on the number of transactions it must process per second. Lease times and usage patterns have a large effect on the number of clients that a server can support. For example, if leases are set to 12 hours and users turn their systems off at night and on at the same time the next morning, the server must handle transaction peaks each morning as many clients request leases simultaneously. The

DHCP server can support fewer clients in such an environment compared to an environment with longer leases, or an environment that consists of constantly connected devices such as cable modems.

The section “Choosing the Data Store” on page 142 compares data store options.

Updating System Files and Netmask Tables

During the configuration process, DHCP Manager or the `dhcpconfig` utility scans various system files on your server for information it can use to configure the server.

You must be sure the information in the system files is current before you run DHCP Manager or `dhcpconfig` to configure your server. If you notice errors after you configure the server, use DHCP Manager or `dhtadm` to modify the macros on the server.

The following table lists some of the information gathered during DHCP server configuration, and the sources for the information. Be sure this information is set correctly on the server before you configure DHCP on it. If you make changes to the system files after you configure the server, you should reconfigure the service to pick up the changes.

TABLE 8–2 Information for DHCP Configuration

Information	Source	Comments
Time zone	System date, time zone settings	The date and time zone are initially set during the Solaris installation. You can change the date by using the `date` command and change the time zone by editing the `/etc/TIMEZONE` file, which sets the TZ variable.
DNS parameters	`/etc/resolv.conf`	The DHCP server uses the `/etc/resolv.conf` file to look up DNS parameters such as DNS domain name and DNS server addresses. See *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for more information about `resolv.conf`.

TABLE 8–2 Information for DHCP Configuration *(Continued)*

Information	Source	Comments
NIS or NIS+ parameters	System domain name, `nsswitch.conf`, NIS, NIS+	The DHCP server uses the `domainname` command to obtain the domain name of the server system, and the `nsswitch.conf` file to determine where to look for domain-based information. If the server system is a NIS or NIS+ client, the DHCP server queries NIS or NIS+ services to get NIS/NIS+ server IP addresses.
Default router	System routing tables, user prompt	The DHCP server searches the network routing tables to find the default router for clients attached to the local network. For clients not on the same network, the DHCP server must prompt the administrator for the information.
Subnet mask	Network interface, `netmasks` table	The DHCP server looks to its own network interfaces to determine the netmask and broadcast address for local clients. If the request had been forwarded by a relay agent, the server looks up the subnet mask in the `netmasks` table on the relay agent's network.
Broadcast address	Network interface, `netmasks` table	For the local network, the DHCP server obtains the broadcast address by querying the network interface. For remote networks, the server uses the BOOTP relay agent's IP address and the remote network's netmask to calculate the broadcast address for the network.

Making Decisions for DHCP Server Configuration (Task Map)

This section discusses some of the decisions to make before you configure the first DHCP server on your network. Use this task map to identify the decisions you must make.

Task	Description	For Instructions
Select a server for DHCP	Determine if a server meets the system requirements to run the DHCP service.	"Selecting a Server for DHCP" on page 142
Choose a data store	Compare the data store choices to determine the best for your site.	"Choosing the Data Store" on page 142
Set a lease policy	Learn about IP address leases to help you determine appropriate leasing for your site.	"Setting a Lease Policy" on page 143
Select router address or router discovery	Determine whether DHCP clients use a specific router or router discovery	"Determining Routers for DHCP Clients" on page 144

Selecting a Server for DHCP

With your network topology in mind, you can use the following guidelines to select a host on which to set up a DHCP server.

The server must:

- Run the Solaris 2.6, Solaris 7, or Solaris 8 operating environment. You must install the Solaris 8 7/01 operating environment or a later version if you need to support a large number of clients.
- Be accessible to all the networks that have clients that will use DHCP, either directly on the network or through a BOOTP relay agent.
- Be configured to use routing.
- Have a correctly configured `netmasks` table that reflects your network topology.

Choosing the Data Store

You can choose to store the DHCP data in text files, binary files, or the NIS+ directory service. The following table summarizes the features of each type of data store, and recommends the environment to which each is best suited.

TABLE 8–3 Comparison of Data Stores

Data Store Type	Performance	Maintenance	Sharing	Recommended Environment
Binary files	High performance, high capacity.	Low-maintenance, no database servers required. Contents must be viewed with DHCP Manager or `dhtadm` and `pntadm`. Regular file backups suggested.	Containers cannot be shared among DHCP servers.	Midsize to large environments with many networks with thousands of clients per network. Useful for small to medium ISPs.
NIS+	Moderate performance and capacity, dependent upon NIS+ service's performance and capacity	DHCP server system must be configured as a NIS+ client. Requires NIS+ service maintenance. Contents must be viewed with DHCP Manager or `dhtadm` and `pntadm`. Regular backups with `nisbackup` suggested.	DHCP data is distributed in NIS+, multiple servers can access the same containers.	Small to midsize environments with up to 5000 clients per network.
Text files	Moderate performance, low capacity.	Low-maintanence, no database servers required. ASCII format is readable without DHCP Manager, `dhtadm`, or `pntadm`. Regular file backups suggested.	Containers can be shared among DHCP servers if DHCP data is stored on one file system that is exported though an NFS mount point.	Small environments with a few hundred to a thousand clients per network, less than 10,000 total clients.

Traditional NIS (as opposed to NIS+) is not offered as a data store option because it does not support fast incremental updates. If your network uses NIS, you should use text files or binary files for your data store.

Setting a Lease Policy

A lease specifies the amount of time the DHCP server grants permission to a DHCP client to use a particular IP address. During the initial server configuration, you must specify a site-wide lease policy to indicate the lease time and whether clients can renew their leases. The server uses the information you supply to set option values in

the default macros it creates during configuration. You can set different lease policies for specific clients or type of clients, by setting options in configuration macros you create.

The lease time is specified as a number of hours, days, or weeks for which the lease is valid. When a client is assigned an IP address (or renegotiates a lease on an IP address it is already assigned), the lease expiration date and time is calculated by adding the number of hours in the lease time to the timestamp on the client's DHCP acknowledgment. For example, if the timestamp of the DHCP acknowledgment is September 16, 2001 9:15 A.M., and the lease time is 24 hours, the lease expiration time is September 17, 2001 9:15 A.M. The lease expiration time is stored in the client's DHCP network record, viewable in DHCP Manager or with `pntadm`.

The lease time value should be relatively small, so that expired addresses are reclaimed quickly, but large enough so that if your DHCP service becomes unavailable, the clients continue to function until the system(s) that run the DHCP service can be repaired. A rule of thumb is to specify a time that is two times the predicted down time of a server. For example, if it generally takes four hours to obtain and replace a defective part and reboot the server, you should specify a lease time of eight hours.

The lease negotiation option determines whether or not a client can renegotiate its lease with the server before the lease expires. If lease negotiation is allowed, the client tracks the time that remains in its lease, and when half the lease time is used, the client requests the DHCP server to extend its lease to the original lease time. It is useful to disable lease negotiation in environments where there are more systems than IP addresses, so the time limit is enforced on the use of IP addresses. If there are enough IP addresses, you should enable lease negotiation so you do not force a client to take down its network interface and obtain a new lease, which can interrupt the client's TCP connections (such as NFS and telnet sessions). You can enable lease negotiation site-wide during the server configuration, and for particular clients or types of clients through the use of the `LeaseNeg` option in configuration macros.

Note – Systems that provide services on the network should retain their IP addresses, and should not be subject to short-term leases. You can use DHCP with such systems if you assign them reserved (manual) IP addresses, rather than IP addresses with permanent leases. This enables you to detect when the system's IP address is no longer in use.

Determining Routers for DHCP Clients

Clients use routers for any network communication beyond their local network, and they must know the IP addresses of these routers in order to use them.

When you configure a DHCP server, you must provide the IP address of a router the clients can use or, if you use DHCP Manager, you can specify that clients should find routers themselves with the router discovery protocol.

If clients on your network support router discovery, you should use router discovery protocol, even if there is only one router. Discovery enables a client to adapt easily to router changes in the network. For example, if a router fails and is replaced by one with a new address, clients can discover the new address automatically without having to obtain a new network configuration to get the new router address.

Making Decisions for IP Address Management (Task Map)

As part of the DHCP service setup, you determine several aspects of the IP addresses that the server is to manage. If your network needs more than one DHCP server, you must decide how to divide responsibility for the addresses so you can assign some to each server. The following task map can help you make IP address management decisions.

Task	Description	Information
Specify what addresses the server should manage	Determine how many addresses you want the DHCP server to manage, and what those addresses are.	"Number and Ranges of IP Addresses" on page 145
Decide if server should automatically generate host names for clients	Learn about client host name generation and decide whether to use it.	"Client Host Name Generation" on page 146
Determine what configuration macro to assign to clients	Learn about client configuration macros so you can select an appropriate macro for clients.	"Default Client Configuration Macros" on page 146
Determine lease types to use	Learn about lease types to help you determine what type is best for your DHCP clients	"Dynamic and Permanent Lease Type" on page 147

Number and Ranges of IP Addresses

During the initial server configuration, DHCP Manager allows you to add one block, or range, of IP addresses under DHCP management by specifying the total number of

addresses and the first address in the block. DHCP Manager adds a list of contiguous addresses from this information. If you have several blocks of noncontiguous addresses, you can add the others by running DHCP Manager's Address Wizard again, after the initial configuration.

Before you configure your IP addresses, know how many addresses are in the initial block of addresses you want to add and the IP address of the first address in the range.

Client Host Name Generation

The dynamic nature of DHCP means that an IP address is not permanently associated with the host name of the system that is using it. The DHCP management tools can generate a client name to associate with each IP address if you select this option. The client names consist of a prefix, or root name, plus a dash and a number assigned by the server. For example, if the root name is `charlie`, the client names will be `charlie-1`, `charlie-2`, `charlie-3`, and so on.

By default, generated client names begin with the name of the DHCP server that manages them. This is useful in environments that have more than one DHCP server because you can quickly see in the DHCP network tables which clients any given DHCP server manages. However, you can change the root name to any name you choose.

Before you configure your IP addresses, decide if you want the management tools to generate client names, and if so, what root name to use for the names.

The generated client names can be mapped to IP addresses in `/etc/inet/hosts`, DNS, or NIS+ if you specify this at configuration. See "Client Host Name Registration" on page 134 for more information.

Default Client Configuration Macros

In Solaris DHCP, a macro is a collection of network configuration options and their assigned values. The DHCP server uses macros to determine what network configuration information to send to a DHCP client.

When you configure the DHCP server, the management tools gather information from system files and directly from you through prompts or command-line options you specify. With this information, the management tools create the following macros:

- Network address macro, named to match the IP address of the client network. The macro contains information needed by any client that is part of the network, such as subnet mask, network broadcast address, default router or router discovery token, and NIS/NIS+ domain and server if the server uses NIS/NIS+. Other

options applicable to your network might be included.

- Locale macro, named `Locale`. The macro contains the offset (in seconds) from Universal Time to specify the time zone.
- Server macro, named to match the server's host name. The macro contains information about the lease policy, time server, DNS domain, and DNS server, and possibly other information that the configuration program was able to obtain from system files. This macro includes the `Locale` macro.

The network address macro is automatically processed for all clients located on that network. The locale macro is included in the server macro, so it is processed when the server macro is processed.

When you configure IP addresses for the first network, you must select a client configuration macro to be used for all DHCP clients using the addresses you are configuring. By default, the server macro is selected because it is contains information needed by all clients that use this server. Clients receive the options contained in the network address macro before those in the server macro. See "Order of Macro Processing" on page 128 for more information about the order in which macros are processed.

Dynamic and Permanent Lease Type

The lease type determines if the lease policy applies to the addresses you are configuring. During initial server configuration, DHCP Manager allows you to select either dynamic or permanent leases for the addresses you are adding. If you configure with the `dhcpconfig` command, leases are dynamic.

When an address has a dynamic lease, the DHCP server can manage the address by allocating it to a client, extending the lease time, detecting when it is no longer in use, and reclaiming it. When an address has a permanent lease, the DHCP server can only allocate it to a client, after which the client owns the address until the client explicitly releases it. When the address is released, the server can assign it to another client. The address is not subject to the lease policy as long as it is configured with a permanent lease type.

When you configure a range of IP addresses, the lease type you select applies to all the addresses in the range. To get the most benefit from DHCP, you should use dynamic leases for most of the addresses. You can later modify individual addresses to make them permanent if necessary, but the total number of permanent leases should be kept to a minimum.

Reserved Addresses and Lease Type

Addresses can be reserved by manually assigning them to particular clients. A reserved address can have a permanent or dynamic lease associated with it. When a reserved address is assigned a permanent lease:

- The address can be allocated only to the client that is bound to the address
- The DHCP server cannot allocate the address to another client
- The address cannot be reclaimed by the DHCP server

If a reserved address is assigned a dynamic lease, the address can be allocated only to the client that is bound to the address, but the client must track lease time and negotiate for a lease extension as if the address were not reserved. This allows you to track when the client is using the address by looking at the network table.

You cannot create reserved addresses for all the IP addresses during the initial configuration because they are intended to be used sparingly for individual addresses.

Planning for Multiple DHCP Servers

If you want to configure more than one DHCP server to manage your IP addresses, consider the following guidelines:

- Divide the pool of IP addresses so that each server is responsible for a range of addresses and there is no overlap of responsibility.
- Choose NIS+ as your data store, if available. If not, choose text files and specify a shared directory for the absolute path to the data store. The binary files data store cannot be shared.
- Configure each server separately so that address ownership is allocated correctly and that server-based macros can be automatically created.
- Set up the servers to scan the options and macros in the `dhcptab` table at specified intervals so they each are using the latest information. You can do this by using DHCP Manager to schedule automatic reading of `dhcptab` as described in "Customizing DHCP Service Performance Options" on page 179.
- Be sure all clients can access all DHCP servers so that the servers can support one another. If a client has a valid IP address lease, and is either trying to verify its configuration or extend the lease, but the server that owns the client's address is not reachable, another server can respond to the client after the client has attempted to contact the primary server for 20 seconds. If a client requests a specific address, and the server that owns the address is not available, one of the other servers handles the request. The client receives a different address from the one requested.

Planning for Remote Network Configuration

After the initial configuration, you can place IP addresses in remote networks under DHCP management. However, because the system files are not local to the server, DHCP Manager and `dhcpconfig` cannot look up information to provide default values, so you must provide the information. Before you attempt to configure a remote network, be sure you know the following information:

- Remote network's IP address.
- Subnet mask of the remote network – This can be obtained from the `netmasks` table in the name service. If the network uses local files, look in `/etc/netmasks` on a system in the network. If the network uses NIS+, use the command `niscat netmasks.org_dir`. If the network uses NIS, use the command `ypcat -k netmasks.byaddr`. Make sure the `netmasks` table contains all the topology information for all the subnets you want to manage.
- Network type – Do the clients connect to the network through a local area network (LAN) connection or point-to-point protocol (PPP)?
- Routing – Can the clients use router discovery? If not, you must determine the IP address of a router they can use.
- NIS domain and NIS servers, if applicable.
- NIS+ domain and NIS+ servers, if applicable.

See "Adding DHCP Networks" on page 184 for the procedure for adding DHCP networks.

Selecting the Tool for Configuring DHCP

After you have gathered information and made decisions as outlined in the previous sections, you are ready to configure a DHCP server. You can use the graphical DHCP Manager or the command-line utility `dhcpconfig` to configure a server. DHCP Manager lets you select options and enter data that is then used to create the `dhcptab` and network tables used by the DHCP server. The `dhcpconfig` utility requires you to use command-line options to specify data.

DHCP Manager Features

DHCP Manager, a Java-based graphical tool, provides a DHCP Configuration Wizard, which starts automatically the first time you run DHCP Manager on a system that is not configured as a DHCP server. The DHCP Configuration Wizard provides a series of dialog boxes that prompt you for the essential information required to configure a server: data store format, lease policy, DNS/NIS/NIS+ servers and domains, and router addresses. Some of the information is obtained by the wizard from system files, and you only need to confirm that the information is correct, or correct it if necessary.

When you progress through the dialog boxes and approve the information, and the DHCP server daemon starts on the server system, you are prompted to start the Add Addresses Wizard to configure IP addresses for the network. Only the server's network is configured for DHCP initially, and other server options are given default values. You can run DHCP Manager again after the initial configuration is complete to add networks and modify other server options.

`dhcpconfig` Features

The `dhcpconfig` utility supports a list of options that allow you to configure and unconfigure a DHCP server, as well as convert to a new data store and import/export data to and from other DHCP servers. When you use the `dhcpconfig` utility to configure a DHCP server, it obtains information from the system files discussed in "Updating System Files and Netmask Tables" on page 140. You cannot view and confirm the information it obtains from system files as you can with DHCP Manager, so it is important that the system files be updated before you run `dhcpconfig`. You can also use command-line options to override the values `dhcpconfig` would obtain by default from system files. The `dhcpconfig` command can be used in scripts. Please see the `dhcpconfig` man page for more information.

Comparison of DHCP Manager and `dhcpconfig`

The following table summarizes the differences between the two server configuration tools.

TABLE 8–4 Comparison of DHCP Manager and the `dhcpconfig` Command

Feature	DHCP Manager	`dhcpconfig` With Options
Network information gathered from system.	Allows you to view the information gathered from system files, and change it if needed.	You can specify the network information with command-line options.

TABLE 8–4 Comparison of DHCP Manager and the `dhcpconfig` Command *(Continued)*

Feature	DHCP Manager	`dhcpconfig` With Options
Configuration experience for user.	Speeds the configuration process by omitting prompts for nonessential server options by using default values for them. Allows you to change nonessential options after initial configuration.	Fastest configuration process, but user may need to specify values for many options.

The next chapter includes procedures you can use to configure your server with both DHCP Manager and the `dhcpconfig` utility.

CHAPTER 9

Configuring DHCP Service (Task)

When you configure DHCP service on your network, you configure and start the first DHCP server. Other servers can be added later, and may access the same data from a shared location if the data store supports shared data. This chapter includes procedures to enable you to configure the DHCP server and place networks and their associated IP addresses under DHCP management. It also explains how to unconfigure a server.

This chapter also provides instructions for procedures that use both DHCP Manager and the `dhcpconfig` utility in separate sections. This chapter contains the following information:

Configuring and Unconfiguring a DHCP Server Using DHCP Manager

This section includes procedures to help you configure and unconfigure a DHCP server with DHCP Manager. Note that you must be running an X Window system such as CDE to use DHCP Manager.

When you run DHCP Manager on a server not configured for DHCP, the following screen is displayed to allow you to specify whether you want to configure a DHCP server or a BOOTP relay agent.

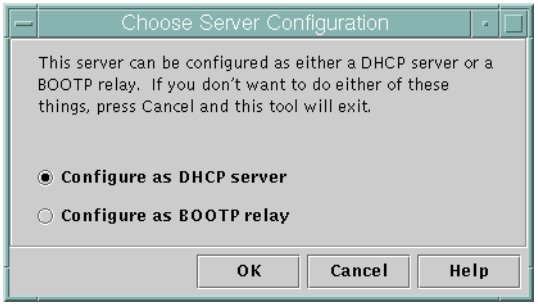


FIGURE 9–1 Choose Server Configuration Dialog Box

Configuring DHCP Servers

When you configure a DHCP server, DHCP Manager starts the DHCP Configuration Wizard, which prompts you for information needed to configure the server. The initial screen of the wizard is shown in the following figure.

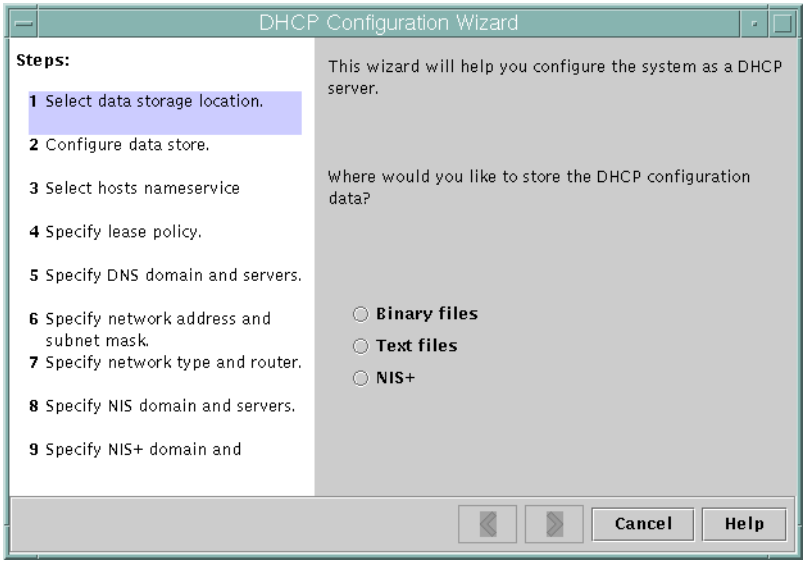


FIGURE 9–2 DHCP Configuration Wizard's Initial Screen

When you finish answering the wizard prompts, DHCP Manager creates the items listed in the following table.

TABLE 9–1 Items Created During DHCP Server Configuration

Item	Description	Contents
Service configuration file, `/etc/inet/dhcpsvc.conf`	Records keywords and values for server configuration options.	Data store type and location, options used with `in.dhcpd` to start the DHCP daemon when system boots.
`dhcptab` table	DHCP Manager creates a `dhcptab` table if it does not already exist.	Macros and options with assigned values.
`Locale` macro, optional	Contains the local time zone's offset in seconds from Universal Time (UTC).	`UTCoffst` option
Server macro, named to match server's node name	Contains options whose values were determined by input from the administrator who configured the DHCP server. Options apply to all clients that use addresses owned by the server.	The `Locale` macro, plus the following options: ■ `Timeserv`, set to point to the server's primary IP address ■ `LeaseTim`, and `LeaseNeg` if you selected negotiable leases ■ `DNSdmain` and `DNSserv`, if DNS is configured ■ `Hostname`, which *must not* be assigned a value. The presence of this option indicates that the hostname must be obtained from the name service.
Network address macro, whose name is the same as the network address of client's network	Contains options whose values were determined by input from the administrator who configured the DHCP server. Options apply to all clients that reside on the network specified by the macro name.	The following options: ■ `Subnet` ■ `Router` or `RDiscvyF` ■ `Broadcst`, if the network is a LAN ■ `MTU` ■ `NISdmain` and `NISservs`, if NIS is configured ■ `NIS+dom` and `NIS+serv`, if NIS+ is configured
Network table for the network.	Empty table is created until you create IP addresses for the network.	None, until you add IP addresses.

▼ How to Configure a DHCP Server (DHCP Manager)

1. **Select the system you want to use as a DHCP server.**

   Use the guidelines in "Making Decisions for DHCP Server Configuration (Task Map)" on page 141.

2. **Make decisions about your data store, lease policy, and router information.**

   Use the guidelines in "Making Decisions for DHCP Server Configuration (Task Map)" on page 141.

3. **Become superuser on the server system.**

4. **Type the following command:**

   ```
   #/usr/sadm/admin/bin/dhcpmgr &
   ```

5. **Choose the option Configure as DHCP Server.**

   This starts the DHCP Configuration Wizard, which helps you configure your server.

6. **Select options or type requested information based on the decisions you made in the planning phase.**

   If you have difficulty, click Help in the wizard window to open your web browser and display help for the DHCP Configuration Wizard.

7. **Click Finish to complete the server configuration when you have finished entering the requested information.**

8. **At the Start Address Wizard window, click Yes to configure addresses for the server.**

   The Address Wizard enables you to specify which addresses to place under the control of DHCP.

9. **Answer the prompts based on decisions made in the planning phase.**

   See "Making Decisions for IP Address Management (Task Map)" on page 145 for more information. If you have difficulty, click Help in the wizard window to open your web browser and display help for the Add Addresses Wizard.

10. **Review your selections, and click Finish to add the addresses to the network table.**

    The network table is updated with records for each address in the range you specified.

    You can add more networks to the DHCP server with the Network Wizard, as explained in "Adding DHCP Networks" on page 184.

Configuring BOOTP Relay Agents

When you configure a BOOTP relay agent, DHCP Manager takes the following actions:

- Prompts you for IP addresses of the DHCP server to which requests should be relayed.
- Edits `/etc/inet/dhcpsvc.conf`, to specify the options needed for BOOTP relay service.

The following figure shows the screen displayed when you choose to configure a BOOTP relay agent.

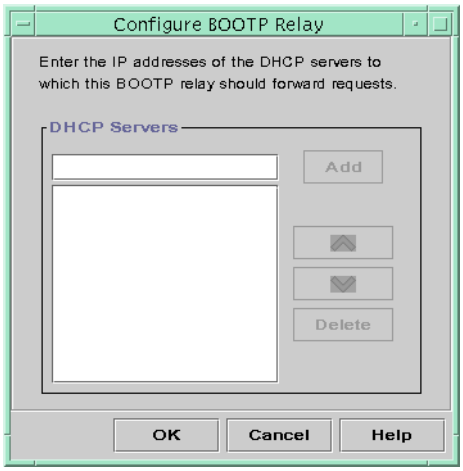


FIGURE 9–3 Configure BOOTP Relay Dialog Box

▼ How to Configure a BOOTP Relay Agent (DHCP Manager)

1. **Select the system you want to use as a BOOTP relay agent.**
   See "Selecting a Server for DHCP" on page 142.
2. **Become superuser on the server system.**
3. **Type the following command:**

   ```
   #/usr/sadm/admin/bin/dhcpmgr &
   ```

   If the system has not been configured as a DHCP server or BOOTP relay agent, the DHCP Configuration Wizard starts. If the system has already been configured as a DHCP server, you cannot configure it as a BOOTP relay agent unless you unconfigure

the server first. See “Unconfiguring DHCP Servers and BOOTP Relay Agents” on page 158.

4. **Select Configure as BOOTP Relay.**

   The Configure BOOTP Relay dialog box opens.

5. **Type the IP address or host name of one or more DHCP servers that are configured to handle BOOTP or DHCP requests received by this BOOTP relay agent, and click Add.**

6. **Click OK to exit the dialog box.**

   Notice that the DHCP Manager offers only the File menu to exit the application and the Service menu to manage the server. Other menu options are disabled because they are useful only on a DHCP server.

Unconfiguring DHCP Servers and BOOTP Relay Agents

When you unconfigure a DHCP server or BOOTP relay agent, DHCP Manager takes the following actions:

- Stops the DHCP daemon (`in.dhpcd`) process
- Removes the `/etc/inet/dhcpsvc.conf` file, which records information about daemon startup and the data store location

The following figure shows the screen that is displayed when you choose to unconfigure a DHCP server.

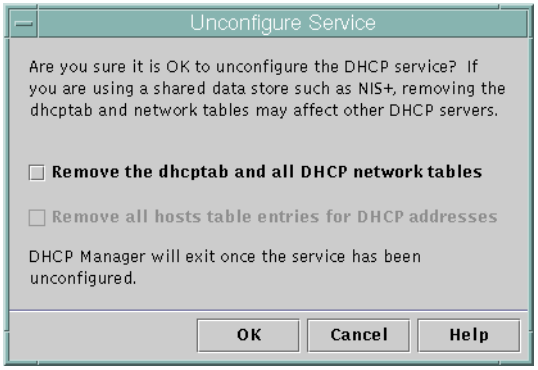


FIGURE 9–4 Unconfigure Service Dialog Box

DHCP Data on an Unconfigured Server

When you unconfigure a DHCP server you must decide what to do with the `dhcptab` table and the DHCP network tables. If the data is shared among servers, you should not remove the `dhcptab` and DHCP network tables because this would render DHCP unusable across your network. Data can be shared through NIS+ or on exported local file systems. The file `/etc/inet/dhcpsvc.conf` records the data store used and its location.

You can unconfigure a DHCP server but leave the data intact by not selecting any of the options to remove data. If you unconfigure the server and leave the data intact, you disable the DHCP server.

If you want another DHCP server to take ownership of the IP addresses belonging to the server you are unconfiguring, you must move the DHCP data to the other DHCP server before you unconfigure the current server. See "Moving Configuration Data Between DHCP Servers (Task Map)" on page 238 for more information.

If you are certain you want to remove the data, you can select an option to remove the `dhcptab` and network tables. If you had generated client names for the DHCP addresses, you can also elect to remove those entries from the hosts table (whether in DNS, `/etc/inet/hosts`, or NIS+).

Before you unconfigure a BOOTP relay agent, be sure that no clients rely on this agent to forward requests to a DHCP server.

▼ How to Unconfigure a DHCP Server or BOOTP Relay Agent (DHCP Manager)

1. **Become superuser.**
2. **Type the following command:**

   ```
   #/usr/sadm/admin/bin/dhcpmgr &
   ```

3. **From the Service menu, choose Unconfigure.**

   The Unconfigure Service dialog box is displayed. If the server is a BOOTP relay agent, the dialog box enables you to confirm your intention to unconfigure the relay agent. If the server is a DHCP server, you must decide what to do with the DHCP data and make selections in the dialog box. See Figure 9–4.

4. **(Optional) Select options to remove data.**

   If the server uses shared data (through NIS+ or files shared through NFS), do not select any options to remove the data. If the server does not use shared data, select one or both options to remove the data.

See "DHCP Data on an Unconfigured Server" on page 159 for more information about removing data.

5. **Click OK to confirm.**

Configuring and Unconfiguring a DHCP Server Using `dhcpconfig` Commands

This section includes procedures to help you configure and unconfigure a DHCP server or BOOTP relay agent by using `dhcpconfig` with command-line options.

▼ How to Configure a DHCP Server (`dhcpconfig -D`)

1. **Select the system you want to use as a DHCP server.**

   Use the guidelines in "Making Decisions for DHCP Server Configuration (Task Map)" on page 141.

2. **Make decisions about your data store, lease policy, and router information.**

   Use the guidelines in "Making Decisions for DHCP Server Configuration (Task Map)" on page 141.

3. **Become superuser or a user assigned to the DHCP Management profile.**

4. **Type a command of the following format:**

   # **`/usr/sbin/dhcpconfig -D -r`** *datastore* **`-p`** *location*

   datastore is one of `SUNWfiles`, `SUNWbinfiles`, or `SUNWnisplus`.

   location is the data-store-dependent location where you want to store the DHCP data. For `SUNWfiles` and `SUNWbinfiles`, this must be a UNIX absolute path name. For `SUNWnisplus`, this must be a fully specified NIS+ directory.

   The `dhcpconfig` utility uses the server machine's system and network files to determine values used to configure the DHCP server. See the `dhcpconfig` man page for information about additional options to the `dhcpconfig` command that enable you to override the default values.

5. **Add one or more networks to the DHCP service.**

   See "How to Add a DHCP Network (`dhcpconfig`)" on page 185 for the procedure to add a network.

▼ How to Configure a BOOTP Relay Agent (`dhcpconfig -R`)

1. **Select the system you want to use as a BOOTP relay agent.**

   Use the guidelines in "Making Decisions for DHCP Server Configuration (Task Map)" on page 141.

2. **Become superuser or a user assigned to the DHCP Management profile.**

3. **Type the following command:**

   # **`/usr/sbin/dhcpconfig -R`** *addresses*

   addresses are the comma-separated IP addresses of DHCP servers to which you want requests to be forwarded.

▼ How to Unconfigure a DHCP Server or BOOTP Relay Agent (`dhcpconfig -U`)

1. **Become superuser or a user assigned to the DHCP Management profile.**

2. **Type the following on the system that acts as DHCP server or BOOTP relay agent:**

   # **`/usr/sbin/dhcpconfig -U`**

   If the server does not use shared data (through NIS+ or text files shared through NFS), you can also use the `-x` option to remove the `dhcptab` and network tables. If the server uses shared data, do not use the `-x` option. The `-h` option can be used to remove host names from the host table. See the `dhcpconfig` man page for more information about `dhcpconfig` options.

   See "DHCP Data on an Unconfigured Server" on page 159 for more information about removing data.

Configuring and Unconfiguring a Solaris DHCP Client

When you install the Solaris operating environment from CD-ROM, you are prompted to use DHCP to configure network interfaces. If you select yes, the DHCP client software is enabled on your system during Solaris installation. You do not need to do anything else on the Solaris client to use DHCP.

If a client system is already running the Solaris operating environment and not using DHCP, you must unconfigure the system and issue some commands to set up the system to use DHCP when it boots.

If your client is not a Solaris client, consult the client documentation for configuration instructions.

▼ How to Configure a Solaris DHCP Client

This procedure is necessary only if DHCP was not enabled during Solaris installation.

1. **Become superuser on the client system.**
2. **If this system uses preconfiguration instead of interactive configuration, edit the** `sysidcfg` **file to add the** `dhcp` **subkey to the** `network_interface` **keyword.**

   For example, `network_interface=le0 {dhcp}`. See the `sysidcfg`(4) man page for more information.
3. **Unconfigure and shut down the system by typing the following command:**

   ```
   # sys-unconfig
   ```

   See the `sys-unconfig`(1M) man page for more information about what configuration information is removed by this command.
4. **Reboot the system after it has completely shut down.**

   You are prompted for system configuration information by `sysidtool` programs when the system reboots. See the `sysidtool`(1M) man page for more information.
5. **When prompted to use DHCP to configure network interfaces, specify Yes.**

   If you preconfigured the system by using a `sysidcfg` file, insert the `network_interface` keyword, and specify `dhcp` as a dependent keyword. For example, `network_interface=le0 {dhcp}`.

▼ How to Unconfigure a Solaris DHCP Client

1. **Become superuser on the client system.**
2. **If you use a** `sysidcfg` **file to preconfigure the client, remove the** `dhcp` **subkey from the** `network_interface` **keyword.**
3. **Unconfigure and shut down the system by typing the following command:**

   ```
   # sys-unconfig
   ```

   See the `sys-unconfig`(1M) man page for more information about which configuration information is removed by this command.

4. **Reboot the system after it has completely shut down.**

   Because you unconfigured the system, you will be prompted for configuration information by `sysidtool` programs when the system reboots. See the `sysidtool`(1M) man page for more information.

5. **When prompted to use DHCP to configure network interfaces, specify No.**

   If you use `sysidcfg` to specify configuration, you will not be prompted.

CHAPTER 10

Administering DHCP (Task)

This chapter describes tasks you might find useful when you administer the Solaris DHCP service. The chapter includes tasks for the server, BOOTP relay agent, and client. Each task includes a procedure to help you perform the task in DHCP Manager and a procedure for the equivalent task with DHCP command-line utilities. DHCP command-line utilities are more fully documented in man pages.

You should have already completed the initial configuration of your DHCP service and initial network before you use this chapter. Chapter 9 discusses DHCP configuration.

The chapter contains the following information:

DHCP Manager

DHCP Manager is a graphical interface you can use to perform administration tasks on the DHCP service.

The DHCP Manager Window

The DHCP Manager window's appearance differs, depending on whether the server on which it is running was configured as a DHCP server or a BOOTP relay agent.

When the server is configured as a DHCP server, DHCP Manager uses a tab-based window, in which you select a tab for the type of information you want to work with. DHCP Manager features the following tabs:

- **Addresses** – Lists all networks and IP addresses placed under DHCP management. From the Addresses tab, you can add or delete networks and add or delete IP addresses individually or in blocks. You can also modify the properties of individual networks or IP addresses or make the same property modifications for a block of addresses simultaneously. When you start DHCP Manager, it opens on the Addresses tab.
- **Macros** – Lists all macros available in the DHCP configuration database (`dhcptab`) and the options contained within them. From the Macros tab, you can create or delete macros, and modify them by adding options and providing values for the options.
- **Options** – Lists all options that have been defined for this DHCP server. Options listed on this tab are not the standard ones defined in the DHCP protocol. The options are extensions to the standard options, and have a class of Extended, Vendor, or Site. Standard options cannot be changed in any way so they are not listed here.

The following figure shows the DHCP Manager window as it appears when you start it on a DHCP server.

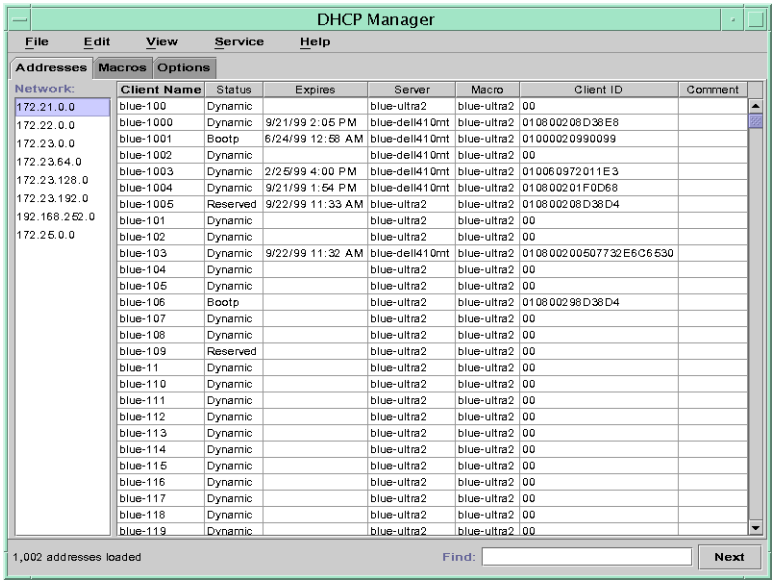


FIGURE 10–1 DHCP Manager on a DHCP Server System

When the server is configured as a BOOTP relay agent, the DHCP Manager window does not show these tabs because the BOOTP relay agent does not need any of this information. You can only modify the BOOTP relay agent's properties and stop/start the DHCP daemon with DHCP Manager. The following figure shows the DHCP Manager window as it appears when you start it on a system configured as a BOOTP relay agent.

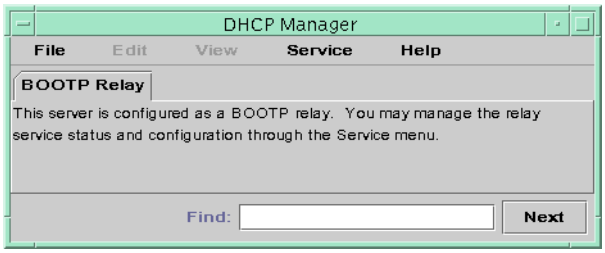


FIGURE 10–2 DHCP Manager on a BOOTP Relay Agent System

DHCP Manager Menus

DHCP Manager menus include:

- **File** – Exit DHCP Manager

- **Edit** – Perform management tasks upon networks, addresses, macros, and options
- **View** – Change the look of the tab currently selected
- **Service** – Manage the DHCP daemon and data store.
- **Help** – Open your web browser and display help for DHCP Manager

When DHCP Manager runs on a BOOTP relay agent, the Edit and View menus are disabled.

All DHCP service management activities are accomplished through the Edit and Service menus. You use the commands in the Edit menu to create, delete, and modify networks, addresses, macros, and options, depending on which tab is selected. When the Addresses tab is selected, the Edit menu also lists wizards, which are sets of dialogs that make it easy to create networks and multiple IP addresses. The Service menu lists commands that enable you to manage the DHCP daemon. You can start/stop, enable/disable, modify the server configuration, and unconfigure the server. The Service menu also lists commands that enable you to convert the data store and export and import data on the server.

Starting and Stopping DHCP Manager

You must run DHCP Manager on a DHCP server system as superuser, but you can display it remotely on another UNIX system using the X Window remote display feature.

▼ How to Start and Stop DHCP Manager

1. **(Optional) Become superuser on the DHCP server system.**
2. **If you are logged in to the DHCP server system remotely, you can display DHCP Manager on your local system as follows.**
   a. **Type the following on the local system:**

      # **xhost +***server-name*

   b. **Type the following on the remote DHCP server system:**

      # DISPLAY=*local-hostname;*export DISPLAY

3. **Type the following command:**

   # **/usr/sadm/admin/bin/dhcpmgr &**

   The DHCP Manager window opens, displaying the Addresses tab if the server is configured as a DHCP server, or no tabs if the server is configured as a BOOTP relay agent.

4. **To stop the DHCP Manager, choose Exit from the File menu.**
   The DHCP Manager window closes.

Setting Up User Access to DHCP Commands

To allow users other than root to execute `dhcpconfig`, `dhtadm`, and `pntadm` commands without first becoming superuser, you must set up role-based access control (RBAC) for those commands. RBAC enables you to more precisely define which users can perform which tasks on the system. See `rbac`(5), `exec_attr`(4), and `user_attr`(4) man pages for more information.

The following procedure explains how to assign a user the DHCP Management profile, which enables the user to execute the DHCP commands.

▼ How to Grant Users Access to DHCP Commands

1. **Become superuser on the DHCP server system.**
2. **Edit the file `/etc/user_attr` to add an entry of the following form for each user you want to be able to manage the DHCP service:**

   username`::::type=normal;profiles=DHCP Management`

   For example, for user `ram`, add the following entry:

   ```
   ram::::type=normal;profiles=DHCP Management
   ```

Starting and Stopping the DHCP Service

The starting and stopping of the DHCP service encompasses several degrees of action you can take to affect the operation of the DHCP daemon. You must understand what it means to start/stop, enable/disable, and configure/unconfigure the DHCP service in order to select the correct procedure to obtain the result you want. The terms are explained below.

- **Start, stop, and restart commands** affect the daemon only at the current session. For example, if you stop the DHCP service, the daemon terminates but restarts

when you reboot the system. DHCP data tables are not affected when you stop the service.

- **Enable and disable commands** affect the daemon for current and future sessions. If you disable the DHCP service, the currently running daemon terminates and does not start when you reboot the server. You must enable the DHCP daemon for the automatic start at system boot to occur. DHCP data tables are not affected. You can disable and enable the DHCP service only from DHCP Manager.
- **Unconfigure command** shuts down the daemon, prevents the daemon from starting on system reboot, and enables you to remove the DHCP data tables. Unconfiguration is described in Chapter 9.

Note – If a server has multiple network interfaces and you do not want to provide DHCP services on all the networks, see "Specifying Network Interfaces to Monitor for DHCP Service" on page 182.

This section provides the procedures to help you start and stop the DHCP service, and enable and disable it.

▼ How to Start and Stop the DHCP Service (DHCP Manager)

1. **Become superuser on the DHCP server system.**
2. **Start DHCP Manager.**
   See "How to Start and Stop DHCP Manager" on page 168 for the procedure.
3. **Select one of the following operations:**
   a. **Choose Start from the Service menu to start the DHCP service.**
   b. **Choose Stop from the Service menu to stop the DHCP service.**
      The DHCP daemon stops until it is manually started again, or the system reboots.
   c. **Choose Restart from the Service menu to stop the DHCP service and immediately restart it.**

▼ How to Start and Stop the DHCP Service (Command Line)

1. **Become superuser on the server system.**
2. **Choose one of the following operations:**
   a. **To start the DHCP service, type the following command:**

      ```
      # /etc/init.d/dhcp start
      ```

      The DHCP daemon starts, using the configuration parameters set in `/etc/inet/dhcpsvc.conf`.

   b. **To stop the DHCP service, type the following command:**

      ```
      # /etc/init.d/dhcp stop
      ```

      The DHCP daemon stops until it is manually started again, or the system reboots.

▼ How to Enable and Disable the DHCP Service (DHCP Manager)

1. **Start DHCP Manager.**
2. **Choose one of the following operations:**
   a. **Choose Enable from the Service menu to start the DHCP service immediately and configure it for automatic startup when the system boots.**
   b. **Choose Disable from the Service menu to stop the DHCP service immediately and prevent it from starting automatically when the system boots.**

Modifying DHCP Service Options (Task Map)

You can change values for some additional features of the DHCP service, some of which were not offered during the initial configuration with DHCP Manager. If you configured your server with `dhcpconfig`, the server is using default values for these options. You can use the Modify Service Options dialog box in DHCP Manager or specify options on the `in.dhcpd` command to change service options.

The following task map shows the tasks related to service options and the procedures to use:

Task	Description	Instructions
Change logging options	Enable or disable verbose logging, enable or disable logging of DHCP transactions, and select a `syslog` facility to use for logging DHCP transactions.	"How to Generate Verbose DHCP Log Messages (DHCP Manager)" on page 174 "How to Generate Verbose DHCP Log Messages (Command Line)" on page 175 "How to Enable and Disable DHCP Transaction Logging (DHCP Manager)" on page 175 "How to Enable and Disable DHCP Transaction Logging for Current Session (Command Line)" on page 176 "How to Log DHCP Transactions to a Separate `syslog` File" on page 176
Change DNS update options	Enable or disable server's adding DNS entries for clients that supply a host name, and determine the maximum time the server should spend attempting to update DNS.	"How to Enable Dynamic DNS Updating for DHCP Clients" on page 178
Enable or disable duplicate IP address detection	Enable or disable the DHCP server's determination that an IP address is not already in use before offering it to a client.	"How to Customize DHCP Server Performance Options (DHCP Manager)" on page 180 "How to Customize DHCP Server Performance Options (Command Line)" on page 181
Change options for DHCP server's reading of configuration information	Enable or disable automatic reading of `dhcptab` at specified intervals, or change the interval between reads.	"How to Customize DHCP Server Performance Options (DHCP Manager)" on page 180 "How to Customize DHCP Server Performance Options (Command Line)" on page 181
Change the number of relay agent hops	Increase or decrease the number of networks a request can travel through before being dropped by the DHCP daemon.	"How to Customize DHCP Server Performance Options (DHCP Manager)" on page 180 "How to Customize DHCP Server Performance Options (Command Line)" on page 181
Change the length of time an IP address offer is cached	Increase or decrease the number of seconds that the DHCP service reserves an offered IP address before offering to a new client.	"How to Customize DHCP Server Performance Options (DHCP Manager)" on page 180 "How to Customize DHCP Server Performance Options (Command Line)" on page 181

The following figure shows DHCP Manager's Modify Service Options dialog box.

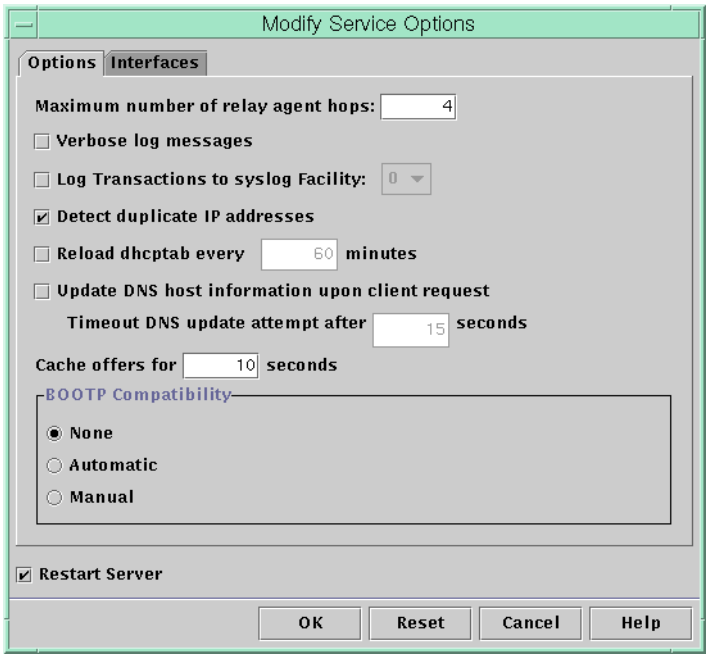


FIGURE 10–3 Modify Service Options Dialog Box

Changing DHCP Logging Options

The DHCP service can log DHCP service messages and DHCP transactions to `syslog`. See the`syslogd`(1M) and`syslog.conf`(4) man pages for more information about `syslog`.

DHCP service messages logged to `syslog` include:

- Error messages, which notify the administrator of conditions that prevent the DHCP service from fulfilling a request by a client or by the administrator.
- Warnings and notices, which notify the administrator of conditions that are abnormal, but do not prevent the DHCP service from fulfilling a request.

You can increase the amount of information reported by using the verbose option for the DHCP daemon. Verbose message output can help you troubleshoot DHCP problems. See "How to Generate Verbose DHCP Log Messages (DHCP Manager)" on page 174.

Another useful troubleshooting technique is transaction logging. Transactions provide information about every interchange between a DHCP server or BOOTP relay and clients. DHCP transactions include:

- ASSIGN – IP address assignment
- ACK – Server acknowledges that client accepts the offered IP address, and sends configuration parameters
- EXTEND – Lease extension
- RELEASE – IP address release
- DECLINE – Client is declining address assignment
- INFORM – Client is requesting network configuration parameters but not an IP address
- NAK – Server does not acknowledge a client's request to use a previously used IP address
- ICMP_ECHO – Server detects potential IP address is already in use by another host.

BOOTP relay transactions include:

- RELAY-CLNT – Message being relayed from the DHCP client to a DHCP server
- RELAY–SRVR – Message being relayed from the DHCP server to the DHCP client

Transaction logging is disabled by default. When enabled, transaction logging uses the `local0` `syslog` facility by default. DHCP transaction messages are generated with a `syslog` severity level of *notice*, so by default, transactions are logged to the file where other notices are logged. However, because they use a local facility, the transaction messages can be logged separately from other notices if you edit the `syslog.conf` file to specify a separate log file.

You can disable or enable transaction logging, and specify a different `syslog` facility, from 0 through 7, as explained in "How to Enable and Disable DHCP Transaction Logging (DHCP Manager)" on page 175. If you edit the server system's `syslog.conf` file, you can also instruct `syslogd` to store the DHCP transaction messages in a separate file, as explained in "How to Log DHCP Transactions to a Separate `syslog` File" on page 176.

▼ How to Generate Verbose DHCP Log Messages (DHCP Manager)

1. **Choose Modify from the Service menu.**
2. **Select Verbose Log Messages.**
3. **Select Restart Server if it is not already selected.**

4. **Click OK.**

   The daemon runs in verbose mode for this session and each subsequent session until you reset this option. Verbose mode can reduce daemon efficiency because of the time taken to display messages.

▼ How to Generate Verbose DHCP Log Messages (Command Line)

1. **Become superuser on the DHCP server system.**
2. **Type the following commands to stop the DHCP daemon and restart it in verbose mode:**

   ```
   # /etc/init.d/dhcp stop
   # /usr/lib/inet/in.dhcpd -v options
   ```

   where *options* are any other options you normally use to start the daemon.

   The daemon runs in verbose mode for this session only.

   Verbose mode can reduce daemon efficiency because of the time taken to display messages.

▼ How to Enable and Disable DHCP Transaction Logging (DHCP Manager)

This procedure enables/disables transaction logging for all subsequent DHCP server sessions.

1. **Choose Modify from the Service menu.**
2. **Select Log Transactions to Syslog Facility.**

   To disable transaction logging, deselect this option.

3. **(Optional) Select a local facility from 0 to 7 to use for logging transactions.**

   By default, DHCP transactions are logged to the location where system notices are logged, which depends on how `syslogd` is configured. If you want the DHCP transactions to be logged to a file separate from other system notices, see "How to Log DHCP Transactions to a Separate `syslog` File" on page 176.

   Message files can quickly become very large when transaction logging is enabled.

4. **Select Restart Server if it is not already selected.**
5. **Click OK.**

The daemon will log transactions to the selected `syslog` facility for this session and each subsequent session until you disable it in this dialog box.

▼ How to Enable and Disable DHCP Transaction Logging for Current Session (Command Line)

1. **Become superuser on the DHCP server system.**
2. **Type the following commands to enable logging for the current session:**

   ```
   # /etc/init.d/dhcp stop
   # /usr/lib/inet/in.dhcpd -l syslog-local-facility
   ```

   where *syslog-local-facility* is a number from 0 through 7. If you omit this option, 0 is used by default. See "How to Enable and Disable DHCP Transaction Logging (DHCP Manager)" on page 175.

Note – To disable transaction logging, omit the `-l` option when starting `in.dhcpd`.

By default, DHCP transactions are logged to the location where system notices are logged, which depends on how `syslogd` is configured. If you want the DHCP transactions to be logged to a file separate from other system notices, see "How to Log DHCP Transactions to a Separate `syslog` File" on page 176.

Message files can quickly become very large when transaction logging is enabled.

▼ How to Log DHCP Transactions to a Separate `syslog` File

1. **Become superuser on the DHCP server system.**
2. **Edit the `/etc/syslog.conf` file on the server system and add a line of the following format:**

   `local`*n*`.notice` *path-to-logfile*

   where *n* is the `syslog` facility number you specified for transaction logging, and *path-to-logfile* is the complete path to the file to use for logging transactions.

   For example, you might add the following line:

   ```
   local0.notice /var/log/dhcpsrvc
   ```

   See the `syslog.conf`(4) man page for more information about the `syslog.conf` file.

Enabling Dynamic DNS Updates by DHCP Server

If a host name is mapped to the IP address leased to a DHCP client and the DHCP server has been configured to supply host names, the DHCP server will inform the client of the name it has been assigned. Alternatively, the DHCP server may be configured so that DHCP clients may supply their own host names and the DHCP server will attempt DNS updates on their behalf.

DNS provides basic name-to-address and address-to-name services for the Internet. Once a DNS update is made, other systems may refer to the DHCP client system by name.

You can enable the DHCP service to update the DNS service with the host names of DHCP clients that supply their own host names. When a system's name is registered with DNS, the system is visible outside its domain. In order for the DNS update feature to work, the DNS server, DHCP server, and DHCP client must all be set up correctly, and the requested name must not be in use by another system in the domain.

The DHCP server's DNS update feature works if all the following are true:

- DNS server supports RFC 2136.
- DNS software that is BIND-based, whether on the DHCP server system or the DNS server system, must be v8.2.2, patch level 5 or newer.
- DNS server is configured to accept dynamic DNS updates from the DHCP server.
- DHCP server is configured to make dynamic DNS updates.
- DNS support is configured for the DHCP client's network on the DHCP server.
- DHCP client is configured to supply a requested host name in its DHCP request message.
- Requested host name corresponds to a DHCP-owned address or has no corresponding address.

▼ How to Enable Dynamic DNS Updating for DHCP Clients

Note – Be aware that dynamic DNS updates are by nature a *security risk*.

By default, the Solaris DNS daemon (`in.named`) does not allow dynamic updates. Authorization for dynamic DNS updates is granted if the requesting host's IP address is assigned to the `allow-update` keyword in the appropriate zones of the `named.conf` configuration file on the DNS server system. No other security is provided. You must carefully weigh the convenience of this facility for users against the security risk created when you enable dynamic DNS updates.

---

1. **At the DNS server, edit the `/etc/named.conf` file as root.**
2. **Find the `zone` section for the appropriate domain and add the DHCP server's IP addresses to the `allow-update` keyword.**

   For example, if the DHCP server resides at addresses 10.0.0.1 and 10.0.0.2, a `named.conf` file for the `dhcp.domain.com` zone would be modified as follows:

```
zone "dhcp.domain.com" in {
             type master;
             file "db.dhcp";
             allow-update { 10.0.0.1; 10.0.0.2; };
};

zone "10.IN-ADDR.ARPA" in {
             type master;
             file "db.10";
             allow-update { 10.0.0.1; 10.0.0.2; };
};
```

   Note that `allow-update` for both zones must be enabled to allow the DHCP server to update both A and PTR records on the DNS server.
3. **On the DHCP server, start DHCP Manager.**
4. **Choose Modify from the Service menu.**

   The Modify Service Options dialog box opens.
5. **Select Update DNS Host Information Upon Client Request.**
6. **Specify the number of seconds to wait for a response from the DNS server before timing out, then click OK.**

   The default value should be adequate. If you have timeout problems, you can increase the value later.
7. **Click the Macros tab and ensure that the correct DNS domain is specified.**

The `DNSdmain` option must be passed with the correct domain name to any client that expects dynamic DNS update support. By default, `DNSdmain` is specified in the server macro, which is used as the configuration macro bound to each IP address.

8. **Set up the DHCP client to specify its host name when requesting DHCP service.**

   If you use the Solaris DHCP client, see "How to Enable a Solaris Client to Request Specific Host Name" on page 179. If your client is not a Solaris DHCP client, see the documentation for your DHCP client for information about how to do this.

▼ How to Enable a Solaris Client to Request Specific Host Name

1. **On the client system, edit the `/etc/default/dhcpagent` file as root.**

2. **Find the keyword `REQUEST_HOSTNAME` in the `/etc/default/dhcpagent` file and modify it as follows:**

   ```
   REQUEST_HOSTNAME=yes
   ```

   If there is a comment sign (#) in front of the keyword, remove the #. If the keyword is not present, insert it.

3. **Edit the `/etc/hostname.`*interface* file on the client system and add the following line:**

   `inet` *hostname*

   where *hostname* is the name you want the client to use.

4. **As root, type the following commands to have the client perform a full DHCP negotiation upon rebooting:**

   ```
   # pkill dhcpagent
   # rm /etc/dhcp/interface.dhc
   # reboot
   ```

   The DHCP server makes sure that the host name is not in use by another system on the network before the server assigns it to the client. Depending how it is configured, the DHCP server may update name services with the client's host name.

Customizing DHCP Service Performance Options

You can change options that affect the performance of the DHCP service. These options are described in the following table.

TABLE 10–1 Options Affecting DHCP Server Performance

Server Option	Description	Key in `/etc/inet/dhcpsvc.conf`
Number of BOOTP relay agent hops	If a request has traveled through more than a given number of BOOTP relay agents, it is dropped. The default maximum number of relay agent hops is 4, and it is not likely that this number will be surpassed unless your network is set up to pass requests through several BOOTP relay agents before they reach a DHCP server.	`RELAY_HOPS=`*integer*
Verification of IP address availability before making an offer	By default, the server pings an IP address before offering it to a client to verify that it is not already in use. You can disable this feature to decrease the time it takes to make an offer, but this creates the risk of having duplicate IP addresses in use.	`ICMP_VERIFY=TRUE/FALSE`
Automatic reading of `dhcptab` at specified intervals	The server can be set to automatically read the `dhcptab` at the interval in minutes you specify. If your network configuration information does not change frequently, and you do not have multiple DHCP servers, it is not necessary to reload `dhcptab` automatically. Also note that DHCP Manager gives you the option to have the server reload `dhcptab` after you make a change to the data.	`RESCAN_INTERVAL=`*min*
Length of time to reserve an IP address that has been offered	After a server offers an IP address to a client, it caches the offer, during which time the server does not offer the address again. You can change the number of seconds for which the offer is cached. The default is 10 seconds. On slow networks, you made need to increase the offer time.	`OFFER_CACHE_TIMEOUT=`*sec*

The following procedures describe how to change these options.

▼ How to Customize DHCP Server Performance Options (DHCP Manager)

1. **Choose Modify from the Service menu.**
2. **To change the number of BOOTP relay agents a request can pass through, specify a different Maximum Number of Relay Agent Hops.**
3. **To have the DHCP server verify that an IP address is not in use before it offers the address to a client, select Detect Duplicate IP Addresses.**
4. **To have the DHCP server read `dhcptab` at specified intervals, select Reload dhcptab**

Every *n* Minutes, and type the number of minutes for the interval.

5. **To change the length of time the server holds an IP address open after it makes an offer, type the number of seconds in the field Cache Offers for *n* Seconds.**
6. **Select Restart Server if it is not already selected.**
7. **Click OK.**

▼ How to Customize DHCP Server Performance Options (Command Line)

If you change options with this procedure, the changed options affect only the current server session. If the DHCP server system reboots, the DHCP server starts with the settings specified during server configuration. If you want settings to apply to all future sessions, you must make changes using DHCP Manager.

1. **Become superuser on the DHCP server system.**
2. **Type the following command:**

```
# /etc/init.d/dhcp stop
# /usr/lib/inet/in.dhcpd options
```

where *options* are any of the following:

`-h` *relay-hops*	Specifies the maximum number of relay agent hops that can occur before the daemon drops the DHCP/BOOTP datagram.
`-n`	Disables automatic duplicate IP address detection. This is not recommended.
`-t` *dhcptab_rescan_interval*	Specifies the interval in minutes that the DHCP server should use to schedule the automatic rereading of the `dhcptab` information.
`-o` *seconds*	Specifies the number of seconds the DHCP server should cache the offers it has extended to discovering DHCP clients. The default setting is 10 seconds.

For example, the following command sets the hop count to 2, disables duplicate IP address detection, sets the rescan interval to 30 minutes, and sets the offer time to 20 seconds.

```
# /usr/lib/inet/in.dhcp -h 2 -n -t 30 -o 20
```

Adding, Modifying, and Removing DHCP Networks (Task Map)

When you configure a DHCP server, you must also configure at least one network in order to use the DHCP service. You can add more networks at any time.

The following task map lists tasks you may need to perform when working with DHCP networks and the procedures used to carry them out.

Task	Description	Instructions
Enable or disable DHCP service on server network interfaces	The default behavior is to monitor all network interfaces for DHCP requests, but you can change this.	"How to Specify Network Interfaces for DHCP Monitoring (DHCP Manager)" on page 183
Add a new network to the DHCP service	Place a network under DHCP management, for the purpose of managing IP addresses on the network.	"How to Add a DHCP Network (DHCP Manager)" on page 185
Change parameters of a DHCP-managed network	Modify the information that is passed to clients of a particular network.	"How to Modify Configuration of a DHCP Network (DHCP Manager)" on page 186 "How to Modify Configuration of a DHCP Network (`dhtadm`)" on page 187
Delete a network from the DHCP service	Remove a network so that IP addresses on the network are no longer managed by DHCP.	"How to Remove a DHCP Network (DHCP Manager)" on page 188 "How to Remove a DHCP Network (`pntadm`)" on page 189

Specifying Network Interfaces to Monitor for DHCP Service

By default, both `dhcpconfig` and DHCP Manager's Configuration Wizard configure the DHCP server to monitor all the server system's network interfaces. If you add a new network interface to the server system, the DHCP server automatically monitors the new interface when you boot the system. You can then add any networks that will be monitored through the network interface.

However, DHCP Manager also allows you to specify which network interfaces the DHCP service should monitor and which it should ignore. You might want to ignore an interface if you do not want to offer DHCP service on that network.

If you specify that any interface should be ignored, and then install a new interface, the DHCP server ignores the new interface unless you add it to the server's list of monitored interfaces. You can specify interfaces with DHCP Manager.

This section includes a procedure that enables you to specify which network interfaces DHCP should monitor, and which to ignore. The procedure uses the Interfaces tab of the DHCP Manager's Modify Service Options dialog box, which is shown in the following figure.

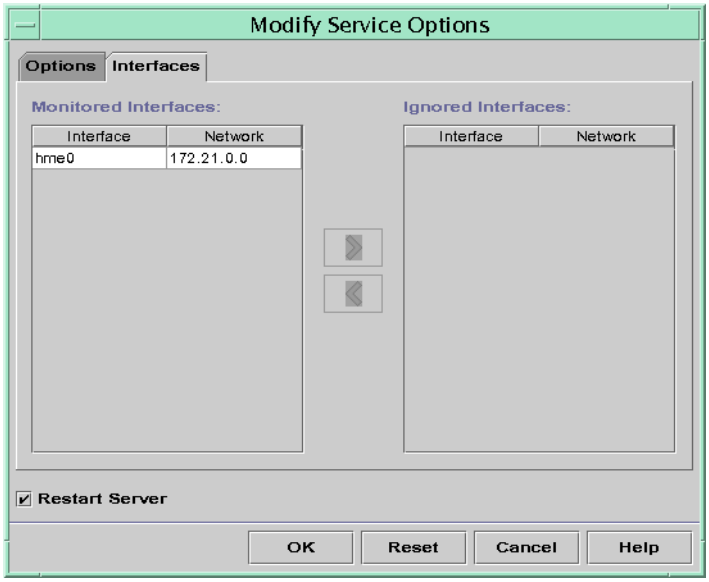


FIGURE 10–4 Interfaces Tab of Modify Service Options Dialog Box

▼ How to Specify Network Interfaces for DHCP Monitoring (DHCP Manager)

1. **Choose Modify from the Service menu.**
   The Modify Service Options dialog box is displayed.
2. **Select the Interfaces tab.**
3. **Select the appropriate network interface and click the arrow buttons to move the interface to the Monitored Interfaces list or the Ignored Interfaces list.**

For example, to ignore an interface, select it in the Monitored Interfaces list and click the right arrow button to move the interface in the Ignored Interfaces list.

4. **Make sure Restart Server is selected and click OK.**

Adding DHCP Networks

When you use DHCP Manager to configure the server, the first network (usually the local one on the server system's primary interface) is also configured at the same time. If you want to configure additional networks, use the DHCP Network Wizard in DHCP Manager.

If you use `dhcpconfig -D` to configure the server, you must manually configure all networks that will be served by the DHCP service. See "How to Add a DHCP Network (`dhcpconfig`)" on page 185 for more information.

The following figure shows the initial dialog box for the DHCP Network Wizard in DHCP Manager.

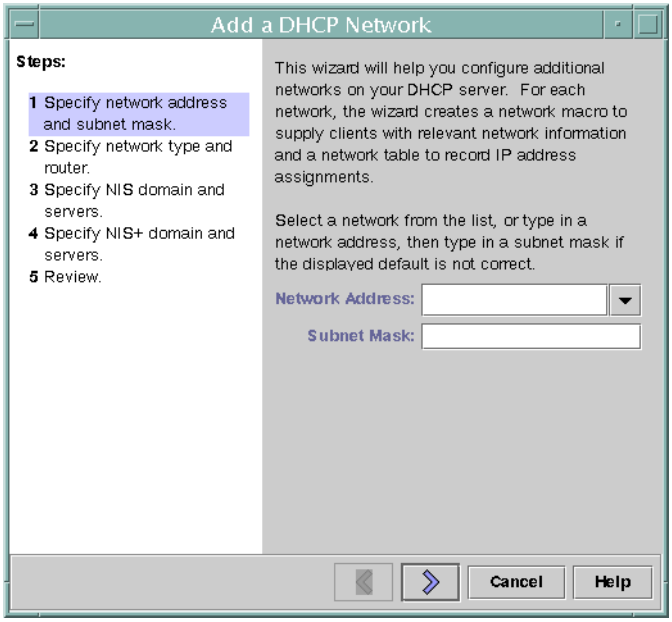


FIGURE 10–5 DHCP Manager's Network Wizard

When you configure a new network, DHCP Manager creates the following components:

- Network table in the data store. The new network is shown in the network list on the Addresses tab of DHCP Manager.
- Network macro that contains information needed by clients that reside on this network. The network macro's name matches the IP address of the network. The network macro is added to the `dhcptab` in the data store.

▼ How to Add a DHCP Network (DHCP Manager)

1. **Click the Addresses tab in DHCP Manager.**

   Any networks already configured for DHCP service are listed.

2. **Choose Network Wizard from the Edit menu.**

3. **Select options or type requested information based on the decisions you made during the planning phase.**

   Planning is described in "Planning for Remote Network Configuration" on page 149.

   If you have difficulty with the wizard, click Help in the wizard window to open your web browser and display help for the DHCP Network Wizard.

4. **Click Finish to complete the network configuration when you have finished entering the requested information.**

   The Network Wizard creates a network macro whose name matches the IP address of the network. If you click the Macros tab in the DHCP Manager window and select the network macro, you can confirm that the information you provided in the wizard has been inserted as values for options contained in the macro.

   The Network Wizard creates an empty network table, which is listed in the left pane of the window. You must add addresses for the network before the network's IP addresses can be managed under DHCP. See "Adding Addresses to the DHCP Service" on page 196 for more information.

▼ How to Add a DHCP Network (`dhcpconfig`)

1. **Become superuser or a user assigned to the DHCP Management profile .**

2. **Type the following command on the DHCP server system:**

   # **`/usr/sbin/dhcpconfig -N`** *network_address*

   where *network_address* is the IP address of the network you want to add to the DHCP service. See the `dhcpconfig` man page for suboptions you can use with the `-N` option.

   If you do not use suboptions, `dhcpconfig` uses network files to obtain information it needs about the network.

3. **Add IP addresses for the network so clients on the network can obtain addresses.**
   See "Adding Addresses to the DHCP Service" on page 196.

Modifying DHCP Network Configuration

After you add a network to the DHCP service, you can modify the configuration information you originally supplied by modifying the network macro used to pass information to the clients on the network.

The following figure shows the Macros tab of the DHCP Manager.

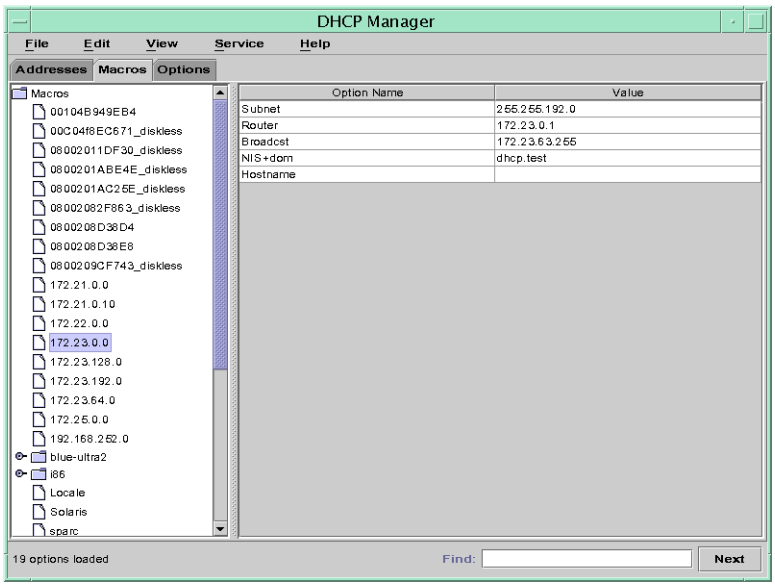


FIGURE 10–6 DHCP Manager's Macros Tab

▼ How to Modify Configuration of a DHCP Network (DHCP Manager)

1. **Select the Macros tab.**
   All macros defined for this DHCP server are listed in the left pane.
2. **Select the network macro whose name matches the network whose configuration you want to change.**
   The network macro name is the network IP address.

3. **Choose Properties from the Edit menu.**

   The Macro Properties dialog box displays a table of the options included in the macro.

4. **Select the option you want to modify.**

   The option name and value are displayed in text fields near the top of the dialog box.

5. **Type the new value for the option and click Modify.**

   You can also add options here by clicking Select in the dialog box. See "Modifying DHCP Macros" on page 210 for more general information about modifying macros.

6. **Select Notify DHCP Server of Change and click OK.**

   The change is made to the `dhcptab` and the DHCP server is signaled to reread the `dhcptab` and put the changes into effect.

▼ How to Modify Configuration of a DHCP Network (`dhtadm`)

1. **Determine which macro includes information for all clients of the network.**

   The network macro's name matches the network IP address.

   If you don't know which macro includes this information, you can display the `dhcptab` database to list all macros by using the command `dhtadm -P`.

2. **Type a command of the following format to change the value of the option you want to change:**

   ```
   # dhtadm -M -m macro-name -e 'symbol=value'
   ```

   For example, to change the `10.25.62.0` macro's lease time to 57600 seconds and NIS domain to `sem.example.com`, type the following commands:

   ```
   # dhtadm -M -m 10.25.62.0 -e 'LeaseTim=57600'
   # dhtadm -M -m 10.25.62.0 -e 'NISdmain=sem.example.com'
   ```

3. **Type the following command as root to make the DHCP daemon reread `dhcptab`:**

   ```
   # pkill -HUP in.dhcpd
   ```

Removing DHCP Networks

DHCP Manager enables you to remove multiple networks at once. You have the option to automatically remove the hosts table entries associated with the

DHCP-managed IP addresses on those networks as well. The following figure shows DHCP Manager's Delete Networks dialog box.

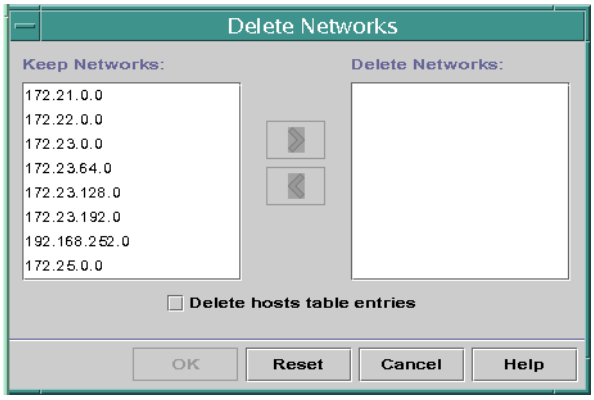


FIGURE 10–7 Delete Networks Dialog Box

The `pntadm` command requires you to delete each IP address entry from a network before you delete that network. You can delete only one network at a time.

▼ How to Remove a DHCP Network (DHCP Manager)

1. **Select the Addresses tab.**
2. **Choose Delete Networks from the Edit menu.**
   The Delete Networks dialog box opens.
3. **In the Keep Networks list, select the networks you want to delete.**
   Press the Control key while you click with the mouse to select multiple networks, or press the Shift key while you click to select a range of networks.
4. **Click the right arrow button to move the selected networks to the Delete Networks list.**
5. **If you want to remove the host table entries for the DHCP–managed addresses on this network, select Delete Host Table Entries.**
   Note that this does not delete the host registrations at the DNS server for these addresses. It affects only the local name service.
6. **Click OK.**

▼ How to Remove a DHCP Network (pntadm)

Note that this procedure deletes the addresses on the network before removing the network. This ensures that the host names are removed from the `hosts` file or database.

1. **On the server system, become superuser or a user assigned to the DHCP Management profile .**
2. **Type a command following this format to remove an IP address and its host name from the name service:**

   `# pntadm -D -y` *IP-address*

   For example, to remove address 10.25.52.1, type the following command:

   `# pntadm -D -y 10.25.52.1`

   The `-y` option specifies to delete the host name.
3. **Repeat the `pntadm -D -y` command for each address in the network.**

   You might want to create a script to do this if you are deleting many addresses.
4. **After all addresses are deleted, type the following command to delete the network from the DHCP service.**

   `# pntadm -R` *network-IP-address*

   For example, to remove network 10.25.52.0, type the following command:

   `# pntadm -R 10.25.52.0`

   See the `pntadm` man page for more information about using `pntadm`.

Supporting BOOTP Clients with DHCP Service (Task Map)

To support BOOTP clients on your DHCP server, you must set up your DHCP server to be BOOTP compatible. You can register BOOTP clients in the DHCP server's network table or reserve a number of IP addresses for allocation to BOOTP clients, depending how you set up BOOTP compatibility.

Note – BOOTP addresses are permanently assigned, whether or not you explicitly assign them a permanent lease.

The following task map lists tasks you may need to perform to support BOOTP clients and the procedures used to carry them out.

Task	Description	Instructions
Set up automatic BOOTP support	Provide IP address for any BOOTP client on a DHCP-managed network, or on a network connected by a relay agent to a DHCP-managed network. This requires you to reserve a pool of addresses for exclusive use by BOOTP clients. This option may be more useful if the server must support a large number of BOOTP clients.	"How to Set Up Support of Any BOOTP Client (DHCP Manager)" on page 190
Set up manual BOOTP support	Provide IP address for only those BOOTP clients that have been manually registered with the DHCP service. This requires you to bind a client's ID to a particular IP address that has been marked for BOOTP clients. This option is useful for a small number of BOOTP clients, or in the event that you want to restrict the BOOTP clients that can use the server.	"How to Set Up Support of Registered BOOTP Clients (DHCP Manager)" on page 191

▼ How to Set Up Support of Any BOOTP Client (DHCP Manager)

1. **Select Modify from the Service menu.**
   The Modify Service Options dialog box opens.
2. **In the BOOTP Compatibility section of the dialog box, select Automatic.**
3. **Select Restart Server, if it is not already selected.**
4. **Click OK.**

5. **Select the Addresses tab in DHCP Manager.**
6. **Select addresses that you want to reserve for BOOTP clients.**
   Select a range of addresses by clicking the first address, pressing the Shift key, and clicking the last address.
   Select multiple non-concurrent addresses by pressing the Control key while clicking each address.
7. **Select Properties from the Edit menu.**
   The Modify Multiple Addresses dialog box opens.
8. **In the BOOTP section, select Assign All Addresses Only to BOOTP Clients.**
   All other options should be set to Keep Current Settings.
9. **Click OK.**
   Any BOOTP client can now obtain an address from this DHCP server.

▼ How to Set Up Support of Registered BOOTP Clients (DHCP Manager)

1. **Select Modify from the Service menu.**
   The Modify Service Options dialog box opens.
2. **In the BOOTP Compatibility section of the dialog box, select Manual.**
3. **Select Restart Server if it is not already selected.**
4. **Click OK.**
5. **Select the Addresses tab in DHCP Manager.**
6. **Select an address you want to assign to a particular BOOTP client.**
7. **Choose Properties from the Edit menu.**
   The Address Properties dialog box opens.
8. **Select the Lease tab.**
9. **In the Client ID field, type the client's identifier.**
   For a BOOTP client that runs the Solaris operating environment on an Ethernet network, the client ID is a string derived from the client's hexadecimal Ethernet address, preceded by the Address Resolution Protocol (ARP) type for Ethernet (01). For example, a BOOTP client having the Ethernet address 8:0:20:94:12:1e would use the client ID 0108002094121E.

Tip – As superuser on a Solaris client system, type the following command to obtain the Ethernet address for the interface:

```
ifconfig -a
```

10. **Select Reserved to reserve the IP address for this client.**
11. **Select Assign Only to BOOTP Clients.**
12. **Click OK.**
    In the Addresses tab, BOOTP is displayed in the Status field, and the client ID you entered is listed in the Client ID field.

Working With IP Addresses in the DHCP Service (Task Map)

You can use DHCP Manager or the `pntadm` command to add IP addresses, modify their properties, and remove them from the DHCP service. Before you work with IP addresses, you should refer to Table 10–2 to become familiar with IP address properties. The table provides information for users of DHCP Manager and `pntadm`.

Note – This section does not include procedures for using the `pntadm` command. However Table 10–2 includes examples of using `pntadm` to specify IP address properties while adding and modifying IP addresses. Also refer to the `pntadm` man page for more information about `pntadm`.

The following task map lists tasks you must perform to add, modify, remove IP addresses and the procedures used to carry them out.

Task	Description	Instructions
Add single or multiple IP addresses to DHCP service.	Add IP addresses on networks that are already managed by the DHCP service by using DHCP Manager.	"How to Add a Single IP Address (DHCP Manager)" on page 198 "How to Duplicate an Existing IP Address (DHCP Manager)" on page 198 "How to Add Multiple Addresses (DHCP Manager)" on page 199 "How to Add Addresses (`pntadm`)" on page 199
Change properties of an IP address.	Change any of the IP address properties described in Table 10–2.	"How to Modify IP Address Properties (DHCP Manager)" on page 201 "How to Modify IP Address Properties (`pntadm`)" on page 202
Remove IP addresses from DHCP service.	Prevent the use of specified IP addresses by DHCP.	"How to Mark Addresses Unusable (DHCP Manager)" on page 202 "How to Mark Addresses Unusable (`pntadm`)" on page 203 "How to Delete IP Addresses from DHCP Service (DHCP Manager)" on page 204 "How to Delete IP Addresses from DHCP Service (`pntadm`)" on page 205
Assign consistent address to a DHCP client.	Set up a client to receive the same IP address each time it requests its configuration.	"How to Assign a Consistent IP Address to a DHCP Client (DHCP Manager)" on page 206 "How to Assign a Consistent IP Address to a DHCP Client (`pntadm`)" on page 207

The following table lists and describes the properties of IP addresses.

TABLE 10–2 IP Address Properties

Property	Description	How to Specify in `pntadm` Command
Network address	Address of the network that contains the IP address you are working with. The network address is displayed in the Networks list on the Addresses tab in DHCP Manager.	The network address must be the last argument on the `pntadm` command line used to create, modify, or delete an IP address. For example, to add an IP address to network 10.21.0.0 **`pntadm -A`** *ip-address options* **`10.21.0.0`**
IP address	Address you are working with, whether you are creating, modifying, or deleting it. The IP address is displayed in the first column of the DHCP Manager's Addresses tab.	The IP address must accompany the `-A`, `-M`, and `-D` options to the `pntadm` command. For example, to modify IP address 10.21.5.12 **`pntadm -M 10.21.5.12`** *options* **`10.21.0.0`**
Client name	Host name mapped to the IP address in the hosts table. This name may be automatically generated by DHCP Manager when addresses are created. If you create a single address, you can supply the name.	Specify the client name with the `-h` option. For example, to specify client name carrot12 for 10.21.5.12: **`pntadm -M 10.21.5.12 -h carrot12 10.21.0.0`**
Owning server	DHCP server that manages the IP address and is responsible for responding to the DHCP client's request for IP address allocation.	Specify the owning server name with the `-s` option. For example to specify server blue2 to own 10.21.5.12: **`pntadm -M 10.21.5.12 -s blue2 10.21.0.0`**
Configuration macro	Macro the DHCP server uses to obtain network configuration options from the `dhcptab`. Several macros are created automatically when you configure a server and add networks. See "About Macros" on page 127 for more information about macros. When DHCP Manager creates addresses, it creates a server macro and assigns that macro as the configuration macro for each address.	Specify the macro name with the `-m` option. For example, to assign the server macro blue2 to address 10.21.5.12 **`pntadm -M 10.21.5.12 -m blue2 10.21.0.0`**

TABLE 10–2 IP Address Properties *(Continued)*

Property	Description	How to Specify in `pntadm` Command
Client ID	Text string that is unique within the DHCP service. If the client ID is listed as 00, the address is not allocated to any client. If you specify a client ID when modifying the properties of an IP address, you manually bind the address to that client for its exclusive use. The client ID is determined by the vendor of the DHCP client. If your client is not a Solaris DHCP client, consult your DHCP client documentation for more information. For Solaris DHCP clients, the client ID is derived from the client's hexadecimal hardware address, preceded by the ARP code for the type of network, such as 01 for Ethernet. The ARP codes are assigned by the Internet Assigned Numbers Authority (IANA) in the ARP Parameters section of the Assigned Numbers standard at http://www.iana.com/numbers.html For example, a Solaris client with the hexadecimal Ethernet address 8:0:20:94:12:1e would use the client ID 0108002094121E. The client ID is listed in DHCP Manager and `pntadm` when a client is currently using an address. **Tip:** As superuser on the Solaris client system, type the following command to obtain the Ethernet address for the interface: `ifconfig -a`	Specify the client ID with the `-i` option. For example, to assign client ID 08002094121E to address 10.21.5.12 **`pntadm -M 10.21.5.12 -i 0108002094121E 10.21.0.0`**

TABLE 10–2 IP Address Properties *(Continued)*

Property	Description	How to Specify in `pntadm` Command
Reserved	The setting that specifies the address is reserved exclusively for the client indicated by the client ID, and the DHCP server cannot reclaim the address. If you choose this option, you manually assign the address to the client.	Specify that the address is reserved, or manual, with the `-f` option. For example, to specify that IP address 10.21.5.12 is reserved for a client: **`pntadm -M 10.21.5.12 -f MANUAL 10.21.0.0`**
Lease type/policy	The setting that determines how DHCP manages the use of the IP address by clients. A lease may be dynamic or permanent. See "Dynamic and Permanent Lease Type" on page 147 for a complete explanation.	Specify that the address would be permanently assigned with the `-f` option. Addresses are dynamically leased by default. For example, to specify that IP address 10.21.5.12 has a permanent lease: **`pntadm -M 10.21.5.12 -f PERMANENT 10.21.0.0`**
Lease expiration time	Date and time when the lease expires, applicable only when a dynamic lease is specified. The date is specified in *mm/dd/yyyy* format.	Specify an absolute lease expiration time with `-e`. For example, to specify an expiration time of January 1, 2002: **`pntadm -M 10.21.5.12 -e 01/01/2002 10.21.0.0`**
BOOTP setting	The setting that marks the address as reserved for BOOTP clients. See "Supporting BOOTP Clients with DHCP Service (Task Map)" on page 189 for more information about supporting BOOTP clients.	Reserve an address for BOOTP clients with `-f`. For example, to reserve IP address 10.21.5.12 for BOOTP clients: **`pntadm -M 10.21.5.12 -f BOOTP 10.21.0.0`**
Unusable setting	The setting that marks the address so it cannot be assigned to any client.	Mark an address unusable with `-f`. For example, to mark IP address 10.21.5.12 unusable: **`pntadm -M 10.21.5.12 -f UNUSABLE 10.21.0.0`**

Adding Addresses to the DHCP Service

Before you add addresses, you must add the network that owns them to the DHCP service. See "Adding DHCP Networks" on page 184 for information about adding networks.

You can add addresses with DHCP Manager or `pntadm`.

On networks that are already managed by the DHCP service, you can add addresses in several ways with DHCP Manager:

- **Add a single IP address** – Place one new IP address under DHCP management.
- **Duplicate an existing IP address** – Copy the properties of an existing IP address managed by DHCP, and supply a new IP address and client name.
- **Add a range of multiple IP addresses** – Use the Address Wizard to place a series of IP addresses under DHCP management.

The following figure shows the Create Address dialog box. The Duplicate Address dialog box is identical to the Create Address dialog box, except that the text fields display the values for an existing address.

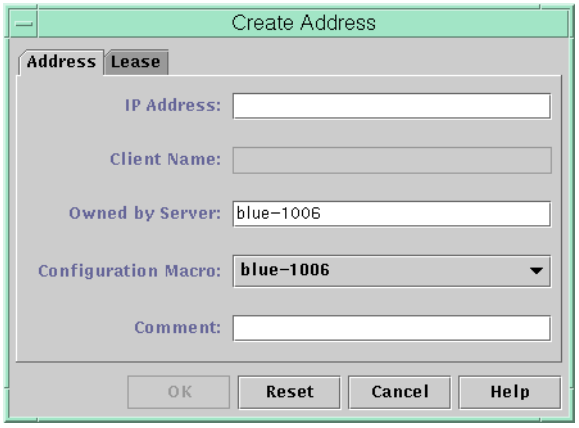


FIGURE 10–8 Create Address Dialog Box

The following figure shows the first dialog of the Address Wizard, used to add a range of IP addresses.

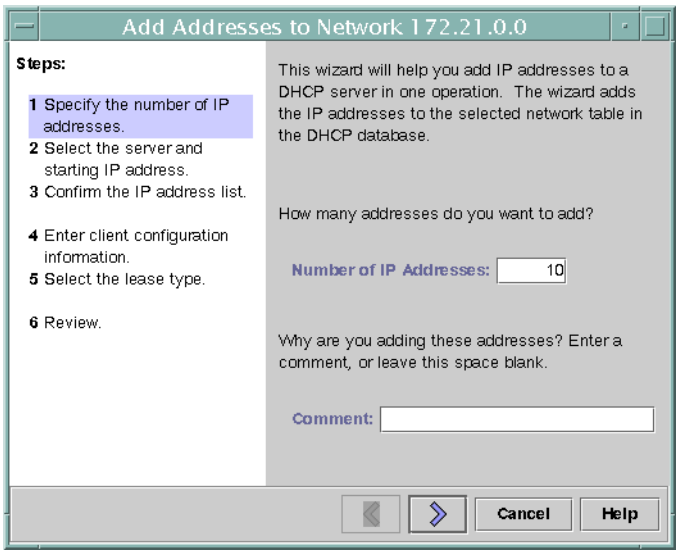


FIGURE 10–9 Address Wizard

▼ How to Add a Single IP Address (DHCP Manager)

1. **Select the Addresses tab.**
2. **Select the network where the new IP address is to be added.**
3. **Choose Create from the Edit menu.**
   The Create Address dialog box opens.
4. **Select or type values for the address settings on the Address and Lease tabs.**
   See Table 10–2 for information about the settings.
5. **Click OK.**

▼ How to Duplicate an Existing IP Address (DHCP Manager)

1. **Select the Addresses tab.**
2. **Select the network where the new IP address is located.**
3. **Select the address whose properties you want to duplicate.**

4. **Choose Duplicate from the Edit menu.**
5. **Change the IP address and client name for the address.**
   Most other options should remain the same, but you can change them if necessary.
6. **Click OK.**

▼ How to Add Multiple Addresses (DHCP Manager)

1. **Select the Addresses tab.**
2. **Select the network where the new IP addresses are to be added.**
3. **Choose Address Wizard from the Edit menu.**
   The Address Wizard prompts you to provide values for the IP address properties. See Table 10–2 for more information about the properties. "Making Decisions for IP Address Management (Task Map)" on page 145 includes more extensive information.
4. **Click the right arrow button as you finish entering information in each screen, and click Finish on the last screen.**
   The Addresses tab is updated with the new addresses.

▼ How to Add Addresses (`pntadm`)

1. **Become superuser or a user assigned to the DHCP Management profile .**
2. **Type a command of the following format:**

   # **`pntadm -A`** *ip-address options network-address*

   Refer to the `pntadm` man page for a list of options you can use with `pntadm -A`. In addition, Table 10–2 shows some sample `pntadm` commands that specify options.

Note – You can write a script to add multiple addresses with `pntadm`. See Example 12–1 for an example.

Modifying IP Addresses in the DHCP Service

After you add IP addresses to the DHCP service, you can modify any of the properties described in Table 10–2 by using DHCP Manager or the `pntadm -M` command. See the `pntadm` man page for more information about `pntadm -M`.

The following figure shows the Address Properties dialog box that you use to modify IP address properties.

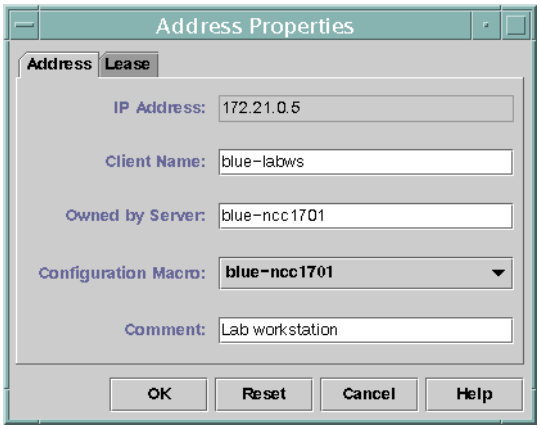


FIGURE 10–10 Address Properties Dialog Box

The following figure shows the Modify Multiple Addresses dialog box that you use to modify multiple IP addresses.

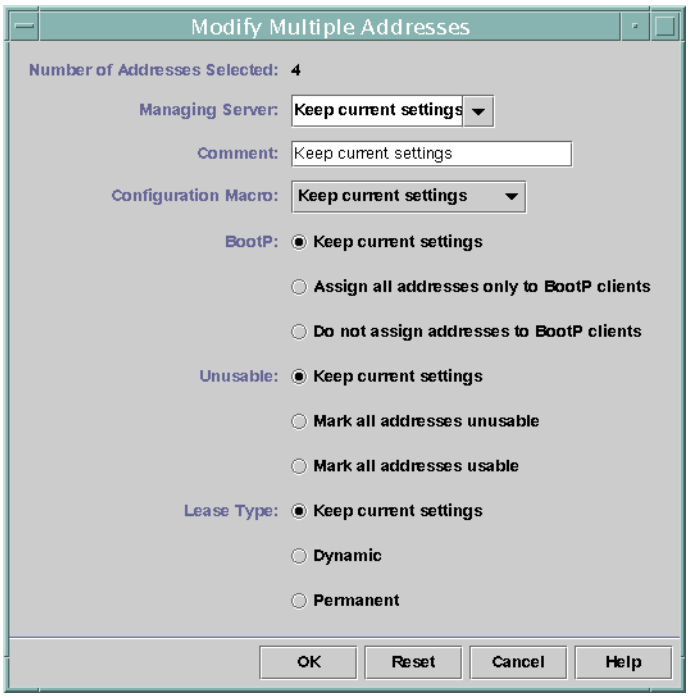


FIGURE 10–11 Modify Multiple Addresses Dialog Box

▼ How to Modify IP Address Properties (DHCP Manager)

1. **Select the Addresses tab.**
2. **Select the IP address's network.**
3. **Select one or more IP addresses you want to modify.**

   If you want to modify more than one address, press the Control key while you click with the mouse to select multiple addresses. You can also press the Shift key while you click to select a block of addresses.
4. **Choose Properties from the Edit menu.**

   The Modify Addresses dialog box or the Modify Multiple Address dialog box opens.
5. **Change the appropriate properties.**

   Click the Help button or refer to Table 10–2 for information about the properties.
6. **Click OK.**

▼ How to Modify IP Address Properties (pntadm)

1. **Become superuser or a user assigned to the DHCP Management profile .**
2. **Enter a command of the following format:**

   # **pntadm -M** *ip-address options network-address*

   Many options can be used with the pntadm command, which are documented in the pntadm man page.

   Table 10–2 shows some sample pntadm commands that specify options.

Removing Addresses From DHCP Service

At times you might want the DHCP service to stop managing a particular address or group of addresses. The method you use to remove an address from DHCP depends on whether you want the change to be temporary or permanent.

- To temporarily prevent the use of addresses, you can mark them unusable in the Address Properties dialog box as described in "Marking IP Addresses Unusable by the DHCP Service" on page 202.
- To permanently prevent the use of addresses by DHCP clients, delete the addresses from the DHCP network tables, as described in "Deleting IP Addresses from DHCP Service" on page 203.

Marking IP Addresses Unusable by the DHCP Service

You can use the pntadm -M command with the -f UNUSABLE option to mark addresses unusable.

In DHCP Manager, you use the Address Properties dialog box, shown in Figure 10–10, to mark individual addresses, and the Modify Multiple Addresses dialog box, show in Figure 10–11, to mark multiple addresses, as described in the following procedure.

▼ How to Mark Addresses Unusable (DHCP Manager)

1. **Select the Addresses tab.**
2. **Select the IP address's network.**

3. **Select one or more IP addresses you want to mark unusable.**

   If you want to mark more than one address unusable, press the Control key while you click with the mouse to select multiple addresses. You can also press the Shift key while you click to select a block of addresses.

4. **Choose Properties from the Edit menu.**

   The Modify Addresses dialog box or the Modify Multiple Address dialog box opens.

5. **If you are modifying one address, select the Lease tab.**

6. **Select Address is Unusable.**

   If you are editing multiple addresses, select Mark All Addresses Unusable.

7. **Click OK.**

▼ How to Mark Addresses Unusable (`pntadm`)

1. **Become superuser or a user assigned to the DHCP Management profile .**

2. **Enter a command of the following format:**

   `#` **`pntadm -M`** *ip-address* **`-f UNUSABLE`** *network-address*

   For example, to mark address `10.64.3.3` as unusable, type:

   `pntadm -M 10.64.3.3 -f UNUSABLE 10.64.3.0`

Deleting IP Addresses from DHCP Service

You should delete IP addresses from the DHCP service database if you no longer want the address to be managed by DHCP. You can use the `pntadm -D` command or DHCP Manager's Delete Address dialog box.

The following figure shows the Delete Address dialog box.

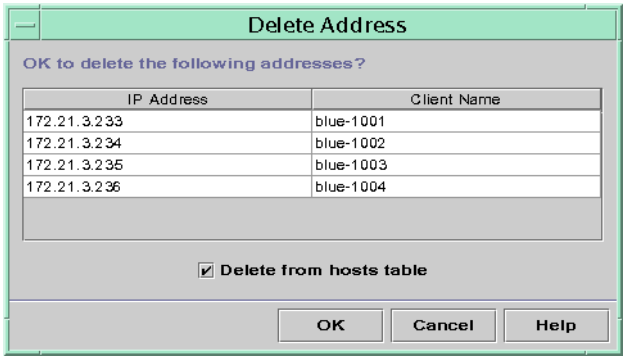


FIGURE 10–12 Delete Address Dialog Box

▼ How to Delete IP Addresses from DHCP Service (DHCP Manager)

1. **Select the Addresses tab.**
2. **Select the IP address's network.**
3. **Select one or more IP addresses you want to delete.**

   If you want to delete more than one address, press the Control key while you click with the mouse to select multiple addresses. You can also press the Shift key while you click to select a block of addresses.
4. **Choose Delete from the Edit menu.**

   The Delete Address dialog box lists the address you selected so you can confirm the deletion.
5. **If you want to delete the host names from the hosts table, select Delete From Hosts Table.**

   If the host names were generated by DHCP Manager, you might want to delete the names from the hosts table.
6. **Click OK.**

▼ How to Delete IP Addresses from DHCP Service (`pntadm`)

1. **Become superuser or a user assigned to the DHCP Management profile .**
2. **Type a command of the following format:**

   # **`pntadm -D`** *ip-address*

   If you include the `-y` option, the host name is deleted from the name service in which it is maintained.

Setting Up DHCP Clients for a Consistent IP Address

The Solaris DHCP service attempts to provide the same IP address to a client that has previously obtained an address through DHCP. However, it is not always possible when a dynamic lease is used.

Routers, NIS/NIS+, DNS servers, and other hosts critical to the network should not use DHCP because they should not rely on the network to obtain their IP addresses. Clients such as print or file servers should have consistent IP addresses as well, but can be set up to receive their network configurations through DHCP.

You can set up a client to receive the same IP address each time it requests its configuration if you reserve, or manually assign, the client's ID to the address you want it to use. You can set up the reserved address to use a dynamic lease to make it easy to track the use of the address, or a permanent lease if you do not need to track address use. However, you might not want to use permanent leases because once a client obtains a permanent lease, it does not contact the server again and cannot obtain updated configuration information unless it releases the IP address and restarts the DHCP lease negotiation. A diskless client is an example of a client that should use a reserved address with a dynamic lease.

You can use the `pntadm -M` command or DHCP Manager's Address Properties dialog box.

The following figure shows the Lease tab of the Address Properties dialog box used to modify the lease.

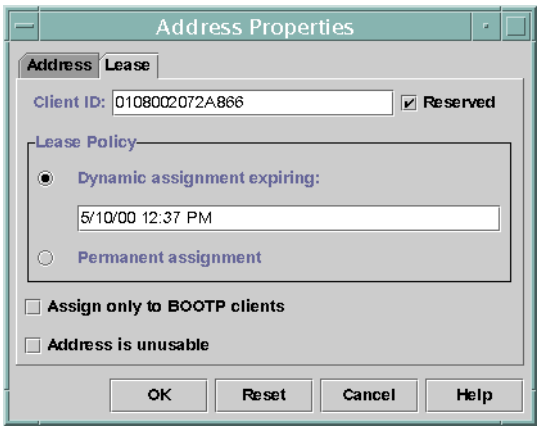


FIGURE 10–13 Address Properties Lease Tab

▼ How to Assign a Consistent IP Address to a DHCP Client (DHCP Manager)

1. **Determine the client ID for the client you want to have a permanent IP address.**
   See the entry for client ID in Table 10–2 for information about how to determine the client ID.
2. **Select the Addresses tab in DHCP Manager.**
3. **Select the appropriate network.**
4. **Double-click the IP address you want to the client to use.**
   The Address Properties window opens.
5. **Select the Lease tab.**
6. **In the Client ID field, type the client ID you determined from the client's hardware address.**
   See the Client ID entry in Table 10–2 for more information.
7. **Select the Reserved option to prevent the IP address from being reclaimed by the server.**
8. **In the Lease Policy area of the window, select Dynamic or Permanent assignment.**
   Select Dynamic if you want the client to negotiate to renew leases, and thus be able to track when the address is used. Because you selected Reserved, the address cannot be reclaimed even when it uses a dynamic lease. You do not need to enter an expiration

date for this lease. The DHCP server calculates the expiration date based on the lease time.

If you select Permanent, you cannot track the use of the IP address unless you enable transaction logging.

▼ How to Assign a Consistent IP Address to a DHCP Client (`pntadm`)

1. **Become superuser or a user assigned to the DHCP Management profile .**
2. **Type a command of the following format:**

   # **`pntadm -M`** *ip-address* **`-i`** *client-id* **`-f MANUAL+BOOTP`** *network-address*

   Refer to the Client ID entry in Table 10–2 for more information about how to determine client identifiers.

Working With DHCP Macros (Task Map)

DHCP macros are containers of DHCP options. The Solaris DHCP service uses macros to gather together options that should be passed to clients. DHCP Manager and `dhcpconfig` create a number of macros automatically when you configure the server. See "About Macros" on page 127 for background information about macros, and Chapter 9 for information about macros created by default.

You might find that when changes occur on your network, you need to make changes to the configuration information passed to clients. To do this, you need to work with DHCP macros. You can view, create, modify, duplicate, and delete DHCP macros.

When you work with macros, you must know about DHCP standard options, which are described in the `dhcp_inittab` man page.

The following task map lists tasks to help you view, create, modify, and delete DHCP macros.

Task	Description	Instructions
View DHCP macros.	Display a list of all the macros defined on the DHCP server.	"How to View Macros Defined on a DHCP Server (DHCP Manager)" on page 209 "How to View Macros Defined on a DHCP Server (`dhtadm`)" on page 210
Create DHCP macros.	Create new macros to support DHCP clients.	"How to Create a DHCP Macro (DHCP Manager)" on page 214 "How to Create a DHCP Macro (`dhtadm`)" on page 215
Modify values passed in macros to DHCP clients.	Change macros by modifying existing options, adding options to macros, removing options from macros.	"How to Change Values for Options in a DHCP Macro (DHCP Manager)" on page 211 "How to Change Values for Options in a DHCP Macro (`dhtadm`)" on page 211 "How to Add Options to a DHCP Macro (DHCP Manager)" on page 212 "How to Add Options to a DHCP Macro (`dhtadm`)" on page 212 "How to Delete Options from a DHCP Macro (DHCP Manager)" on page 213 "How to Delete Options from a DHCP Macro (`dhtadm`)" on page 213
Delete DHCP macros.	Remove DHCP macros that are no longer used.	"How to Delete a DHCP Macro (DHCP Manager)" on page 216 "How to Delete a DHCP Macro (`dhtadm`)" on page 216

The following figure shows the Macros tab in the DHCP Manager window.

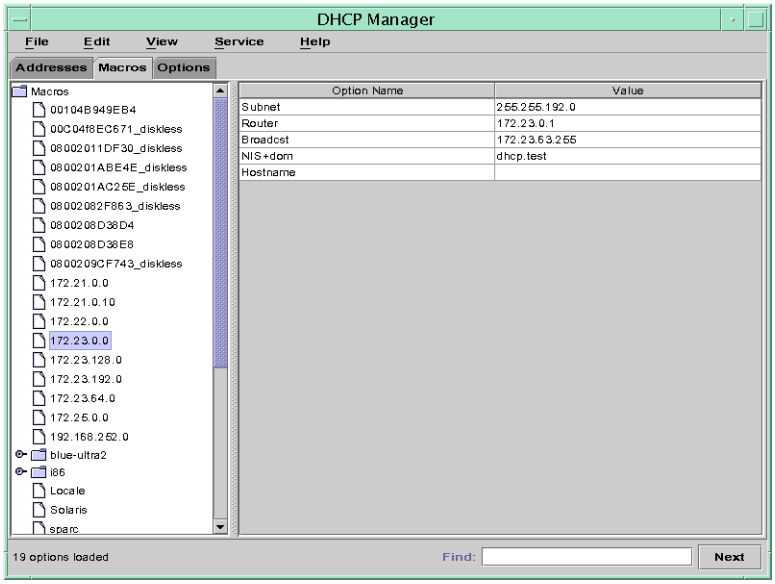


FIGURE 10–14 DHCP Manager's Macros Tab

▼ How to View Macros Defined on a DHCP Server (DHCP Manager)

1. **Select the Macros tab.**

   The Macros area on the left side of the window displays, in alphabetical order, all macros defined on the server. Macros preceded by a folder icon include references to other macros, while macros preceded by a document icon do not reference other macros.

2. **To open a macro folder, click the open/close widget to the left of the folder icon.**

   The macros included in the selected macro are listed.

3. **To view the contents of a macro, click the macro name and look at the area on the right side of the window.**

   Options and their assigned values are displayed.

▼ How to View Macros Defined on a DHCP Server (`dhtadm`)

1. **Become superuser or a user assigned to the DHCP Management profile .**
2. **Type the following command:**

   ```
   # dhtadm -P
   ```

   This command prints to standard output the formatted contents of the `dhcptab`, including all macros and symbols defined on the server.

Modifying DHCP Macros

You might need to modify macros when some aspect of your network changes and one or more clients need to know about the change. For example, you might add a router or a NIS server, create a new subnet, or decide to change the lease policy.

When you modify a macro, you must know the name of the DHCP option that corresponds to the parameter you want to change, add, or delete. The standard DHCP options are listed in the DHCP Manager help and in the `dhcp_inittab` man page.

You can use the `dhtadm -M -m` command or DHCP Manager to modify macros. See the `dhtadm` man page for more information about `dhtadm`.

The following figure shows DHCP Manager's Macro Properties dialog box.

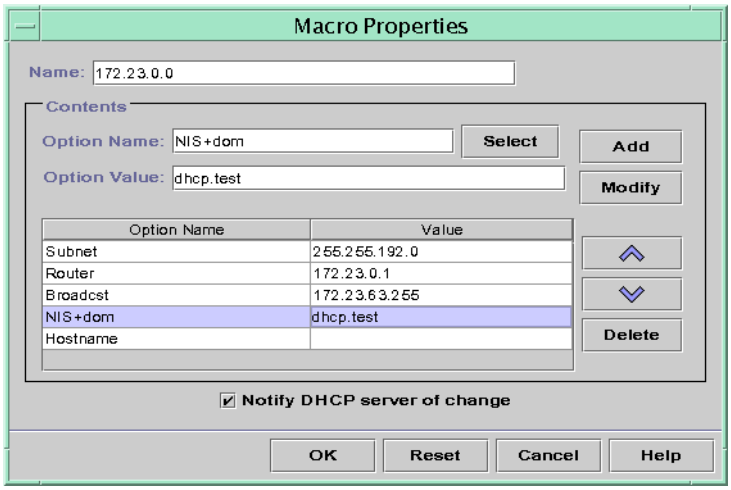


FIGURE 10–15 Macro Properties Dialog Box

▼ How to Change Values for Options in a DHCP Macro (DHCP Manager)

1. **Select the Macros tab.**
2. **Select the macro you want to change.**
3. **Choose Properties from the Edit menu.**
   The Macro Properties dialog box opens.
4. **In the table of Options, select the option you want to change.**
   The option's name and value are displayed in the Option Name and Option Value fields.
5. **In the Option Value field, select the old value and type the new value for the option.**
6. **Click Modify.**
   The new value is displayed in the options table.
7. **Select Notify DHCP Server of Change.**
   This selection tells the DHCP server to reread the `dhcptab` to put the change into effect immediately after you click OK.
8. **Click OK.**

▼ How to Change Values for Options in a DHCP Macro (`dhtadm`)

1. **Become superuser or a user assigned to the DHCP Management profile .**
2. **Type a command of the following format:**

   # **dhtadm -M -m** *macroname* **-e '***option***=***value***:***option***=***value***'**

   For example, to change the lease time and the Universal Time Offset in macro `bluenote`, type the following command:

   # dhtadm -M -m bluenote -e 'LeaseTim=43200:UTCOffst=28800'

▼ How to Add Options to a DHCP Macro (DHCP Manager)

1. **Select the Macros tab.**
2. **Select the macro you want to change.**
3. **Choose Properties from the Edit menu.**

   The Macro Properties dialog box opens.
4. **In the Option Name field, specify the name of an option by using one of the following methods:**

   a. **Click the Select button next to the Option Name field and select the option you want to add to the macro.**

      The Select Option dialog box displays an alphabetized list of names of Standard category options and descriptions. If you want to add an option that is not in the Standard category, use the Category list to select the category you want.

      See "About Macros" on page 127 for more information about macro categories.

   b. **Type `Include` if you want to include a reference to an existing macro in the new macro.**
5. **Type the value for the option in the Option Value field.**

   If you typed **`Include`** as the option name, you must specify the name of an existing macro in the Option Value field.
6. **Click Add.**

   The option is added to the bottom of the list of options displayed for this macro. If you want to change the option's position in the list, select the option and click the arrow keys next to the list to move the option up or down.
7. **Select Notify DHCP Server of Change.**

   This selection tells the DHCP server to reread the `dhcptab` to put the change into effect immediately after you click OK.
8. **Click OK.**

▼ How to Add Options to a DHCP Macro (`dhtadm`)

1. **Become superuser or a user assigned to the DHCP Management profile .**
2. **Type a command of the following format:**

   # **`dhtadm -M -m`** *macroname* **`-e '`***option=value***`'`**

For example, to add the ability to negotiate leases, in macro `bluenote`, type the following command:

```
# dhtadm -M -m bluenote -e 'LeaseNeg=_NULL_VALUE'
```

Note that if an option does not require a value, you must use `_NULL_VALUE` as the value for the option.

▼ How to Delete Options from a DHCP Macro (DHCP Manager)

1. **Select the Macros tab.**
2. **Select the macro you want to change.**
3. **Choose Properties from the Edit menu.**

   The Macro Properties dialog box opens.
4. **Select the option you want to remove from the macro.**
5. **Click Delete.**

   The option is removed from the list of options for this macro.
6. **Select Notify DHCP Server of Change.**

   This selection tells the DHCP server to reread the `dhcptab` to put the change into effect immediately after you click OK.
7. **Click OK.**

▼ How to Delete Options from a DHCP Macro (`dhtadm`)

1. **Become superuser or a user assigned to the DHCP Management profile .**
2. **Type a command of the following format:**

   # **dhtadm -M -m** *macroname* **-e '***option***='**

   For example, to remove the ability to negotiate leases in macro `bluenote`, type the following command:

   ```
   # dhtadm -M -m bluenote -e 'LeaseNeg='
   ```

   If an option is specified with no value, it is removed from the macro.

Creating DHCP Macros

You may want to add new macros to your DHCP service to support clients with specific needs. You can use the `dhtadm -A -m` command or DHCP Manager's Create Macro dialog box to add macros. See the `dhtadm` man page for more information about the `dhtadm` command.

The following figure shows DHCP Manager's Create Macro dialog box.

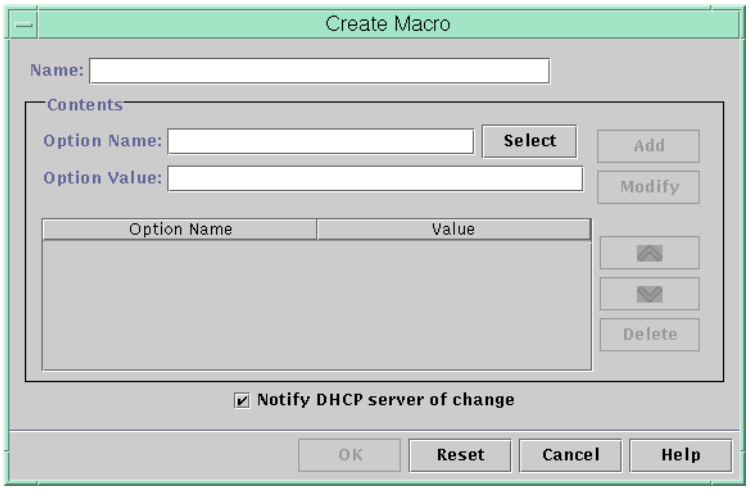


FIGURE 10–16 Create Macro Dialog Box

▼ How to Create a DHCP Macro (DHCP Manager)

1. **Select the Macros tab.**
2. **Choose Create from the Edit menu.**

   The Create Macro dialog box opens.
3. **Type a unique name for the macro.**

   The name can be up to 128 alphanumeric characters. If you use a name that matches a vendor class identifier, network address, or client ID, the macro will be processed automatically for appropriate clients. If you use a different name, the macro can only be processed if it is assigned to a specific IP address or included in another macro that is processed. See "Macro Processing by the DHCP Server" on page 127 for more detailed information.
4. **Click the Select button next to the Option Name field.**

The Select Option dialog box displays an alphabetized list of names of Standard category options and their descriptions.

5. **If you want to add an option that is not in the Standard category, use the Category list to select the category you want.**

   See "About Options" on page 126 for more information about option categories.

6. **Select the option you want to add to the macro and click OK.**

   The Macro Properties dialog box displays the selected option in the Option Name field.

7. **Type the value for the option in the Option Value field.**

8. **Click Add.**

   The option is added to the bottom of the list of options displayed for this macro. If you want to change the option's position in the list, select the option and click the arrow keys next to the list to move the option up or down.

9. **Repeat Step 6 through Step 8 for each option you want to add to the macro.**

10. **Select Notify DHCP Server of Change when you are finished adding options.**

    This selection tells the DHCP server to reread the `dhcptab` to put the change into effect immediately after you click OK.

11. **Click OK.**

▼ How to Create a DHCP Macro (`dhtadm`)

1. **Become superuser or a user assigned to the DHCP Management profile .**

2. **Type a command of the following format:**

   # **dhtadm -A -m** *macroname* **-d ':***option***=***value***:***option***=***value***:***option***=***value:***'**

   There is no limit to the number of option/value pairs included in the argument to `-d`. The argument must begin and end with colons, with colons separating each option/value pair.

   For example, to create macro `bluenote`, type the following command:

   ```
   # dhtadm -A -m bluenote -d \
   ':Router=10.63.6.121:LeaseNeg=_NULL_VALUE:'DNSserv=10.63.28.12:'
   ```

   Note that if an option does not require a value, you must use `_NULL_VALUE` as the value for the option.

Deleting DHCP Macros

You might want to delete a macro from the DHCP service. For example, if you delete a network from the DHCP service, you can also delete the associated network macro.

You can use the `dhtadm -D -m` command or DHCP Manager to delete macros.

▼ How to Delete a DHCP Macro (DHCP Manager)

1. **Select the Macros tab.**
2. **Select the macro you want to delete.**
   The Delete Macro dialog box prompts you to confirm that you want to delete the specified macro.
3. **Select Notify DHCP Server of Change.**
4. **Click OK.**

▼ How to Delete a DHCP Macro (`dhtadm`)

1. **Become superuser or a user assigned to the DHCP Management profile .**
2. **Type a command of the following format:**

   # **`dhtadm -D -m`** *macroname*

   For example, to delete macro `bluenote`, type the following command:

   # **`dhtadm -D -m bluenote`**

Working With DHCP Options (Task Map)

Options are keywords for network configuration parameters that the DHCP server can pass to clients. In the Solaris DHCP service, the only options that you can create, delete, or modify are those that are not specified as standard options in the Solaris DHCP service. For this reason, when you first set up your DHCP service, the Options tab in DHCP Manager is empty until you create options for your site.

If you create options on the DHCP server, you must also add information about the options on the DHCP client. For the Solaris DHCP client, you must edit the `/etc/dhcp/inittab` file to add entries for the new options. See the `dhcp_inittab` man page for more information about this file.

If you have DHCP clients that are not Solaris clients, refer to the documentation for those clients for information about adding new options or symbols. See "About Options" on page 126 for more information about options in Solaris DHCP.

You can use either DHCP Manager or the `dhtadm` command to create, modify, or delete options.

Note – Options are called *symbols* in the DHCP literature. The `dhtadm` command and man page also refer to options as symbols.

The following task map lists tasks you must perform to create, modify, and delete DHCP options and the procedures needed to carry them out.

Task	Description	Instructions
Create DHCP options.	Add new options for information not covered by a standard DHCP option.	"How to Create DHCP Options (DHCP Manager)" on page 220 "How to Create DHCP Options (`dhtadm`)" on page 220 "Modifying the Solaris DHCP Client's Option Information" on page 224
Modify DHCP options.	Change properties of DHCP options you have created.	"How to Modify DHCP Option Properties (DHCP Manager)" on page 222 "How to Modify DHCP Option Properties (`dhtadm`)" on page 222
Delete DHCP options.	Remove DHCP options you have created.	"How to Delete DHCP Options (DHCP Manager)" on page 223 "How to Delete DHCP Options (`dhtadm`)" on page 224

Before you create options, you should be familiar with the option properties listed in the following table.

TABLE 10–3 DHCP Option Properties

Option Properties	Description
Category	The category of an option must be one of the following: Vendor – Options specific to a client's vendor platform, either hardware or software. Site – Options specific to your site. Extend – Newer options that have been added to the DHCP protocol, but not yet implemented as standard options in Solaris DHCP.
Code	The code is a unique number you assign to an option. The same code cannot be used for any other option within its option category. The code must be appropriate for the option category: Vendor – Code values of 1–254 for each vendor class Site – Code values of 128–254 Extend – Code values of 77-127
Data type	The data type specifies what kind of data can be assigned as a value for the option. Valid data types are: ASCII – Text string value. BOOLEAN – No value is associated with the Boolean data type. The presence of the option indicates a condition is true, while the absence of the option indicates false. For example, the Hostname option (which is a Standard option and cannot be modified) is a Boolean. If it is included in a macro, it tells the DHCP server that it should consult name services to see if there is a host name associated with the assigned address. IP – One or more IP addresses, in dotted decimal format (*xxx.xxx.xxx.xxx*). OCTET – Uninterpreted hexadecimal ASCII representation of binary data. For example, a client ID uses the octet data type. UNUMBER8, UNUMBER16, UNUMBER32, UNUMBER64, SNUMBER8, SNUMBER16, SNUMBER32, or SNUMBER64 – Numeric value. An initial U or S indicates whether the number is unsigned or signed, and the digits at the end indicates the amount of bits in the number.
Granularity	Specifies how many "instances" of the data type are needed to represent a complete option value. For example, a data type of IP and a granularity of 2 would mean that the option value must contain two IP addresses.

TABLE 10–3 DHCP Option Properties *(Continued)*

Option Properties	Description
Maximum	The maximum number of values that can be specified for the option. Building on the previous example, a maximum of 2, with a granularity of 2 and a data type of IP Address would mean that the option value could contain a maximum of two pairs of IP addresses.
Vendor client classes	This option is available only when the option category is Vendor. It identifies the client class(es) with which the Vendor option is associated. The Class is an ASCII string that represents the client machine type and/or operating system, for example, `SUNW.Ultra5_10`. This type of option makes it possible to define configuration parameters that are passed to all clients of the same class, and *only* clients of that class. You can specify multiple client classes. Only those DHCP clients with a client class value that matches one you specify will receive the options scoped by that class. The client class is determined by the vendor of the DHCP client. For DHCP clients that are not Solaris clients, refer to the vendor documentation for the DHCP client for the client class. For Solaris clients, the Vendor client class can be obtained by typing `uname -i` on the client. To specify the Vendor client class, substitute periods for any commas in the string returned by the `uname` command. For example, if the string `SUNW,Ultra5_10` is returned by the `uname -i` command, you should specify the Vendor client class as `SUNW.Ultra5_10`.

Creating DHCP Options

If you need to pass client information for which there is not already an existing option in the DHCP protocol, you can create an option. See the `dhcp_inittab` man page for a list of all the options that are defined in Solaris DHCP before you create your own.

You can use the `dhtadm -A -s` command or DHCP Manager's Create Option dialog box to create new options.

The following figure shows DHCP Manager's Create Option dialog box.

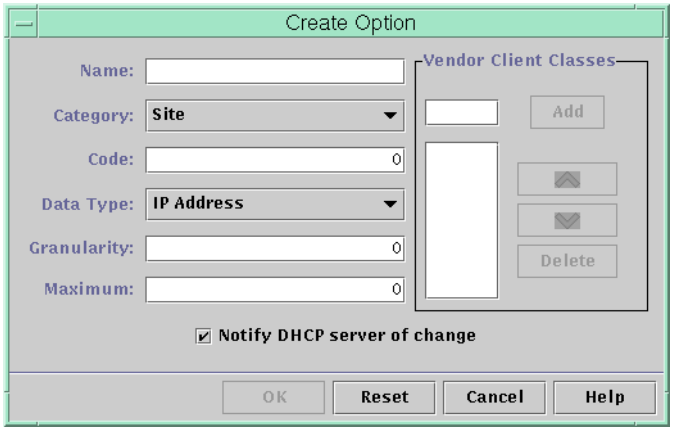


FIGURE 10–17 Create Option Dialog Box

▼ How to Create DHCP Options (DHCP Manager)

1. **Select the Options tab.**
2. **Choose Create from the Edit menu.**
   The Create Options dialog box opens.
3. **Type a short descriptive name for the new option.**
   The name may contain up to 128 alphanumeric characters including spaces.
4. **Type or select values for each setting in the dialog box.**
   Refer to Table 10–3 for information about each setting.
5. **Select Notify DHCP Server of Change if you are finished creating options.**
6. **Click OK.**
   You can now add the option to macros and assign a value to the option to pass to clients.

▼ How to Create DHCP Options (`dhtadm`)

1. **Become superuser or a user assigned to the DHCP Management profile .**
2. **Type a command using the following format:**

   # **`dhtadm -A -s`** *option-name*`-d '`*category,code,data-type,granularity,maximum*`'`

   where

option-name	is an alphanumeric string of 128 characters or less.
category	is `Site`, `Extend`, or `Vendor=`*list-of-classes*, and *list-of-classes* is a space-separated list of vendor client classes to which the option applies. See Table 10–3 for information about how to determine the vendor client class.
code	is a numeric value appropriate to the option category, as explained in Table 10–3.
data-type	is a keyword that indicates the type of data passed with the option, as explained in Table 10–3.
granularity	is a nonnegative number, as explained in Table 10–3.
maximum	is a nonnegative number, as explained in as explained in Table 10–3.

The following two commands are examples:

```
# dhtadm -A -s NewOpt -d 'Site,130,UNUMBER8,1,1'

# dhtadm -A -s NewServ -d 'Vendor=SUNW.Ultra-1 \
SUNW.SPARCstation10,200,IP,1,1'
```

Modifying DHCP Options

If you have created options for your DHCP service, you can change the properties for an option by using either DHCP Manager or the `dhtadm` command.

You can use the `dhtadm -M -s` command or DHCP Manager's Option Properties dialog box to modify options.

Note that you should modify the Solaris DHCP client's option information to reflect the same modification you make to the DHCP service. See "Modifying the Solaris DHCP Client's Option Information" on page 224.

The following figure shows DHCP Manager's Option Properties dialog box.

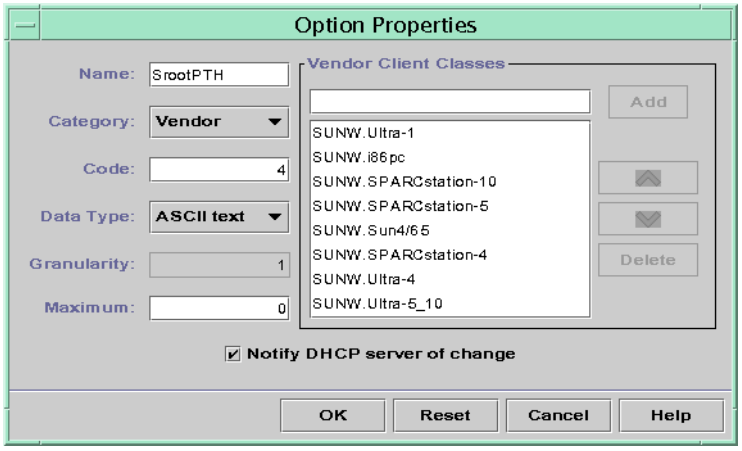


FIGURE 10–18 Option Properties Dialog Box

▼ How to Modify DHCP Option Properties (DHCP Manager)

1. **Select the Options tab.**
2. **Select the option whose properties you want to change.**
3. **Choose Properties from the Edit menu.**
   The Option Properties dialog box opens.
4. **Edit the properties as needed.**
   See Table 10–3 for information about the properties.
5. **Select Notify Server of Change when you are finished with options.**
6. **Click OK.**

▼ How to Modify DHCP Option Properties (`dhtadm`)

1. **Become superuser or a user assigned to the DHCP Management profile .**
2. **Type a command using the following format:**

   # **`dhtadm -M -s`** *option-name*`-d '`*category,code,data-type,granularity,maximum*`'`
   where

option-name	is the option name whose definition you want to change.
category	is `Site`, `Extend`, or `Vendor=`*list-of-classes*, and *list-of-classes* is a space-separated list of vendor client classes to which the option applies. For example, `SUNW.Ultra5_10 SUNW.Ultra-1 SUNWi86pc`.
code	is a numeric value appropriate to the option category, as explained in Table 10–3.
data-type	is a keyword that indicates the type of data passed with the option, as explained in Table 10–3.
granularity	is a nonnegative number, as explained in Table 10–3.
maximum	is a nonnegative number, as explained in as explained in Table 10–3.

Note that you must specify all of the DHCP option properties with the `-d` switch, not just the properties you want to change.

The following two commands are examples:

```
# dhtadm -M -s NewOpt -d 'Site,135,UNUMBER8,1,1'
# dhtadm -M -s NewServ -d 'Vendor=SUNW.Ultra-1 \
SUNW.i86pc,200,IP,1,1'
```

Deleting DHCP Options

You cannot delete standard DHCP options, but if you have defined options for your DHCP service, you can delete them by using DHCP Manager or the `dhtadm` command.

▼ How to Delete DHCP Options (DHCP Manager)

1. **Select the Options tab.**
2. **Choose Delete from the Edit menu.**
   The Delete Options dialog box opens.
3. **Confirm the deletion by clicking OK.**

▼ How to Delete DHCP Options (`dhtadm`)

1. **Become superuser or a user assigned to the DHCP Management profile .**
2. **Type a command using the following format:**

   # **`dhtadm -D -s`** *option-name*

Modifying the Solaris DHCP Client's Option Information

If you add a new DHCP option to your DHCP server, you must add a complementary entry to each DHCP client's option information. If you have a DHCP client that is not a Solaris DHCP client, refer to that client's documentation for information about adding options or symbols.

On a Solaris DHCP client, you must edit the `/etc/dhcp/inittab` file and add an entry for each option that you add to the DHCP server. If you later modify the option on the server, you must also modify the entry in the client's `/etc/dhcp/inittab` file.

Refer to the `dhcp_inittab` man page for detailed information about the syntax of the `/etc/dhcp/inittab` file.

Note – If you added DHCP options to the `dhcptags` file in a previous release of Solaris DHCP, you must add the options to the `/etc/dhcp/inittab` file. See "DHCP Option Information" on page 274 for more information.

Supporting Solaris Network Installation with the DHCP Service (Task Map)

You can use DHCP to install the Solaris operating environment on certain client systems on your network. Only Sun Enterprise Ultra systems and Intel systems that meet the hardware requirements for running the Solaris operating environment can use this feature.

For information about supporting diskless clients, see "Supporting Remote Boot and Diskless Boot Clients (Task Map)" on page 231.

The following task map shows the high-level tasks that must be performed to enable clients to obtain installation parameters using DHCP.

Task	Description	Instructions
Set up an install server.	Set up a Solaris server to support clients that must install the Solaris operating environment from the network.	"Preparing to Install Solaris Software From the Network (Overview)" in *Solaris 9 Installation Guide*
Set up client systems for Solaris installation over the network using DHCP.	Use `add_install_client -d` to add DHCP network installation support for a class of client (such as those of a certain machine type) or a particular client ID.	Using Solaris DVD: "Adding Systems to Be Installed From the Network" in *Solaris 9 Installation Guide* Using Solaris CD: "Adding Systems to Be Installed From the Network" in *Solaris 9 Installation Guide* `add_install_client`(1M)
Create DHCP options for installation parameters and macros that include the options.	Use DHCP Manager or `dhtadm` to create new Vendor options and macros which the DHCP server can use to pass installation information to the clients.	"Creating DHCP Options and Macros for Solaris Installation Parameters" on page 225

Creating DHCP Options and Macros for Solaris Installation Parameters

When you add clients with the `add_install_client -d` script on the install server, the script reports DHCP configuration information to standard output. This information can be used when you create the options and macros needed to pass network installation information to clients.

To support clients that require Solaris installation from the network, you must create Vendor category options to pass information that is needed to correctly install the Solaris operating environment. The following table shows the options you must create and the properties needed to create them.

TABLE 10–4 Values for Creating Vendor Category Options for Solaris Clients

Name	Code	Data Type	Granularity	Maximum	Vendor Client Classes *	Description
SrootOpt	1	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	NFS mount options for the client's root file system
SrootIP4	2	IP address	1	1	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	IP address of root server
SrootNM	3	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Host name of root server
SrootPTH	4	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Path to the client's root directory on the root server
SswapIP4	5	IP address	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	IP address of swap server
SswapPTH	6	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Path to the client's swap file on the swap server
SbootFIL	7	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Path to the client's boot file
Stz	8	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Time zone for client
SbootRS	9	NUMBER	2	1	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	NFS read size used by standalone boot program when it loads the kernel
SinstIP4	10	IP address	1	1	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	IP address of Jumpstart Install server
SinstNM	11	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Host name of install server
SinstPTH	12	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Path to installation image on install server

TABLE 10–4 Values for Creating Vendor Category Options for Solaris Clients *(Continued)*

Name	Code	Data Type	Granularity	Maximum	Vendor Client Classes *	Description
SsysidCF	13	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Path to sysidcfg file, in the format *server:/path*
SjumpsCF	14	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Path to JumpStart configuration file in the format *server:/path*
Sterm	15	ASCII text	1	0	SUNW.Ultra-1, SUNW.Ultra-30, SUNW.i86pc	Terminal type

* The vendor client classes determine what classes of client can use the option. Vendor client classes listed here are suggestions only. You should specify client classes that indicate the actual clients in your network that need to install from the network. See Table 10–3 for information about how to determine a client's vendor client class.

When you have created the options, you can create macros that include those options. The following table lists suggested macros you can create to support Solaris installation for clients.

TABLE 10–5 Suggested Macros to Support Network Installation Clients

Macro Name	Contains These Options and Macros
Solaris	SrootIP4, SrootNM, SinstIP4, SinstNM, Sterm
sparc	SrootPTH, SinstPTH
sun4u	Solaris and sparc macros
i86pc	Solaris macro, SrootPTH, SinstPTH, SbootFIL
SUNW.i86pc *	i86pc macro
SUNW.Ultra-1 *	sun4u macro, SbootFIL
SUNW.Ultra-30 *	sun4u macro, SbootFIL macro
xxx.xxx.xxx.xxx (network address macros)	BootSrvA option could be added to existing network address macros. The value of BootSrvA should indicate the tftboot server.

* These macro names match the Vendor client classes of the clients that will install from the network. These names are examples of clients you might have on your network. See Table 10–3 for information about determining a client's vendor client class.

You can create these options and macros by using the dhtadm command or DHCP Manager. If you use dhtadm, it is better to create the options and macros by writing a script that uses the dhtadm command repeatedly.

The following section, “Writing a Script That Uses dhtadm to Create Options and Macros” on page 228, shows a sample script that uses the dhtadm command. If you prefer to use DHCP Manager, see “Using DHCP Manager to Create Install Options and Macros” on page 229.

Writing a Script That Uses dhtadm to Create Options and Macros

You can create a Korn shell script by adapting the example in Example 10–1 to create all the options listed in Table 10–4 and some useful macros. Be sure to change all IP addresses and values contained in quotes to the correct IP addresses, server names, and paths for your network. You should also edit the Vendor= key to indicate the class of clients you have. Use the information reported by add_install_client -d to obtain the data needed to adapt the script.

EXAMPLE 10–1 Sample Script to Support Network Installation

```
# Load the Solaris vendor specific options. We'll start out supporting
# the Ultra-1, Ultra-30, and i86 platforms. Changing -A to -M would replace
# the current values, rather than add them.
dhtadm -A -s SrootOpt -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,1,ASCII,1,0'
dhtadm -A -s SrootIP4 -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,2,IP,1,1'
dhtadm -A -s SrootNM -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,3,ASCII,1,0'
dhtadm -A -s SrootPTH -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,4,ASCII,1,0'
dhtadm -A -s SswapIP4 -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,5,IP,1,0'
dhtadm -A -s SswapPTH -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,6,ASCII,1,0'
dhtadm -A -s SbootFIL -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,7,ASCII,1,0'
dhtadm -A -s Stz -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,8,ASCII,1,0'
dhtadm -A -s SbootRS -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,9,NUMBER,2,1'
dhtadm -A -s SinstIP4 -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,10,IP,1,1'
dhtadm -A -s SinstNM -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,11,ASCII,1,0'
dhtadm -A -s SinstPTH -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,12,ASCII,1,0'
dhtadm -A -s SsysidCF -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,13,ASCII,1,0'
dhtadm -A -s SjumpsCF -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,14,ASCII,1,0'
dhtadm -A -s Sterm -d 'Vendor=SUNW.Ultra-1 SUNW.Ultra-30 SUNW.i86pc,15,ASCII,1,0'
# Load some useful Macro definitions
# Define all Solaris-generic options under this macro named Solaris.
dhtadm -A -m Solaris -d ':SrootIP4=10.21.0.2:SrootNM="blue2":SinstIP4=10.21.0.2:\
SinstNM="red5":Sterm="xterm":'
# Define all sparc-platform specific options under this macro named sparc.
dhtadm -A -m sparc -d ':SrootPTH="/export/sparc/root":SinstPTH="/export/sparc/install":'
# Define all sun4u architecture-specific options under this macro named sun4u. (Includes
# Solaris and sparc macros.)
dhtadm -A -m sun4u -d ':Include=Solaris:Include=sparc:'
# Solaris on IA32-platform-specific parameters are under this macro named i86pc.
dhtadm -A -m i86pc -d \
':Include=Solaris:SrootPTH="/export/i86pc/root":SinstPTH="/export/i86pc/install"\
:SbootFIL="/platform/i86pc/kernel/unix":'
# Solaris on IA32 machines are identified by the "SUNW.i86pc" class. All
# clients identifying themselves as members of this class will see these
```

EXAMPLE 10–1 Sample Script to Support Network Installation *(Continued)*

```
# parameters in the macro called SUNW.i86pc, which includes the i86pc macro.
dhtadm -A -m SUNW.i86pc -d ':Include=i86pc:'
# Ultra-1 platforms identify themselves as part of the "SUNW.Ultra-1" class.
# By default, we boot these machines in 32bit mode. All clients identifying
# themselves as members of this class will see these parameters.
dhtadm -A -m SUNW.Ultra-1 -d ':SbootFIL="/platform/sun4u/kernel/unix":Include=sun4u:'
# Ultra-30 platforms identify themselves as part of the "SUNW.Ultra-30" class.
# By default, we will boot these machines in 64bit mode. All clients
# identifying themselves as members of this class will see these parameters.
dhtadm -A -m SUNW.Ultra-30 -d ':SbootFIL="/platform/sun4u/kernel/sparcv9/unix":\
Include=sun4u:'
# Add our boot server IP to each of the network macros for our topology served by our
# DHCP server. Our boot server happens to be the same machine running our DHCP server.
dhtadm -M -m 10.20.64.64 -e BootSrvA=10.21.0.2
dhtadm -M -m 10.20.64.0 -e BootSrvA=10.21.0.2
dhtadm -M -m 10.20.64.128 -e BootSrvA=10.21.0.2
dhtadm -M -m 10.21.0.0 -e BootSrvA=10.21.0.2
dhtadm -M -m 10.22.0.0    -e BootSrvA=10.21.0.2
# Make sure we return host names to our clients.
dhtadm -M -m DHCP-servername -e Hostname=_NULL_VALUE_
# The client with this MAC address is a diskless client. Override the root settings
# which at the network scope setup for Install with our client's root directory.
dhtadm -A -m 0800201AC25E -d \
':SrootIP4=10.23.128.2:SrootNM="orange-svr-2":SrootPTH="/export/root/10.23.128.12":'
```

As superuser, execute `dhtadm` in batch mode and specify the name of the script to add the options and macros to your `dhcptab`. For example, if your script is named `netinstalloptions`, type the command:

```
dhtadm -B netinstalloptions
```

When you have done this, clients that have vendor client classes that are listed in the `Vendor=` string can use DHCP to obtain the parameters they need for Solaris installation over the network.

Using DHCP Manager to Create Install Options and Macros

You can create the options listed in Table 10–4 and the macros listed in Table 10–5 with DHCP Manager.

See Figure 10–17 and Figure 10–16 for illustrations of the dialog boxes you use to create options and macros.

▼ How to Create Options to Support Solaris Installation (DHCP Manager)

1. **Select the Options tab in DHCP Manager.**
2. **Choose Create from the Edit menu.**
   The Create Option dialog box opens.
3. **Type the option name for the first option and type values appropriate for that option.**
   Use Table 10–4 to look up the option names and values for options you must create. Notice that the vendor client classes are only suggested values. You should create classes to indicate the actual client types that need to obtain Solaris installation parameters from the DHCP service. See Table 10–3 for information about how to determine a client's vendor client class.
4. **Click OK when you have entered all the values.**
5. **In the Options tab, select the option you just created.**
6. **Select Duplicate from the Edit menu.**
   The Duplicate Option dialog box opens.
7. **Type the name of another option and modify other values appropriately.**
   The values for code, data type, granularity, and maximum are most likely to need modification. See Table 10–4 for the values.
8. **Repeat Step 5 through Step 7 until you have created all the options.**
   You can now create macros to pass the options to network installation clients, as explained in the following procedure.

Note – You do not need to add these options to a Solaris client's `/etc/dhcp/inittab` file because they are already included in that file.

▼ How to Create Macros to Support Solaris Installation (DHCP Manager)

1. **Select the Macros tab in DHCP Manager.**
2. **Choose Create from the Edit menu.**
   The Create Macro dialog box opens.
3. **Type the name of a macro.**

See Table 10–5 for macro names you might use.

4. **Click the Select button.**
   The Select Option dialog box opens.
5. **Select Vendor in the Category list.**
   The Vendor options you created are listed.
6. **Select an option you want to add to the macro and click OK.**
7. **Type a value for the option.**
   See Table 10–4 for the option's data type and refer to the information reported by `add_install_client -d`.
8. **Repeat Step 6 through Step 7 for each option you want to include.**
   To include another macro, type **`Include`** as the option name and type the macro name as the option value.
9. **Click OK when the macro is complete.**

Supporting Remote Boot and Diskless Boot Clients (Task Map)

The Solaris DHCP service can support Solaris client systems that mount their operating system files remotely from another machine, called the OS server. Such clients are often called diskless clients. They can be thought of as persistent remote boot clients in that each time they boot, they must obtain the name and IP address of the server that hosts their operating system files, and then boot remotely from those files.

Each diskless client has its own root partition on the OS server, which is shared to the client host name. This means that the DHCP server must always return the same IP address to the client, and that address must remain mapped to the same host name in the name service (such as DNS). To accomplish this, each diskless client must be assigned a consistent IP address.

In addition to the IP address and host name, the DHCP server can supply a diskless client with all the information needed to locate its operating system files on the OS server. However, you must create options and macros that can be used to pass the information in a DHCP message packet.

The following task map lists the tasks required to support diskless clients or any other persistent remote boot clients, and includes links to procedures to help you carry them out.

Task	Description	Instructions
Set up OS services on a Solaris server.	Use the `smosservice` command to create operating system files for clients.	"Managing Diskless Client Support (Tasks)" in *System Administration Guide: Basic Administration* Also see the `smosservice` man page.
Set up DHCP Service to support network boot clients	Use DHCP Manager or `dhtadm` to create new Vendor options and macros which the DHCP server can use to pass booting information to the clients. Note that if you already created the options for network install clients, you need only create macros for the Vendor client types of the diskless clients.	"Supporting Solaris Network Installation with the DHCP Service (Task Map)" on page 224
Assign reserved IP addresses to the diskless clients.	Use DHCP Manager or `pntadm` to mark addresses reserved (or manual) for diskless clients.	"Setting Up DHCP Clients for a Consistent IP Address" on page 205
Set up diskless clients for OS service	Use the `smdiskless` command to add operating system support on the OS server for each client. Specify the IP addresses you reserved for each client.	"Managing Diskless Client Support (Tasks)" in *System Administration Guide: Basic Administration* Also see the `smdiskless` man page
Assign reserved IP addresses to the diskless clients.	Use DHCP Manager or `pntadm` to mark addresses reserved (or manual) for diskless clients.	"Setting Up DHCP Clients for a Consistent IP Address" on page 205
Set up diskless clients for OS service	Use the `smdiskless` command to add operating system support on the OS server for each client. Specify the IP addresses you reserved for each client.	"Managing Diskless Client Support (Tasks)" in *System Administration Guide: Basic Administration* Also see the `smdiskless` man page

Setting Up DHCP Clients as NIS+ Clients

You can use the NIS+ name service on Solaris systems that are DHCP clients, but to do so requires you to partially circumvent one of the security-enhancing features of NIS+ - the creation of DES credentials. When you set up a NIS+ client that is *not* using DHCP, you add unique DES credentials for the new NIS+ client system to the `cred` table on the NIS+ server. There are several ways to accomplish this, such as using the `nisclient` script or the `nisaddcred` command.

For DHCP clients, you *cannot* use these methods because they depend on a static host name to create and store the credentials. If you want to use NIS+ and DHCP, you must create identical credentials to be used for all the host names of DHCP clients. In this way, no matter what IP address (and associated host name) a DHCP client receives, it can use the same DES credentials.

Note – Before you do this, remember that NIS+ was designed with security in mind, and this procedure weakens that security because it allows random DHCP clients to receive NIS+ credentials.

The following procedure shows you how to create identical credentials for all DHCP host names. This procedure is only valid if you know the host names that DHCP clients will use, such as when the host names are generated by the DHCP server.

▼ How to Set Up Solaris DHCP Clients as NIS+ Clients

A DHCP client workstation that is to be a NIS+ client must use credentials copied from another NIS+ client workstation in the NIS+ domain. This procedure only produces credentials for the workstation, which apply only to the superuser logged in to the workstation. Other users logged in to the DHCP client workstation must have their own unique credentials in the NIS+ server, created according to the procedure in the *System Administration Guide: Naming and Directory Services (FNS and NIS+)*.

1. **Type the following command on the NIS+ server to write the `cred` table entry for the NIS+ client to a temporary file.**

   # **`nisgrep`** *nisplus-client-name* **`cred.org_dir > /tmp/`***file*

2. **View the contents of the temporary file so you can copy the credentials and use them to create credentials for DHCP clients.**

You must copy the public key and private key, which are long strings of numbers and letters separated by colons.

3. **Type the following commands to add credentials for a DHCP client. Copy the public and private key information from the temporary file.**

```
# nistbladm -a cname=" dhcp-client-name@nisplus-domain" auth_type=DES \
auth_name="unix.dhcp-client-name@nisplus-domain" \
public_data=copied-public-data \
private_data=copied-private-data
```

4. **Type the following commands on each DHCP client system to remote copy NIS+ client files to the DHCP client system.**

```
# rcp nisplus-client-name:/var/nis/NIS_COLD_START /var/nis
# rcp nisplus-client-name:/etc/.rootkey /etc
# rcp nisplus-client-name:/etc/defaultdomain /etc
```

   If you get a "permission denied" message, the systems may not be set up to allow remote copying. You can copy the files as a regular user to an intermediate location and then copy them to the proper location as root on the DHCP client systems.

5. **Type the following command on the DHCP client system to use the correct name service switch file for NIS+:**

```
# cp /etc/nsswitch.nisplus /etc/nsswitch.conf
```

6. **Reboot the DHCP client system.**

   The DHCP client system should now be able to use NIS+ services.

Example – Setting up a Solaris DHCP Client as an NIS+ Client

The following example assumes that you have one workstation, `nisei`, which is a NIS+ client in the NIS+ domain `dev.example.net`, and one DHCP client, `dhow`, that you want to be a NIS+ client.

```
    (first log in as root on the NIS+ server)
# nisgrep nisei cred.org_dir > /tmp/nisei-cred
# cat /tmp/nisei-cred
nisei.dev.example.net.:DES:unix.nisei@dev.example.net:46199279911a84045b8e0
c76822179138173a20edbd8eab4:90f2e2bb6ffe7e3547346dda624ec4c7f0fe1d5f37e21cff63830
c05bc1c724b
# nistbladm -a cname="dhow@dev.example.net." \
auth_type=DES auth_name="unix.dhow@dev.example.net" \
public_data=46199279911a84045b8e0c76822179138173a20edbd8eab4 \
private_data=90f2e2bb6ffe7e3547346dda624ec4c7f0fe1d5f37e21cff63830\
c05bc1c724b
# rlogin dhow
    (log in as root on dhow)
# rcp nisei:/var/nis/NIS_COLD_START /var/nis
```

```
# rcp nisei:/etc/.rootkey /etc
# rcp nisei:/etc/defaultdomain /etc
# cp /etc/nsswitch.nisplus /etc/nsswitch.conf
# reboot
```

The DHCP client system `dhow` should now be able to use NIS+ services.

Adding Credentials With a Script

If you want to set up a large number of DHCP clients as NIS+ clients, you can write a script to quickly add the entries to the `cred` table. The following sample shows how this might be done.

EXAMPLE 10–2 Sample Script for Adding Credentials for DHCP Clients

```
#! /usr/bin/ksh
#
# Copyright (c) by Sun Microsystems, Inc. All rights reserved.
#
# Sample script for cloning a credential. Hosts file is already populated
# with entries of the form dhcp-[0-9][0-9][0-9]. The entry we're cloning
# is dhcp-001.
#
#
PUBLIC_DATA=6e72878d8dc095a8b5aea951733d6ea91b4ec59e136bd3b3
PRIVATE_DATA=3a86729b685e2b2320cd7e26d4f1519ee070a60620a93e48a8682c5031058df4
HOST="dhcp-"
DOMAIN="mydomain.example.com"

for
i in 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019
do
     print - ${HOST}${i}
     #nistbladm -r [cname="${HOST}${i}.${DOMAIN}."]cred.org_dir
     nistbladm -a cname="${HOST}${i}.${DOMAIN}." \
         auth_type=DES auth_name="unix.${HOST}${i}@${DOMAIN}" \
         public_data=${PUBLIC_DATA} private_data=${PRIVATE_DTA} cred.org_Dir
done

exit 0
```

Converting to a New Data Store

Solaris DHCP provides a utility to convert the DHCP configuration data from one data store to another. You may need to convert to a new data store if, for example, your number of DHCP clients increases to the point that you need higher performance or

higher capacity from the DHCP service, or if you want to share the DHCP server duties among multiple servers. See “Choosing the Data Store” on page 142 for a comparison of the relative benefits and drawbacks of each type of data store.

Note – If you upgraded from a Solaris release older than the Solaris 8 7/01 release on the DHCP server system, the first time you run any Solaris DHCP management tool after Solaris installation, you are prompted to convert your DHCP data tables to the new data store. The conversion is required because the format of the data stored in both files and NIS+ changed in the Solaris 8 7/01 release. If you do not convert to the new data store, the DHCP server continues to read the old data tables to extend leases for existing clients. You cannot register new DHCP clients or use management tools with the old data tables.

The conversion utility is also useful for sites converting from a Sun-provided data store to a third-party data store. The conversion utility looks up entries in the existing data store and adds new entries that contain the same data to the new data store. Data store access is implemented in separate modules for each data store, which enables the conversion utility to convert DHCP data from any data store format to any other data store format, provided each data store has a module. See *Solaris DHCP Service Developer's Guide* for more information about how to write a module to support a third-party data store.

The data store conversion can be accomplished with DHCP Manager through the Data Store Conversion wizard, or with the `dhcpconfig -C` command.

The initial dialog box of the Data Store Conversion wizard is shown in the following figure.

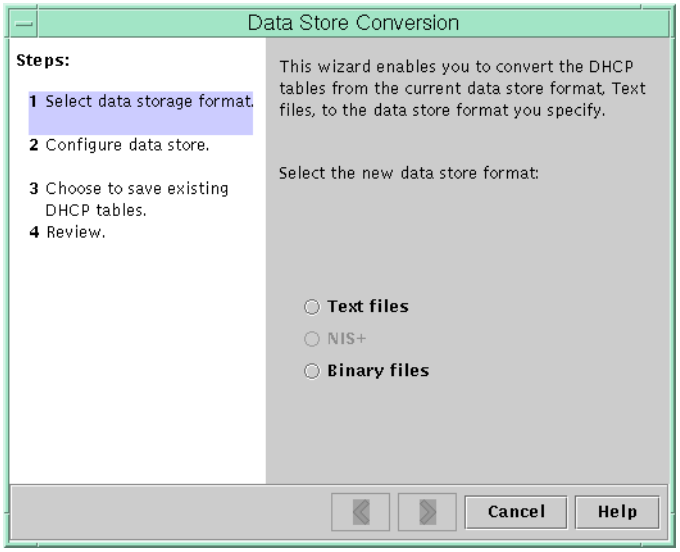


FIGURE 10–19 Data Store Conversion Wizard Dialog Box

Before the conversion begins, you must specify whether to save the old data store's tables (`dhcptab` and network tables) . The conversion utility then stops the DHCP server, converts the data store, and restarts the server when the conversion has completed successfully. If you did not specify to save the old tables, the utility deletes them after it determines the conversion is successful. The process of converting can be time-consuming, so the conversion runs in the background with a meter to inform you of its progress.

▼ How to Convert the DHCP Data Store (DHCP Manager)

1. **Choose Convert Data Store from the Service menu.**
   The Data Store Conversion wizard opens.
2. **Answer the wizard's prompts.**
   If you have trouble providing the requested information, click Help to view detailed information about each dialog box.

▼ How to Convert the DHCP Data Store (`dhcpconfig -C`)

1. **Become superuser or a user assigned to the DHCP Management profile .**
2. **Type a command of the following format:**

   # **`/usr/sbin/dhcpconfig -C -r`** *resource* **`-p`** *path*

   where *resource* is the data store (such as `SUNWbinfiles`) and *path* is the path to the data (such as `/var/dhcp`).

   Note that if you want to keep the original data (in the old data store) after the conversion, specify the `-k` option.

Moving Configuration Data Between DHCP Servers (Task Map)

The DHCP Manager and `dhcpconfig` utilities enable you to move some or all the DHCP configuration data from one Solaris DHCP server to another. You can move entire networks and all the addresses, macros, and options associated with it, or select specific IP addresses, macros, and options to move. You can also copy useful macros or options without removing them from the first server when you specify to keep the data on the server.

You might want to move data if you are going to do any of the following tasks:

- Add a server to share DHCP duties
- Replace the DHCP server's system
- Change the path for the data store (while still using the same data store)

The following task map identifies the procedures you must perform when you move DHCP configuration data.

Task	Description	Instructions
1. Export the data from the first server	Select the data you want to move to another server and create a file of exported data.	"How to Export Data From a DHCP Server (DHCP Manager)" on page 240 "How to Export Data From a DHCP Server (`dhcpconfig -x`)" on page 242

Task	Description	Instructions
2. Import the data to the second server	Copy exported data to another DHCP server's data store.	"How to Import Data On a DHCP Server (DHCP Manager)" on page 241 "How to Import Data on a DHCP Server (`dhcpconfig -I`)" on page 242
3. Modify the imported data for the new server environment	Change server-specific configuration data to match the new server's information.	"How to Modify Imported DHCP Data (DHCP Manager)" on page 241 "How to Modify Imported DHCP Data (`pntadm`, `dhtadm`)" on page 243

In DHCP Manager, you use the Export Data wizard and Import Data wizard to move the data from one server to the other, and modify macros in the Macros tab. The following figures show the initial dialog boxes for the wizards.

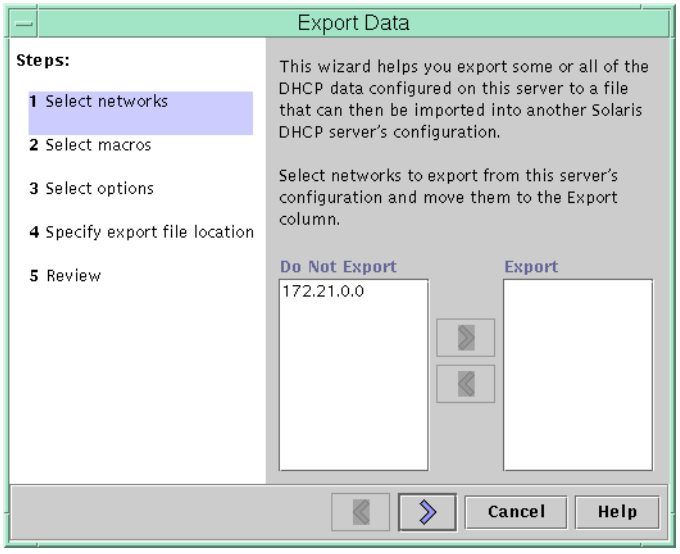


FIGURE 10–20 Export Data Wizard Dialog Box

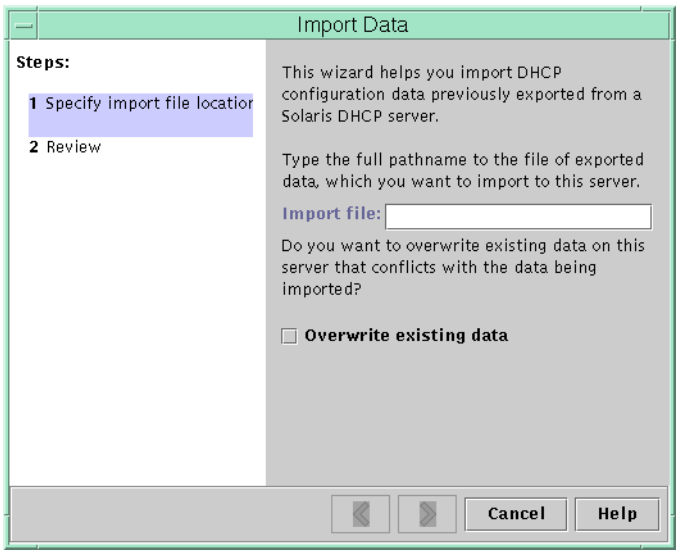


FIGURE 10–21 Import Data Wizard Dialog Box

▼ How to Export Data From a DHCP Server (DHCP Manager)

1. **Become superuser on the server from which you want to move or copy data.**
2. **Choose Export Data on the Service menu.**
   The Export Data wizard opens as shown in Figure 10–20.
3. **Answer the wizard's prompts.**
   If you have difficulty, click Help for detailed information about the prompts.
4. **Move the export file to a file system that is accessible to the DHCP server to which you want to move the data.**
   Import the data as described in "How to Import Data On a DHCP Server (DHCP Manager)" on page 241.

▼ How to Import Data On a DHCP Server (DHCP Manager)

1. **Become superuser on the server to which you want to move data that you previously exported from a DHCP server.**
2. **In DHCP Manager, choose Import Data from the Service menu.**
   The Import Data Wizard opens, as shown in Figure 10–21.
3. **Answer the wizard's prompts.**
   If you have difficulty, click Help for detailed information about the prompts.

▼ How to Modify Imported DHCP Data (DHCP Manager)

1. **Become superuser on the server to which you imported data.**
2. **Examine imported data for network–specific information that needs modification.**
   For example, if you moved networks, you must open the Addresses tab and change the owning server of addresses in the imported networks. You might also need to open the Macros tab to specify the correct domain names for NIS, NIS+ or DNS in the macros that specify them.
3. **Open the Addresses tab and select a network that you imported.**
4. **To select all the addresses, click the first address, press and hold the Shift key, and click the last address.**
5. **From the Edit menu, choose Properties.**
   The Modify Multiple Addresses dialog box opens.
6. **At the Managing Server prompt, select the new server's name.**
7. **At the Configuration Macro prompt, select the macro that should be used for all clients on this network.**
8. **Click OK.**
9. **Open the Macros tab.**
10. **Use the Find facility at the bottom of the window to locate the options that are likely to need modified values.**
    `DNSdmain`, `DNSserv`, `NISservs`, `NIS+serv`, and `NISdmain` are examples of options that might need modification on the new server.

11. **When you locate an option that needs to be changed, select the macro name and choose Properties from the Edit menu and change its value.**

▼ How to Export Data From a DHCP Server (`dhcpconfig -X`)

1. **Become superuser on the server from which you want to move or copy data.**
2. **Type a command of the following format:**

   # **`/usr/sbin/dhcpconfig -X`** *filename* **`-a`** *network-addresses* **`-m`** *macros* **`-o`** *options*

   where *filename* is the full path name you want to use to store the compressed exported data. You can use the keyword `ALL` with the command options to export all the networks, macros, or options. For example:

   ```
   # /usr/sbin/dhcpconfig -X dhcp1065_data -a ALL -m ALL -o ALL
   ```

   Alternatively, you can specify particular network addresses, macros, and configuration options in comma-separated lists. For example:

   ```
   # /usr/sbin/dhcpconfig -X dhcp1065_data -a 10.63.0.0,10.62.0.0 \
   -m 10.63.0.0,10.62.0.0,SUNW.Ultra-5_10 -o Sterm
   ```

   See the `dhcpconfig` man page for more information about the `dhcpconfig` command.

3. **Move the file that contains the exported data to a location that is accessible to the server to which you want to move the data.**

   Import the data as described in "How to Import Data on a DHCP Server (`dhcpconfig -I`)" on page 242.

▼ How to Import Data on a DHCP Server (`dhcpconfig -I`)

1. **Become superuser on the server to which you want to import the data.**
2. **Type a command of the following format:**

   # **`/usr/sbin/dhcpconfig -I`** *filename*

   where *filename* is the name of the file that contains the data exported from the first server.

   Be sure to modify the imported data as described in "How to Modify Imported DHCP Data (`pntadm`, `dhtadm`)" on page 243

▼ How to Modify Imported DHCP Data (`pntadm`, `dhtadm`)

1. **Become superuser on the server to which you imported data.**

2. **Examine the network tables for data that needs to be modified.**

   If you moved networks, use `pntadm -P` *network_address* to print out the network tables for the networks you moved.

3. **Use the `pntadm` command to modify IP address information.**

   You might need to change the owning server and the configuration macro used for imported addresses. For example, to change the owning server (`10.60.3.4`) and macro (`dhcpsrv-1060`) for address `10.63.0.2`, you would use the following command:

   ```
   pntadm -M 10.63.0.2 -s 10.60.3.4 -m dhcpsrv-1060 10.60.0.0
   ```

   If you have a large number of addresses, you should create a script file that contains commands to modify each address, and then execute the script with the `pntadm -B` command, which runs `pntadm` in batch mode. See the `pntadm` man page.

4. **Examine the `dhcptab` macros for options with values that need modification.**

   Use `dhtadm -P` to print the entire `dhcptab`, and use `grep` or some other tool to search for particular options or values that you might want to change.

5. **Use the `dhtadm -M` command to modify options in macros if necessary.**

   For example, you might need to modify some macros to specify the correct domain names and servers for NIS, NIS+ or DNS. For example, the following command changes the values of `DNSdmain` and `DNSserv` in the macro `mymacro`:

   ```
   dhtadm -M -m mymacro -e 'DNSserv=dnssrv2:DNSdmain=example.net'
   ```

CHAPTER 11

Troubleshooting DHCP (Reference)

This chapter provides information to help you solve problems you might encounter when you configure a DHCP server or client, or problems in using DHCP after configuration is complete.

The chapter includes the following information:

Troubleshooting DHCP Server Problems

The problems you might encounter when you configure the server fall into the following categories:

- NIS+, if you choose to use NIS+ for your data store
- IP address allocation

NIS+ Problems

If you decide to use NIS+ as the DHCP data store, problems you might encounter can be categorized as follows:

- Cannot select NIS+ as a data store
- NIS+ is not adequately configured
- NIS+ access problems due to insufficient permissions and credentials

Cannot Select NIS+ as a Data Store

If you try to use NIS+ as your data store, you might find that DHCP Manager does not offer it as a choice for data store, or `dhcpconfig` returns a message saying NIS+ does not appear to be installed and running. This means that NIS+ has not been configured for this server, although NIS+ might be in use on the network. Before you can select NIS+ as a data store, the server system must be configured as an NIS+ client.

Before you set up the server as an NIS+ client, the domain must have already been configured and its master server must be running. The master server of the domain's tables should be populated, and the hosts table must have an entry for the new client system (the DHCP server system). "Setting Up NIS+ Client Machines" in *System Administration Guide: Naming and Directory Services (FNS and NIS+)* provides detailed information about configuring an NIS+ client.

NIS+ Not Adequately Configured

After you successfully use NIS+ with DHCP, you might encounter errors if changes are made to NIS+ and introduce configuration problems. Use the following table to help you determine the cause of configuration problems.

TABLE 11–1 NIS+ Configuration Problems

Possible Problem	Gather Information	Solution
Root object does not exist in the NIS+ domain.	Enter the following command: `/usr/lib/nis/nisstat` This command displays statistics for the domain. If the root object does not exist, no statistics are returned.	Set up the NIS+ domain using the *System Administration Guide: Naming and Directory Services (FNS and NIS+)*.
NIS+ is not used for `passwd` and `publickey` information.	Enter the following command to view the name service switch configuration file: `cat /etc/nsswitch.conf` Check the `passwd` and `publickey` entries for the "nisplus" keyword.	Refer to the *System Administration Guide: Naming and Directory Services (FNS and NIS+)* for information about configuring the name service switch.
The domain name is empty.	Enter the following command: `domainname` If the command lists an empty string, no domain name has been set for the domain.	Use local files for your data store, or set up an NIS+ domain for your network. Refer to *System Administration Guide: Naming and Directory Services (FNS and NIS+)*.

TABLE 11–1 NIS+ Configuration Problems *(Continued)*

Possible Problem	Gather Information	Solution
The `NIS_COLD_START` file does not exist.	Enter the following command on the server system to determine if the file exists: `cat /var/nis/NIS_COLD_START`	Use local files for your data store, or create an NIS+ client. Refer to *System Administration Guide: Naming and Directory Services (FNS and NIS+).*

NIS+ Access Problems

NIS+ access problems might cause error messages about incorrect DES credentials, or inadequate permissions to update NIS+ objects or tables. Use the following table to determine the cause of NIS+ errors you receive.

TABLE 11–2 NIS+ Access Problems

Possible Problem	Gather Information	Solution
The DHCP server system does not have create access to the `org_dir` object in the NIS+ domain.	Enter the following command: `nisls -ld org_dir` The access rights are listed in the form `r---rmcdrmcdr---`, where the permissions apply respectively to nobody, owner, group, and world. The owner of the object is listed next.	Use the `nischmod` command to change the permissions for `org_dir`. For example, to add create access to the group, type the following command: `nischmod g+c org_dir`
	Normally the `org_dir` directory object provides full (read, modify, create, and destroy) rights to both the owner and the group, while providing only read access to the world and nobody classes. The DHCP server name must either be listed as the owner of the `org_dir` object, or be listed as a principal in the group, and that group must have create access. List the group with the command: `nisls -ldg org_dir`	See the `nischmod`(1) man page for more information.

TABLE 11–2 NIS+ Access Problems *(Continued)*

Possible Problem	Gather Information	Solution

The DHCP server does not have access rights to create a table under the `org_dir` object.
 	
Usually, this means the server system's principal name is not a member of the owning group for the `org_dir` object, or no owning group exists.
	Enter this command to find the owning group name:
	
	`niscat -o org_dir`
	
	Look for a line similar to
	
	`Group : "admin.example.com."`
	
	List the principal names in the group using the command:
	
	`nisgrpadm -l` *groupname*
	
	For example:
	
	`nisgrpadm -l admin.example.com`
	
	The server system's name should be listed as an explicit member of the group or included as an implicit member of the group.
		Add the server system's name to the group using the `nisgrpadm` command.
		
		For example, to add the server name `pacific` to the group `admin.example.com`, type the following command:
		
		`nisgrpadm -a admin.example.com pacific.example.com`
		
		See the `nisgrpadm`(1) man page for more information.

The DHCP server does not have valid Data Encryption Standard (DES) credentials in the NIS+ `cred` table.	If this is the problem, an error message states that the user does not have DES credentials in the NIS+ name service.	Use the `nisaddcred` command to add security credentials for the DHCP server system. The following example shows how to add DES credentials for the system `mercury` in the domain `example.com`: `nisaddcred -p unix.mercury@example.com \ -P mercury.example.com. DES example.com`. The command prompts for the root password (which is required to generate an encrypted secret key). See the `nisaddcred`(1M) man page for more information.

IP Address Allocation Errors

When a client attempts to obtain or verify an IP address, you might see the problems in the following table logged to `syslog` or in server debug output.

TABLE 11–3 IP Address Allocation and Lease Problems

Error Message	Explanation	Solution
`There is no` *n.n.n.n* `dhcp-network table for DHCP client's network.`	A client is requesting a specific IP address or seeking to extend a lease on its current IP address but the DHCP server cannot find the DHCP network table for that address.	The DHCP network table might have been deleted mistakenly. You can recreate the network table by adding the network again using DHCP Manager or `dhcpconfig`.
`ICMP ECHO reply to OFFER candidate:` *n.n.n.n*`, disabling`	The IP address considered for offering to a DHCP client is already in use. This might occur if more than one DHCP server owns the address, or if an address was manually configured for a non-DHCP network client.	Determine the proper ownership of the address and correct either the DHCP server database or the host's network configuration.
`ICMP ECHO reply to OFFER candidate:` *n.n.n.n*`. No corresponding dhcp network record.`	The IP address considered for offering to a DHCP client does not have a record in a network table. This might occur if the IP address record is deleted from the DHCP network table after the address was selected but before the duplicate address check was completed.	Use DHCP Manager or `pntadm` to view the DHCP network table, and if the IP address is missing, create it with DHCP Manager (choose Create from the Edit menu on the Address tab) or `pntadm`.
`DHCP network record for` *n.n.n.n* `is unavailable, ignoring request.`	The record for the requested IP address is not in the DHCP network table, so the server is dropping the request.	Use DHCP Manager or `pntadm` to view the DHCP network table, and if the IP address is missing, create it with DHCP Manager (choose Create from the Edit menu on the Address tab) or `pntadm`.
n.n.n.n `currently marked as unusable.`	The requested IP address cannot be offered because it has been marked in the network table as unusable.	You can use DHCP Manager or `pntadm` to make the address usable.
n.n.n.n `was manually allocated. No dynamic address will be allocated.`	The client's ID has been assigned a manually allocated address, and that address is marked as unusable. The server cannot allocate a different address to this client.	You can use DHCP Manager or `pntadm` to make the address usable, or manually allocate a different address to the client.
`Manual allocation (`*n.n.n.n, client ID* `has` *n* `other records. Should have 0.`	The client that has the specified client ID has been manually assigned more than one IP address. There should be only one. The server selects the last manually assigned address it finds in the network table.	Use DHCP Manager or `pntadm` to modify IP addresses to remove the additional manual allocations.

TABLE 11–3 IP Address Allocation and Lease Problems *(Continued)*

Error Message	Explanation	Solution
`No more IP addresses on` *n.n.n.n* `network.`	All IP addresses currently managed by DHCP on the specified network have been allocated.	Use DHCP Manager or `pntadm` to create new IP addresses for this network.
`Client:` *clientid* `lease on` *n.n.n.n* `expired.`	The lease was not negotiable and timed out.	Client should automatically restart the protocol to obtain a new lease.
`Offer expired for client:` *n.n.n.n*	The server made an IP address offer to the client, but the client took too long to respond and the offer expired.	The client should automatically issue another discover message. If this also times out, increase the cache offer timeout for the DHCP server. In DHCP Manager, choose Modify from the Service menu.
`Client:` *clientid* `REQUEST is missing requested IP option.`	The client's request did not specify the offered IP address, so the DHCP server ignores the request. This might occur if the client is not compliant with the updated DHCP protocol, RFC 2131.	Update client software.
`Client:` *clientid* `is trying to renew` *n.n.n.n*`, an IP address it has not leased.`	The IP address recorded in the DHCP network table for this client does not match the IP address that the client specified in its renewal request. The DHCP server does not renew the lease.	This problem occurs if you delete a client's record while the client is still using the IP address. Use DHCP Manager or `pntadm` to examine the network table, and correct if necessary. The client's ID should be bound to the specified IP address. If it is not, edit the address properties to add the client ID.

TABLE 11–3 IP Address Allocation and Lease Problems *(Continued)*

Error Message	Explanation	Solution

`Client:` *clientid* `is trying to verify unrecorded address:` *n.n.n.n*`, ignored.`	The specified client has not been registered in the DHCP network table with this address, so the request is ignored by this DHCP server.	Another DHCP server on the network might have assigned this client the address.
		
		However, you might also have deleted the client's record while the client was still using the IP address.
		
		Use DHCP Manager or `pntadm` to examine the network table on this server and any other DHCP servers on the network and correct if necessary.
		
		You can also do nothing and allow the lease to expire, after which the client will automatically request a new address lease.
		
		If you want the client to get a new lease immediately, restart the DHCP protocol on the client by typing the following commands:
		
		`ifconfig` *interface* `dhcp release`
		
		`ifconfig` *interface* `dhcp start`

Troubleshooting DHCP Client Configuration Problems

The problems you might encounter with a DHCP client fall into the following categories:

- "Problems Communicating With DHCP Server" on page 251
- "Problems with Inaccurate DHCP Configuration Information" on page 260

Problems Communicating With DHCP Server

This section describes problems you might encounter as you add DHCP clients to the network.

After you enable the client software and reboot the system, the client tries to reach the DHCP server to obtain its network configuration. If the client fails to reach the server, you might see error messages such as:

```
DHCP or BOOTP server not responding
```

Before you can determine the problem you must gather diagnostic information from both the client and the server and analyze the information. To gather information you can:

1. Run the client in debug mode.
2. Run the server in debug mode.
3. Start `snoop` to monitor network traffic.

You can do these things separately or concurrently.

The information you gather can help you determine if the problem is with the client, server, or a relay agent, and then you can find a solution.

▼ How to Run the DHCP Client in Debug Mode

If you have a client that is not a Solaris DHCP client, refer to the client's documentation for information about how to run the client in debug mode.

If you have a Solaris DHCP client, use the following steps.

1. **Become superuser on the client system.**
2. **Type the following commands to kill the DHCP client daemon and restart it in debug mode:**

   ```
   # pkill -x dhcpagent
   # /sbin/dhcpagent -d1 -f &
   # ifconfig interface dhcp start
   ```

   When run in debug mode, the client daemon displays messages to your screen as it performs DHCP requests. See "DHCP Client Debug Output" on page 253 for information about client debug output.

▼ How to Run the DHCP Server in Debug Mode

1. **Become superuser on the server system.**
2. **Type the following commands to kill the DHCP daemon and restart it in debug mode:**

   ```
   # pkill -x in.dhcpd
   # /usr/lib/inet/in.dhcpd -d -v
   ```

You should also use any `in.dhcpd` command-line options that you normally use when you run the daemon. For example, if you run the daemon as a BOOTP relay agent, include the `-r` option with the `in.dhcpd -d -v` command.

When run in debug mode, the daemon displays messages to your screen as it processes DHCP/BOOTP requests. See "DHCP Server Debug Output" on page 254 for information about server debug output.

▼ How to Use `snoop` to Monitor DHCP Network Traffic

1. **Become superuser on the DHCP server system.**
2. **Start `snoop` to begin tracing network traffic across the server's network interface.**

```
# /usr/sbin/snoop -d interface -o snoop-output-filename udp port 67 or udp port 68
```

For example:

```
# /usr/sbin/snoop -d le0 -o /tmp/snoop.output udp port 67 or udp port 68
```

Note that `snoop` continues to monitor the interface until you stop it explicitly by pressing Control-C after you have the information you need.

3. **Boot the client system, or restart the `dhcpagent` on the client system.**

Restarting `dhcpagent` is described in "How to Run the DHCP Client in Debug Mode" on page 252.

4. **On the server system, use `snoop` to display the output file with the contents of network packets:**

```
# /usr/sbin/snoop -i snoop-output-filename -x0 -v
```

For example:

```
# /usr/sbin/snoop -i /tmp/snoop.output -x0 -v
```

The `-d` switch with the `dhcpagent` command puts the client in debug mode with level 1 verbosity, and the `-f` switch causes output to be sent to the console instead of to `syslog`. Replace *interface* in the `ifconfig` command line with the name of the network interface of the client (for example, `le0`).

See "DHCP `snoop` Output" on page 257 for information about interpreting the output.

DHCP Client Debug Output

The following example shows normal debug output when a DHCP client sends its DHCP request and receives its configuration information from a DHCP server.

EXAMPLE 11–1 Sample Normal DHCP Client Debug Output

```
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x27fc8 (DHCP filter)
/sbin/dhcpagent: debug: init_ifs: initted interface le0
/sbin/dhcpagent: debug: insert_ifs: le0: sdumax 1500, optmax 1260, hwtype 1, hwlen 6
/sbin/dhcpagent: debug: insert_ifs: inserted interface le0
/sbin/dhcpagent: debug: register_acknak: registered acknak id 5
/sbin/dhcpagent: debug: unregister_acknak: unregistered acknak id 5
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x26018 (ARP reply filter)
/sbin/dhcpagent: info: setting IP netmask on le0 to 255.255.192.0
/sbin/dhcpagent: info: setting IP address on le0 to 10.23.3.233
/sbin/dhcpagent: info: setting broadcast address on le0 to 10.23.63.255
/sbin/dhcpagent: info: added default router 10.23.0.1 on le0
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x28054 (blackhole filter)
/sbin/dhcpagent: debug: configure_if: bound ifsp->if_sock_ip_fd
/sbin/dhcpagent: info: le0 acquired lease, expires Tue Aug 10 16:18:33 1999
/sbin/dhcpagent: info: le0 begins renewal at Tue Aug 10 15:49:44 1999
/sbin/dhcpagent: info: le0 begins rebinding at Tue Aug 10 16:11:03 1999
```

If the client cannot reach the DHCP server, you might see debug output similar to the following example.

EXAMPLE 11–2 Sample Debug Output for DHCP Client

```
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x27fc8 (DHCP filter)
/sbin/dhcpagent: debug: init_ifs: initted interface le0
/sbin/dhcpagent: debug: select_best: no valid OFFER/BOOTP reply
/sbin/dhcpagent: debug: select_best: no valid OFFER/BOOTP reply
/sbin/dhcpagent: debug: select_best: no valid OFFER/BOOTP reply
```

If you see this message, the request never reached the server, or the server cannot send a response to the client. Run `snoop` on the server as described in "How to Use `snoop` to Monitor DHCP Network Traffic" on page 253 to determine if packets from the client have reached the server.

DHCP Server Debug Output

Normal server debug output shows server configuration information followed by information about each network interface as the daemon starts. After daemon startup, the debug output shows information about requests the daemon processes. Example 11–3 shows debug output for a DHCP server that has just started and then extends the lease for a client that is using an address owned by another DHCP server that is not responding.

EXAMPLE 11–3 Sample Debug Output for DHCP Server

```
Daemon Version: 3.1
Maximum relay hops: 4
```

EXAMPLE 11–3 Sample Debug Output for DHCP Server *(Continued)*

```
Transaction logging to console enabled.
Run mode is: DHCP Server Mode.
Datastore: nisplus
Path: org_dir.dhcp.test..:dhcp.test..:$
DHCP offer TTL: 10
Ethers compatibility enabled.
BOOTP compatibility enabled.
ICMP validation timeout: 1000 milliseconds, Attempts: 2.
Monitor (0005/hme0) started...
Thread Id: 0005 - Monitoring Interface: hme0 *****
MTU: 1500      Type: DLPI
Broadcast: 10.21.255.255
Netmask: 255.255.0.0
Address: 10.21.0.2
Monitor (0006/nf0) started...
Thread Id: 0006 - Monitoring Interface: nf0 *****
MTU: 4352      Type: DLPI
Broadcast: 10.22.255.255
Netmask: 255.255.0.0
Address: 10.22.0.1
Monitor (0007/qe0) started...
Thread Id: 0007 - Monitoring Interface: qe0 *****
MTU: 1500      Type: DLPI
Broadcast: 10.23.63.255
Netmask: 255.255.192.0
Address: 10.23.0.1
Read 33 entries from DHCP macro database on Tue Aug 10 15:10:27 1999
Datagram received on network device: qe0
Client: 0800201DBA3A is requesting verification of address owned by 10.21.0.4
Datagram received on network device: qe0
Client: 0800201DBA3A is requesting verification of address owned by 10.21.0.4
Datagram received on network device: qe0
Client: 0800201DBA3A is requesting verification of address owned by 10.21.0.4
Datagram received on network device: qe0
Client: 0800201DBA3A maps to IP: 10.23.3.233
Unicasting datagram to 10.23.3.233 address.
Adding ARP entry: 10.23.3.233 == 0800201DBA3A
DHCP EXTEND 0934312543 0934316143 10.23.3.233 10.21.0.2
          0800201DBA3A SUNW.SPARCstation-10 0800201DBA3A
```

Example 11–4 shows debug output from a DHCP daemon that starts as a BOOTP relay agent and relays requests from a client to a DHCP server, and relays the servers responses to the client.

EXAMPLE 11–4 Sample Debug Output for BOOTP Relay

```
Relay destination: 10.21.0.4 (blue-servr2)     network: 10.21.0.0
Daemon Version: 3.1
Maximum relay hops: 4
Transaction logging to console enabled.
```

EXAMPLE 11–4 Sample Debug Output for BOOTP Relay *(Continued)*

```
Run mode is: Relay Agent Mode.
Monitor (0005/hme0) started...
Thread Id: 0005 - Monitoring Interface: hme0 *****
MTU: 1500      Type: DLPI
Broadcast: 10.21.255.255
Netmask: 255.255.0.0
Address: 10.21.0.2
Monitor (0006/nf0) started...
Thread Id: 0006 - Monitoring Interface: nf0 *****
MTU: 4352      Type: DLPI
Broadcast: 10.22.255.255
Netmask: 255.255.0.0
Address: 10.22.0.1
Monitor (0007/qe0) started...
Thread Id: 0007 - Monitoring Interface: qe0 *****
MTU: 1500      Type: DLPI
Broadcast: 10.23.63.255
Netmask: 255.255.192.0
Address: 10.23.0.1
Relaying request 0800201DBA3A to 10.21.0.4, server port.
BOOTP RELAY-SRVR 0934297685 0000000000 0.0.0.0 10.21.0.4 0800201DBA3A
N/A 0800201DBA3A
Packet received from relay agent: 10.23.0.1
Relaying reply to client 0800201DBA3A
Unicasting datagram to 10.23.3.233 address.
Adding ARP entry: 10.23.3.233 == 0800201DBA3A
BOOTP RELAY-CLNT 0934297688 0000000000 10.23.0.1 10.23.3.233 0800201DBA3A
N/A 0800201DBA3A
Relaying request 0800201DBA3A to 10.21.0.4, server port.
BOOTP RELAY-SRVR 0934297689 0000000000 0.0.0.0 10.21.0.4 0800201DBA3A
N/A 0800201DBA3A
Packet received from relay agent: 10.23.0.1
Relaying reply to client 0800201DBA3A
Unicasting datagram to 10.23.3.233 address.
Adding ARP entry: 10.23.3.233 == 0800201DBA3A
```

If there is a problem, the debug output might display warnings or error messages. Use Table 11–4 to find error messages and solutions.

TABLE 11–4 DHCP Server Error Messages

Message	Explanation	Solution
`ICMP ECHO reply to OFFER candidate:` *ip_address* `disabling`	Before the DHCP server offers an IP address to a client, it pings the address to verify that the address is not in use. If a client replies, the address is in use.	Make sure the addresses you configured are not already in use.
`No more IP addresses on` *network_address* `network.`	No available IP addresses in the DHCP network table associated with the client's network.	Create more IP addresses using DHCP Manager or `pntadm`. If the DHCP daemon is monitoring multiple subnets, be sure the additional addresses are for the subnet where the client is located.
`No more IP addresses for` *network_address* `network` when you are running the DHCP daemon in BOOTP compatibility mode (`-b` option).	BOOTP does not use a lease time, so the DHCP server looks for free addresses with the BOOTP flag set to allocate to BOOTP clients.	Use DHCP Manager to allocate BOOTP addresses.
`Request to access nonexistent per network database:` *database_name* `in datastore:` *datastore.*	During configuration of the DHCP server, a DHCP network table for a subnet was not created.	Use DHCP Manager or the `pntadm` to create the DHCP network table and new IP addresses.
`There is no` *table_name* `dhcp-network table for DHCP client's network.`	During configuration of the DHCP server, a DHCP network table for a subnet was not created.	Use DHCP Manager or the `pntadm` to create the DHCP network table and new IP addresses.
`Client using non_RFC1048 BOOTP cookie.`	A device on the network is trying to access an unsupported implementation of BOOTP.	Ignore this message, unless you need to configure this device.

DHCP `snoop` Output

In the `snoop` output, you should see that packets are exchanged between the DHCP client system and the DHCP server system. The IP address for each system (and any relay agents or routers in between) is indicated in each packet. If the systems do not exchange packets, the client system might not be able to contact the server system at all, and the problem is at a lower level.

To evaluate `snoop` output, you should know what the expected behavior is (such as if the request should be going through a BOOTP relay agent). You should also know the MAC addresses and IP address of the systems involved (and those of the network interfaces, if there is more than one) so that you can determine if those values are as expected. The following example shows normal `snoop` output for a DHCP acknowledgement message sent from the DHCP server on `blue-servr2` to a client whose MAC address is `8:0:20:8e:f3:7e`. In the message, the servers assigns the client the IP address `172.168.252.6` and the host name `white-6`. The message also includes a number of standard network options and several vendor-specific options for the client.

EXAMPLE 11–5 Sample `snoop` Output for One Packet

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 26 arrived at 14:43:19.14
ETHER:  Packet size = 540 bytes
ETHER:  Destination = 8:0:20:8e:f3:7e, Sun
ETHER:  Source      = 8:0:20:1e:31:c1, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 526 bytes
IP:   Identification = 64667
IP:   Flags = 0x4 IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 254 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 157a
IP:   Source address = 10.21.0.4, blue-servr2
IP:   Destination address = 192.168.252.6, white-6
IP:   No options
IP:   UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 67
UDP:  Destination port = 68 (BOOTPC)
UDP:  Length = 506
UDP:  Checksum = 5D4C
UDP:
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP:
DHCP: Hardware address type (htype) =  1 (Ethernet (10Mb))
DHCP: Hardware address length (hlen) = 6 octets
DHCP: Relay agent hops = 0
```

EXAMPLE 11–5 Sample snoop Output for One Packet *(Continued)*

```
DHCP: Transaction ID = 0x2e210f17
DHCP: Time since boot = 0 seconds
DHCP: Flags = 0x0000
DHCP: Client address (ciaddr) = 0.0.0.0
DHCP: Your client address (yiaddr) = 192.168.252.6
DHCP: Next server address (siaddr) = 10.21.0.2
DHCP: Relay agent address (giaddr) = 0.0.0.0
DHCP: Client hardware address (chaddr) = 08:00:20:11:E0:1B
DHCP:
DHCP: ----- (Options) field options -----
DHCP:
DHCP: Message type = DHCPACK
DHCP: DHCP Server Identifier = 10.21.0.4
DHCP: Subnet Mask = 255.255.255.0
DHCP: Router at = 192.168.252.1
DHCP: Broadcast Address = 192.168.252.255
DHCP: NISPLUS Domainname = dhcp.test
DHCP: IP Address Lease Time = 3600 seconds
DHCP: UTC Time Offset = -14400 seconds
DHCP: RFC868 Time Servers at = 10.21.0.4
DHCP: DNS Domain Name = sem.example.com
DHCP: DNS Servers at = 10.21.0.1
DHCP: Client Hostname = white-6
DHCP: Vendor-specific Options (166 total octets):
DHCP:   (02) 04 octets  0x8194AE1B (unprintable)
DHCP:   (03) 08 octets  "pacific"
DHCP:   (10) 04 octets  0x8194AE1B (unprintable)
DHCP:   (11) 08 octets  "pacific"
DHCP:   (15) 05 octets  "xterm"
DHCP:   (04) 53 octets  "/export/s2/base.s2s/latest/Solaris_8/Tools/Boot"
DHCP:   (12) 32 octets  "/export/s2/base.s2s/latest"
DHCP:   (07) 27 octets  "/platform/sun4m/kernel/unix"
DHCP:   (08) 07 octets  "EST5EDT"
  0: 0800 208e f37e 0800 201e 31c1 0800 4500    .. .ó~.. .1...E.
 16: 020e fc9b 4000 fe11 157a ac15 0004 c0a8    ....@....z......
 32: fc06 0043 0044 01fa 5d4c 0201 0600 2e21    ...C.D..]L.....!
 48: 0f17 0000 0000 0000 0000 c0a8 fc06 ac15    ................
 64: 0002 0000 0000 0800 2011 e01b 0000 0000    ........ .......
 80: 0000 0000 0000 0000 0000 0000 0000 0000    ................
 96: 0000 0000 0000 0000 0000 0000 0000 0000    ................
112: 0000 0000 0000 0000 0000 0000 0000 0000    ................
128: 0000 0000 0000 0000 0000 0000 0000 0000    ................
144: 0000 0000 0000 0000 0000 0000 0000 0000    ................
160: 0000 0000 0000 0000 0000 0000 0000 0000    ................
176: 0000 0000 0000 0000 0000 0000 0000 0000    ................
192: 0000 0000 0000 0000 0000 0000 0000 0000    ................
208: 0000 0000 0000 0000 0000 0000 0000 0000    ................
224: 0000 0000 0000 0000 0000 0000 0000 0000    ................
240: 0000 0000 0000 0000 0000 0000 0000 0000    ................
256: 0000 0000 0000 0000 0000 0000 0000 0000    ................
272: 0000 0000 0000 6382 5363 3501 0536 04ac    ......c.Sc5..6..
288: 1500 0401 04ff ffff 0003 04c0 a8fc 011c    ................
```

EXAMPLE 11–5 Sample `snoop` Output for One Packet *(Continued)*

```
304: 04c0 a8fc ff40 0964 6863 702e 7465 7374    .....@.dhcp.test
320: 3304 0000 0e10 0204 ffff c7c0 0404 ac15    3...............
336: 0004 0f10 736e 742e 6561 7374 2e73 756e    ....sem.example.
352: 2e63 6f6d 0604 ac15 0001 0c07 7768 6974    com.........whit
368: 652d 362b a602 0481 94ae 1b03 0861 746c    e-6+.........pac
384: 616e 7469 630a 0481 94ae 1b0b 0861 746c    ific.........pac
400: 616e 7469 630f 0578 7465 726d 0435 2f65    ific...xterm.5/e
416: 7870 6f72 742f 7332 382f 6261 7365 2e73    xport/sx2/bcvf.s
432: 3238 735f 776f 732f 6c61 7465 7374 2f53    2xs_btf/latest/S
448: 6f6c 6172 6973 5f38 2f54 6f6f 6c73 2f42    olaris_x/Tools/B
464: 6f6f 740c 202f 6578 706f 7274 2f73 3238    oot. /export/s2x
480: 2f62 6173 652e 7332 3873 5f77 6f73 2f6c    /bcvf.s2xs_btf/l
496: 6174 6573 7407 1b2f 706c 6174 666f 726d    atest../platform
512: 2f73 756e 346d 2f6b 6572 6e65 6c2f 756e    /sun4m/kernel/un
528: 6978 0807 4553 5435 4544 54ff              ix..EST5EDT.
```

Problems with Inaccurate DHCP Configuration Information

If a DHCP client receives inaccurate information in its network configuration information, such as the wrong NIS domain name, or incorrect router IP address, you must look at the values of options in the macros that are processed by the DHCP server for this client.

Use the following general guidelines to help you determine the source of the inaccurate information.

- Look at the macros defined on the server as described in "How to View Macros Defined on a DHCP Server (DHCP Manager)" on page 209. Review the information in "Order of Macro Processing" on page 128 and determine which macros are processed automatically for this client.
- Look at the network table to determine what macro (if any) is assigned to the client's IP address as the configuration macro. See "Working With IP Addresses in the DHCP Service (Task Map)" on page 192 for more information.
- Take note of any options that occur in more than one macro and make sure that the value you want for an option is set in the last processed macro.
- Edit the appropriate macro(s) to assure that the correct value is passed to the client. See "Modifying DHCP Macros" on page 210.

Problems with Client-Supplied Host Name

This section describes problems you might experience with DHCP clients that supply their own host names and want the names to be registered with DNS.

Client Does Not Request a Host Name

If your client is not a Solaris DHCP client, consult the client's documentation to determine how to configure the client to request a host name. For Solaris DHCP clients, see "How to Enable a Solaris Client to Request Specific Host Name" on page 179.

DHCP Client Does Not Get Requested Host Name

TABLE 11–5 Problems and Solutions for DHCP Client Host Name Requests

Possible problem	Gather Information	Solution
Client accepted an offer from a DHCP server that does not issue DNS updates.	1. Use `snoop` or other network packet capture application on the client. Look for the DHCP Server Identifier to get the IP address of the server. 2. Log in to the DHCP server to verify that it is configured to make dynamic updates. Look at the `/etc/inet/dhcpsvc.conf` file for the entry `UPDATE_TIMEOUT`. 3. On the DNS server, look at the `/etc/named.conf` file and determine if the DHCP server's IP address is listed in the `allow-update` keyword in the `zone` section of the appropriate domain.	See "Enabling Dynamic DNS Updates by DHCP Server" on page 177 for information about configuring the DHCP server and DNS server. If two DHCP servers are available to the client, the servers should both be configured to provide the DNS updates.

TABLE 11–5 Problems and Solutions for DHCP Client Host Name Requests *(Continued)*

Possible problem	Gather Information	Solution
Client is using FQDN option (option code 89) to specify host name. Solaris DHCP does not currently support FQDN option since it not officially in the DHCP protocol.	Use `snoop` or other network packet capture application on the server, and look for the FQDN option in a packet from client.	Configure the client to specify host name using `Hostname` option (option code 12). Refer to client documentation for instructions.
DHCP server that offers the client its address does not know the client's DNS domain.	On the DHCP server look for the `DNSdmain` option with a valid value.	Set the `DNSdmain` option to the correct DNS domain name in a macro that is processed for this client. `DNSdmain` is usually contained in the network macro.
Host name requested by client corresponds to an IP address that is not managed by the DHCP server. Solaris DHCP servers do not perform DNS updates for IP addresses they do not manage.	Check `syslog` for messages from the DHCP server similar to `There is no` *n.n.n.n* `dhcp-network table for DHCP client's network.` or `DHCP network record for` *n.n.n.n* `is unavailable, ignoring request.`	Configure the client to choose a name for which there is no corresponding IP address, or which corresponds to an address managed by the DHCP server.
Host name requested by client corresponds to an IP address that is currently in use, leased, or under offer to another client.	Check `syslog` for messages from the DHCP server indicating `ICMP ECHO reply to OFFER candidate:` *n.n.n.n*.	Configure the client to choose a name corresponding to a different IP address. Alternatively, reclaim the address from the client that uses the address.
DNS server is not configured to accept updates from the DHCP server.	Examine the `/etc/named.conf` file on the DNS server and look for the DHCP server's IP address with the `allow-update` keyword in the appropriate zone section for the DHCP server's domain.	See "How to Enable Dynamic DNS Updating for DHCP Clients" on page 178 for information about configuring the DNS server. If the DHCP server has multiple interfaces, you may need to configure the DNS server to accept updates from all of the DHCP server's addresses. Enable debugging on the DNS server to see whether the updates are reaching the DNS server; if they are, examine the debugging output to determine why the updates did not occur.

TABLE 11–5 Problems and Solutions for DHCP Client Host Name Requests *(Continued)*

Possible problem	Gather Information	Solution
DNS updates may not have completed in the allotted time. DHCP servers do not return host names to clients if the DNS updates have not completed by the configured time limit. However, attempts to complete the DNS updates continue.	Use the `nslookup` command to determine whether the updates completed successfully. See `nslookup`(1M) man page. For example, if the DNS domain is `hills.example.org`, the DNS server's IP address is 10.76.178.11, and the host name the client wants to register is `cathedral`, you could use the following command to determine if `cathedral` has been registered with DNS: `nslookup cathedral.hills.example.org 10.76.178.11`	If the updates completed successfully, but not in the allotted time, you need to increase the timeout value. See Step 5 in the procedure for enabling DNS updates.

CHAPTER 12

DHCP Files and Commands (Reference)

This chapter explains the relationships between files and the commands that use the files, but does not explain how to use the commands.

The chapter contains the following information:

DHCP Commands

The following table lists the commands you might find useful in managing DHCP on your network.

TABLE 12–1 Commands Used in DHCP

Command	Description
`dhtadm`	Used to make changes to the options and macros in the `dhcptab`. This command is most useful in scripts that you create to automate changes you need to make to your DHCP information. Use `dhtadm` with the `-P` option and pipe it through the `grep` command for a quick way to search for particular option values in the `dhcptab`.
`pntadm`	Used to make changes to the DHCP network tables that map client IDs to IP addresses and optionally associate configuration information with IP addresses.
`dhcpconfig`	Used to configure and unconfigure DHCP servers and BOOTP relay agents, convert to a different data store, and import/export DHCP configuration data.

TABLE 12–1 Commands Used in DHCP *(Continued)*

Command	Description
`in.dhcpd`	The DHCP server daemon. System scripts use this command to start and stop DHCP service. You can start `in.dhcpd` with non-default options, such as `-d` for debugging.
`dhcpmgr`	The DHCP Manager, a graphical tool used to configure and manage the DHCP service. DHCP Manager is the recommended Solaris DHCP management tool.
`ifconfig`	Used at system boot to assign IP addresses to network interfaces, configure network interface parameters, or both. On a Solaris DHCP client, `ifconfig` starts DHCP to get the parameters (including the IP address) needed to configure a network interface.
`dhcpinfo`	Used by system startup scripts on Solaris client systems to obtain information (such as 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.
`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.
`dhcpagent`	The DHCP client daemon, which implements the client side of the DHCP protocol.

Running DHCP Commands in Scripts

The `dhcpconfig`, `dhtadm`, and `pntadm` commands are optimized for use in scripts. In particular, the `pntadm` command is useful for creating a large number of IP address entries in a DHCP network table. The following sample script uses `pntadm` in batch mode to create IP addresses.

EXAMPLE 12–1 `addclient.ksh` Script with the `pntadm` Command

```
#! /usr/bin/ksh
#
# This script utilizes the pntadm batch facility to add client entries
# to a DHCP network table. It assumes that the user has the rights to
# run pntadm to add entries to DHCP network tables.

#
# Based on the nsswitch setting, query the netmasks table for a netmask.
# Accepts one argument, a dotted IP address.
#
get_netmask()
{
    MTMP=‘getent netmasks ${1} | awk ’{ print $2 }’‘
    if [ ! -z "${MTMP}" ]
    then
        print - ${MTMP}
```

EXAMPLE 12–1 addclient.ksh Script with the pntadm Command *(Continued)*

```
    fi
}

#
# Based on the network specification, determine whether or not network is
# subnetted or supernetted.
# Given a dotted IP network number, convert it to the default class
# network.(used to detect subnetting). Requires one argument, the
# network number. (e.g. 10.0.0.0) Echos the default network and default
# mask for success, null if error.
#
get_default_class()
{
    NN01=${1%%.*}
    tmp=${1#*.}
    NN02=${tmp%%.*}
    tmp=${tmp#*.}
    NN03=${tmp%%.*}
    tmp=${tmp#*.}
    NN04=${tmp%%.*}
    RETNET=""
    RETMASK=""

    typeset -i16 ONE=10#${1%%.*}
    typeset -i10 X=$((${ONE}&16#f0))
    if [ ${X} -eq 224 ]
    then
        # Multicast
        typeset -i10 TMP=$((${ONE}&16#f0))
        RETNET="${TMP}.0.0.0"
        RETMASK="240.0.0.0"
    fi
    typeset -i10 X=$((${ONE}&16#80))
    if [ -z "${RETNET}" -a ${X} -eq 0 ]
    then
        # Class A
        RETNET="${NN01}.0.0.0"
        RETMASK="255.0.0.0"
    fi
    typeset -i10 X=$((${ONE}&16#c0))
    if [ -z "${RETNET}" -a ${X} -eq 128 ]
    then
        # Class B
        RETNET="${NN01}.${NN02}.0.0"
        RETMASK="255.255.0.0"
    fi
    typeset -i10 X=$((${ONE}&16#e0))
    if [ -z "${RETNET}" -a ${X} -eq 192 ]
    then
        # Class C
        RETNET="${NN01}.${NN02}.${NN03}.0"
        RETMASK="255.255.255.0"
```

EXAMPLE 12–1 addclient.ksh Script with the pntadm Command *(Continued)*

```
    fi
    print - ${RETNET} ${RETMASK}
    unset NNO1 NNO2 NNO3 NNO4 RETNET RETMASK X ONE
}

#
# Given a dotted form of an IP address, convert it to its hex equivalent.
#
convert_dotted_to_hex()
{
    typeset -i10 one=${1%%.*}
    typeset -i16 one=${one}
    typeset -Z2 one=${one}
    tmp=${1#*.}

    typeset -i10 two=${tmp%%.*}
    typeset -i16 two=${two}
    typeset -Z2 two=${two}
    tmp=${tmp#*.}

    typeset -i10 three=${tmp%%.*}
    typeset -i16 three=${three}
    typeset -Z2 three=${three}
    tmp=${tmp#*.}

    typeset -i10 four=${tmp%%.*}
    typeset -i16 four=${four}
    typeset -Z2 four=${four}

     hex=`print - ${one}${two}${three}${four} | sed -e 's/#/0/g'`
     print - 16#${hex}
     unset one two three four tmp
}

#
# Generate an IP address given the network address, mask, increment.
#
get_addr()
{
    typeset -i16 net=`convert_dotted_to_hex ${1}`
    typeset -i16 mask=`convert_dotted_to_hex ${2}`
    typeset -i16 incr=10#${3}

    # Maximum legal value - invert the mask, add to net.
    typeset -i16 mhosts=~${mask}
    typeset -i16 maxnet=${net}+${mhosts}

    # Add the incr value.
    let net=${net}+${incr}

    if [ $((${net} < ${maxnet})) -eq 1 ]
    then
```

EXAMPLE 12–1 addclient.ksh Script with the pntadm Command *(Continued)*

```
        typeset -i16 a=${net}\&16#ff000000
        typeset -i10 a="${a}>>24"

        typeset -i16 b=${net}\&16#ff0000
        typeset -i10 b="${b}>>16"

        typeset -i16 c=${net}\&16#ff00
        typeset -i10 c="${c}>>8"

        typeset -i10 d=${net}\&16#ff
        print - "${a}.${b}.${c}.${d}"
    fi
    unset net mask incr mhosts maxnet a b c d
}

# Given a network address and client address, return the index.
client_index()
{
    typeset -i NNO1=${1%%.*}
    tmp=${1#*.}
    typeset -i NNO2=${tmp%%.*}
    tmp=${tmp#*.}
    typeset -i NNO3=${tmp%%.*}
    tmp=${tmp#*.}
    typeset -i NNO4=${tmp%%.*}

    typeset -i16 NNF1
    let NNF1=${NNO1}
    typeset -i16 NNF2
    let NNF2=${NNO2}
    typeset -i16 NNF3
    let NNF3=${NNO3}
    typeset -i16 NNF4
    let NNF4=${NNO4}
    typeset +i16 NNF1
    typeset +i16 NNF2
    typeset +i16 NNF3
    typeset +i16 NNF4
    NNF1=${NNF1#16\#}
    NNF2=${NNF2#16\#}
    NNF3=${NNF3#16\#}
    NNF4=${NNF4#16\#}
    if [ ${#NNF1} -eq 1 ]
    then
        NNF1="0${NNF1}"
    fi
    if [ ${#NNF2} -eq 1 ]
    then
        NNF2="0${NNF2}"
    fi
    if [ ${#NNF3} -eq 1 ]
    then
```

EXAMPLE 12–1 addclient.ksh Script with the pntadm Command *(Continued)*

```
    NNF3="0${NNF3}"
fi
if [ ${#NNF4} -eq 1 ]
then
    NNF4="0${NNF4}"
fi
typeset -i16 NN
let NN=16#${NNF1}${NNF2}${NNF3}${NNF4}
unset NNF1 NNF2 NNF3 NNF4

typeset -i NNO1=${2%%.*}
tmp=${2#*.}
typeset -i NNO2=${tmp%%.*}
tmp=${tmp#*.}
typeset -i NNO3=${tmp%%.*}
tmp=${tmp#*.}
typeset -i NNO4=${tmp%%.*}
typeset -i16 NNF1
let NNF1=${NNO1}
typeset -i16 NNF2
let NNF2=${NNO2}
typeset -i16 NNF3
let NNF3=${NNO3}
typeset -i16 NNF4
let NNF4=${NNO4}
typeset +i16 NNF1
typeset +i16 NNF2
typeset +i16 NNF3
typeset +i16 NNF4
NNF1=${NNF1#16\#}
NNF2=${NNF2#16\#}
NNF3=${NNF3#16\#}
NNF4=${NNF4#16\#}
if [ ${#NNF1} -eq 1 ]
then
    NNF1="0${NNF1}"
fi
if [ ${#NNF2} -eq 1 ]
then
    NNF2="0${NNF2}"
fi
if [ ${#NNF3} -eq 1 ]
then
    NNF3="0${NNF3}"
fi
if [ ${#NNF4} -eq 1 ]
then
    NNF4="0${NNF4}"
fi
typeset -i16 NC
let NC=16#${NNF1}${NNF2}${NNF3}${NNF4}
typeset -i10 ANS
```

EXAMPLE 12–1 addclient.ksh Script with the pntadm Command *(Continued)*

```
    let ANS=${NC}-${NN}
    print - $ANS
}

#
# Check usage.
#
if [ "$#" != 3 ]
then
    print "This script is used to add client entries to a DHCP network"
    print "table by utilizing the pntadm batch facilty.\n"
    print "usage: $0 network start_ip entries\n"
    print "where: network is the IP address of the network"
        print "       start_ip is the starting IP address \n"
        print "       entries is the number of the entries to add\n"
    print "example: $0 10.148.174.0 10.148.174.1 254\n"
    return
fi

#
# Use input arguments to set script variables.
#
NETWORK=$1
START_IP=$2
typeset -i STRTNUM=`client_index ${NETWORK} ${START_IP}`
let ENDNUM=${STRTNUM}+$3
let ENTRYNUM=${STRTNUM}
BATCHFILE=/tmp/batchfile.$$
MACRO=`uname -n`

#
# Check if mask in netmasks table. First try
# for network address as given, in case VLSM
# is in use.
#
NETMASK=`get_netmask ${NETWORK}`
if [ -z "${NETMASK}" ]
then
    get_default_class ${NETWORK} | read DEFNET DEFMASK
    # use the default.
    if [ "${DEFNET}" != "${NETWORK}" ]
    then
        # likely subnetted/supernetted.
        print - "\n\n###\tWarning\t###\n"
        print - "Network ${NETWORK} is netmasked, but no entry was found  \n
              in the 'netmasks' table; please update the 'netmasks'  \n
              table in the appropriate nameservice before continuing. \n
              (See /etc/nsswitch.conf.) \n" >&2
        return 1
    else
        # use the default.
        NETMASK="${DEFMASK}"
```

EXAMPLE 12–1 addclient.ksh Script with the pntadm Command *(Continued)*

```
    fi
fi

#
# Create a batch file.
#
print -n "Creating batch file "
while [ ${ENTRYNUM} -lt ${ENDNUM} ]
do
    if [ $((${ENTRYNUM}-${STRTNUM}))%50 -eq 0 ]
    then
        print -n "."
    fi

    CLIENTIP=`get_addr ${NETWORK} ${NETMASK} ${ENTRYNUM}`
    print "pntadm -A ${CLIENTIP} -m ${MACRO} ${NETWORK}" >> ${BATCHFILE}
    let ENTRYNUM=${ENTRYNUM}+1
done
print " done.\n"

#
# Run pntadm in batch mode and redirect output to a temporary file.
# Progress can be monitored by using the output file.
#
print "Batch processing output redirected to ${BATCHFILE}"
print "Batch processing started."

pntadm -B ${BATCHFILE} -v > /tmp/batch.out 2 >&1

print "Batch processing completed."
```

DHCP Files

The following table lists files associated with Solaris DHCP.

TABLE 12–2 Files and Tables Used by DHCP Daemons and Commands

File/Table	Description
dhcptab	A generic term for the table of DHCP configuration information recorded as options with assigned values, which are then grouped into macros. The name of the dhcptab table and its location is determined by the data store you use for DHCP information.

TABLE 12–2 Files and Tables Used by DHCP Daemons and Commands *(Continued)*

File/Table	Description
DHCP network table	Maps IP addresses to client IDs and configuration options. DHCP network tables are named according to the IP address of the network, such as 10.21.32.0. There is no file called `dhcp_network`. The name and location of DHCP network tables is determined by the data store you use for DHCP information.
`dhcpsvc.conf`	Records DHCP daemon startup options and the data store resource and location of the `dhcptab` and network tables. The file is located in the `/etc/inet` directory.
`nsswitch.conf`	Specifies the location of name service databases and the order in which to search them for various kinds of information. The `nsswitch.conf` file is consulted when you configure a DHCP server in order to obtain accurate configuration information. The file is located in the `/etc` directory.
`resolv.conf`	Contains information used by the DNS resolver. During DHCP server configuration, this file is consulted for information about the DNS domain and DNS server. The file is located in the `/etc` directory.
`dhcp.`*interface*	Indicates that DHCP is to be used on the client's network interface specified in the file name, such as `dhcp.qe0`. The `dhcp.`*interface* file might contain commands that are passed as options to the `ifconfig` *interface* `dhcp start` *option* command used to start DHCP on the client. The file is located in the `/etc` directory on Solaris DHCP client systems.
interface`.dhc`	Contains the configuration parameters obtained from DHCP for the given network interface. The client caches the current configuration information in `/etc/dhcp/`*interface*`.dhc` when the interface's IP address lease is dropped. 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 DHCP lease negotiation process.
`dhcpagent`	Sets parameter values for the `dhcpagent` client daemon. The path to the file is `/etc/default/dhcpagent`. See the file itself or the `dhcpagent`(1M) man page for information about the parameters.
DHCP `inittab`	Defines aspects of DHCP option codes, such as the data type, and assigns mnemonic labels. See the `dhcp_inittab` man page for more information about the file syntax. On the client, the information in the `/etc/dhcp/inittab` file is used by `dhcpinfo` to provide more meaningful information to human readers of the information. This file replaces the `/etc/dhcp/dhcptags` file. "DHCP Option Information" on page 274 provides more information about this replacement. On the DHCP server system, this file is used by the DHCP daemon and management tools to obtain DHCP option information.

DHCP Option Information

Historically, DHCP option information has been stored in several places in Solaris DHCP, including the server's `dhcptab` table, the client's `dhcptags` file, and internal tables of `in.dhcpd`, `snoop`, `dhcpinfo`, and `dhcpmgr`. In an effort to consolidate option information, the Solaris 8 DHCP product introduced the `/etc/dhcp/inittab` file. See the dhcp_inittab man page for detailed information about the file.

The Solaris DHCP client uses the DHCP `inittab` file as a replacement for the `dhcptags` file to obtain information about option codes received in its DHCP packet. The `in.dhcpd`, `snoop`, and `dhcpmgr` programs on the DHCP server use the `inittab` file as well.

Note – Most sites that use Solaris DHCP are *not* affected by this change. Your site is affected only if you plan to upgrade to Solaris 8, you previously created new DHCP options and modified the `/etc/dhcp/dhcptags` file, and you want to retain the changes. When you upgrade, the upgrade log notifies you that your `dhcptags` file had been modified and that you should make changes to the DHCP `inittab` file.

Differences Between `dhcptags` and `inittab`

The `inittab` file contains more information than the `dhcptags` file and it uses a different syntax.

A sample `dhcptags` entry is:

```
33 StaticRt - IPList Static_Routes
```

where `33` is the numeric code that is passed in the DHCP packet, `StaticRt` is the option name, `IPList` indicates the expected data is a list of IP addresses, and `Static_Routes` is a more descriptive name.

The `inittab` file consists of one-line records that describe each option. The format is similar to the format that defines symbols in `dhcptab`. The following table describes the syntax of the `inittab`.

TABLE 12–3 DHCP `inittab` File Syntax

Option	Description
option-name	Name of the option. The option name must be unique within its option category, and not overlap with other option names in the Standard, Site, and Vendor categories. For example, you cannot have two Site options with the same name, and you should not create a Site option with the same name as a Standard option.
category	Identifies the namespace in which the option belongs. Must be one of Standard, Site, Vendor, Field, or Internal.
code	Identifies the option when it is sent over the network. In most cases, the code uniquely identifies the option, without a category. However, in the case of internal categories like Field or Internal, a code might be used for other purposes and thus might not be globally unique. The code should be unique within the option's category, and not overlap with codes in the Standard and Site fields.
type	Describes the data associated with this option. Valid types are IP, Ascii, Octet, Boolean, Unumber8, Unumber16, Unumber32, Unumber64, Snumber8, Snumber16, Snumber32, and Snumber64. For numbers, an initial U or S indicates that the number is unsigned or signed, and the digits at the end indicate the amount of bits in the number. The type is not case sensitive.
granularity	Describes how many units of data make up a whole value for this option.
maximum	Describes how many whole values are allowed for this option. 0 indicates an infinite number.
consumers	Describes which programs can use this information. This should be set to `sdmi`, where: s – `snoop` d – `in.dhcpd` m – `dhcpmgr` i – `dhcpinfo`

A sample `inittab` entry is:

```
StaticRt Standard, 33, IP, 2, 0, sdmi
```

This entry describes an option named `StaticRt`, which is in the Standard category and is option code 33. The expected data is a potentially infinite number of pairs of IP addresses because the type is `IP`, granularity is `2`, and maximum is infinite (`0`). The consumers of this option are `sdmi`: `snoop`, `in.dhcpd`, `dhcpmgr`, and `dhcpinfo`.

Converting `dhcptags` Entries to `inittab` Entries

If you previously added entries to your `dhcptags` file, you must add corresponding entries to the new `inittab` file. The following example shows how a sample `dhcptags` entry might be expressed in `inittab` format.

Suppose you had added the following `dhcptags` entry for fax machines connected to the network:

```
128 FaxMchn - IP Fax_Machine
```

The code `128` means that it must be in the site category, the option name is `FaxMchn`, the data type is `IP`.

The corresponding `inittab` entry might be:

```
FaxMchn SITE, 128, IP, 1, 1, sdmi
```

The granularity of 1 and maximum of 1 indicate that one IP address is expected for this option.

CHAPTER 13

IPv6 Topics

Chapter 14	Provides overview information for IPv6
Chapter 15	Provides step-by-step instructions of IPv6 related tasks
Chapter 16	Provides Solaris IPv6 implementation information
Chapter 17	Provides IPv6 transition strategies and mechanism

CHAPTER 14

IPv6 (Overview)

The Internet Protocol, version 6 (IPv6), is a new version of Internet Protocol (IP) that is designed to be an evolutionary step from the current version, IPv4. IPv6 is a natural increment to IPv4. Deploying IPv6 with defined transition mechanisms does not disrupt current operations. IPv6 adds increased address space. IPv6 also improves Internet capability by using a simplified header format, support for authentication and privacy, autoconfiguration of address assignments, and new quality-of-service capabilities.

This chapter contains the following information:

IPv6 Features

Most of the changes from IPv4 to IPv6 are described in the following categories:

- **Expanded routing and addressing capabilities** – IPv6 increases the IP address size from 32 bits to 128 bits to support more levels of addressing hierarchy. IPv6 provides a much greater number of addressable nodes and employs simpler autoconfiguration of addresses.

The addition of a *scope* field improves the scalability of multicast routing to multicast addresses.

IPv6 defines a new type of address that is called an *anycast* address. An anycast address identifies sets of nodes. A packet that is sent to an anycast address is delivered to one of the nodes. The use of anycast addresses in the IPv6 source route allows nodes to control the path over which their traffic flows.

- **Header format simplification** – Some IPv4 header fields have been dropped or have been made optional. This change reduces the common-case processing cost of packet handling. This change also keeps the bandwidth cost of the IPv6 header as low as possible, despite the increased size of the addresses. Even though the IPv6 addresses are four times longer than the IPv4 addresses, the IPv6 header is only twice the size of the IPv4 header.
- **Improved support for options** – Changes in the way IP header options are encoded allow for more efficient forwarding. Also, the length of options has less stringent limits. The changes also provide greater flexibility for introducing new options in the future.
- **Quality-of-service capabilities** – A new capability is added to enable the labeling of packets that belong to particular traffic *flows* for which the sender requests special handling. For example, the sender can request nondefault quality of service or *real-time* service.
- **Authentication and privacy capabilities** – IPv6 includes the definition of extensions that provide support for authentication, data integrity, and confidentiality.

IPv6 Header and Extensions

The IPv6 protocol defines a set of headers, including the basic IPv6 header and the IPv6 extension headers.

Header Format

The following figure shows the elements that appear in the IPv6 header and the order in which the elements appear.

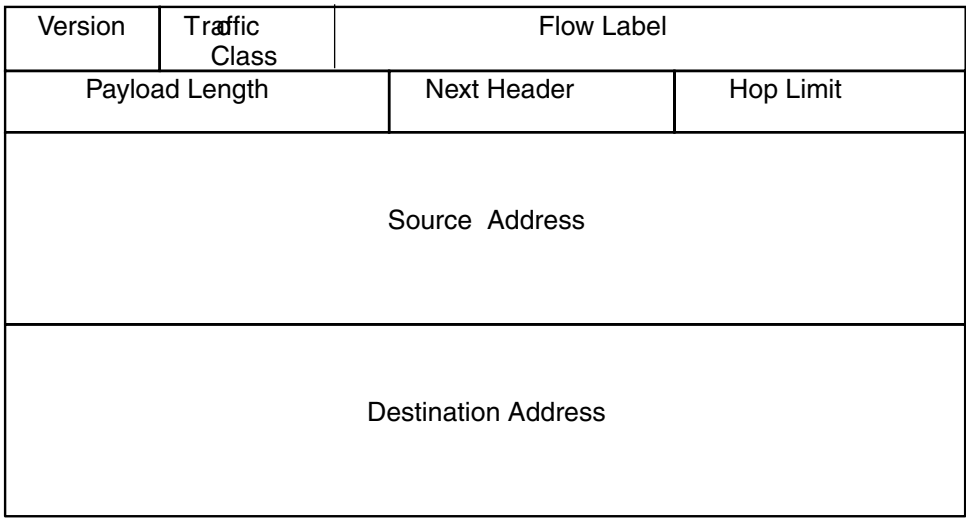


FIGURE 14–1 IPv6 Header Format

The following list describes the function of each header field.

- **Version** – 4-bit Version number of Internet Protocol = 6.
- **Traffic Class** – 8-bit traffic class field (see "Traffic Class" on page 300).
- **Flow Label** – 20-bit field (see "IPv6 Quality-of-Service Capabilities" on page 298).
- **Payload Length** – 16-bit unsigned integer, which is the rest of the packet that follows the IPv6 header, in octets.
- **Next Header** – 8-bit selector. Identifies the type of header immediately following the IPv6 header. Uses the same values as the IPv4 protocol field (see "Extension Headers" on page 281).
- **Hop Limit** – 8-bit unsigned integer. Decremented by one by each node that forwards the packet. The packet is discarded if Hop Limit is decremented to zero.
- **Source Address** – 128 bits. The address of the initial sender of the packet (see "IPv6 Addressing" on page 282).
- **Destination Address** – 128 bits. The address of the intended recipient of the packet. The intended recipient is not necessarily the recipient if an optional Routing Header is present.

Extension Headers

IPv6 includes an improved option mechanism over IPv4. IPv6 options are placed in separate extension headers that are located between the IPv6 header and the

transport-layer header in a packet. Most IPv6 extension headers are not examined or processed by any router along a packet's delivery path until the packet arrives at its final destination. This feature is a major improvement in router performance for packets that contain options. In IPv4, the presence of any options requires the router to examine all options.

The other improvement is that, unlike IPv4 options, IPv6 extension headers can be of arbitrary length. Also, the number of options that a packet carries are not limited to 40 bytes. This feature, plus the manner in which IPv6 options are processed, permits IPv6 options to be used for functions that are not practical in IPv4. A good example of IPv6 options is the IPv6 authentication and security encapsulation options.

To improve performance when handling subsequent option headers, and the transport protocol that follows, IPv6 options are always an integer multiple of eight octets long. The integer multiple of eight octets retains the alignment of subsequent headers.

The following IPv6 extension headers are currently defined.

- **Routing** – Extended routing (like IPv4 loose source route)
- **Fragmentation** – Fragmentation and reassembly
- **Authentication** – Integrity and authentication, security
- **Encapsulation** – Confidentiality
- **Hop-by-Hop Option** – Special options that require hop-by-hop processing
- **Destination Options** – Optional information to be examined by the destination node

IPv6 Addressing

IPv6 addresses are 128 bits long and are identifiers for individual interfaces and sets of interfaces. IPv6 addresses of all types are assigned to interfaces, not nodes (hosts and routers). Because each interface belongs to a single node, any of the unicast addresses of that node's interfaces' can be used as an identifier for the node. A single interface can be assigned multiple IPv6 addresses of any type.

IPv6 addresses consist of the following types: unicast, anycast, and multicast.

- Unicast addresses identify a single interface.
- Anycast addresses identify a set of interfaces. A packet that is sent to an anycast address is delivered to a member of the set.
- Multicast addresses identify a group of interfaces. A packet that is sent to a multicast address is delivered to all of the interfaces in the group.

In IPv6, multicast addresses replace broadcast addresses.

IPv6 supports addresses that are four times the number of bits as IPv4 addresses (128 in contrast to 32). Thus, the number of potential addresses is four billion times four billion times the size of the IPv4 address space. Realistically, the assignment and routing of addresses requires the creation of hierarchies that reduce the efficiency of address space usage. Consequently, the reduction in efficiency reduces the number of available addresses. Nonetheless, IPv6 provides enough address space for the foreseeable future.

The leading bits in the address specify the type of IPv6 address. The variable-length field that contains these leading bits is called the format prefix (FP). The following table shows the initial allocation of these prefixes.

TABLE 14–1 Format Prefix Allocations

Allocation	Prefix (Binary)	Fraction of Address Space
Reserved	0000 0000	1/256
Unassigned	0000 0001	1/256
Reserved for NSAP Allocation	0000 001	1/128
Reserved for IPX Allocation	0000 010	1/128
Unassigned	0000 011	1/128
Unassigned	0000 1	1/32
Unassigned	0001	1/16
Aggregate Global Unicast Address	001	1/8
Unassigned	010	1/8
Unassigned	011	1/8
Reserved for Neutral-Interconnect-Based Unicast Addresses	100	1/8
Unassigned	101	1/8
Unassigned	110	1/8
Unassigned	1110	1/16
Unassigned	1111 0	1/32
Unassigned	1111 10	1/64
Unassigned	1111 110	1/128
Unassigned	1111 1110 0	1/512

TABLE 14–1 Format Prefix Allocations *(Continued)*

Allocation	Prefix (Binary)	Fraction of Address Space
Link Local Use Addresses	1111 1110 10	1/1024
Site Local Use Addresses	1111 1110 11	1/1024
Multicast Addresses	1111 1111	1/256

The allocations support the direct allocation of aggregate global unicast addresses, local-use addresses, and multicast addresses. Space is reserved for Network Service Access Point (NSAP) addresses, Internetwork Packet Exchange Protocol (IPX) addresses, and neutral-interconnect addresses. The remainder of the address space is unassigned for future use. This remaining address space can be used for expansion of existing use (for example, additional aggregate global unicast addresses) or new uses (for example, separate locators and identifiers). Notice that anycast addresses are not shown here because anycast addresses are allocated out of the unicast address space.

Approximately 15 percent of the address space is initially allocated. The remaining 85 percent is reserved for future use.

Unicast Addresses

IPv6 unicast address assignment consists of the following forms:

- Aggregate global unicast address
- Neutral-interconnect unicast address
- NSAP address
- IPX hierarchical address
- Site-local-use address
- Link-local-use address
- IPv4-capable host address

Additional address types can be defined in the future.

Aggregate Global Unicast Addresses

Aggregate global unicast addresses are used for global communication. These addresses are similar in function to IPv4 addresses under classless interdomain routing (CIDR). The following table shows their format.

TABLE 14–2 Aggregate Global Unicast Addresses Format

3 bits	13 bits	8 bits	24 bits	16 bits	64 bits

TABLE 14–2 Aggregate Global Unicast Addresses Format *(Continued)*

FP	TLA ID	RES	NLA ID	SLA ID	Interface ID

FP	Format Prefix (001)
TLA ID	Top-Level Aggregation identifier
RES	Reserved for future use
NLA ID	Next-Level Aggregation identifier
SLA ID	Site-Level Aggregation identifier
INTERFACE ID	Interface identifier

The first 48 bits represent the public topology. The next 16 bits represent the site topology.

The first 3 bits identify the address as an aggregate global unicast address. The next field, TLA ID, is the top level in the routing hierarchy. The next 8 bits are reserved for future use. The NLA ID field is used by organizations that are assigned a TLA ID to create an addressing hierarchy and to identify sites.

The SLA ID field is used by an individual organization to create its own local addressing hierarchy and to identify subnets. Use of the SLA ID field is analogous to subnets in IPv4 except that each organization has a much greater number of subnets. The 16–bit SLA ID field supports 65,535 individual subnets. The Interface ID is used to identify interfaces on a link. The Interface ID must be unique on that link. The Interface ID can also be unique over a broader scope. In many instances, an interface identifier is the same or is based on the interface's link-layer address.

Local-Use Addresses

A local-use address is a unicast address that has only local routability scope. A local-use address can only be used within the subnet or within a subscriber network. These addresses are intended for use inside of a site for *plug and play* local communication and for bootstrap operations for the use of global addresses.

The two types of local-use unicast addresses are link-local and site-local. The Link-Local-Use is for use on a single link. The Site-Local-Use is for use on a single site. The following table shows the Link-Local-Use address format.

TABLE 14–3 Link-Local-Use Addresses Format

10 bits	54 bits	64 bits
1111111010	0	Interface ID

Link-Local-Use addresses are used for addressing on a single link for purposes such as auto-address configuration.

The following table shows the Site-Local-Use address format.

TABLE 14–4 Site-Local-Use Addresses

10 bits	38 bits	16 bits	64 bits
1111111011	0	Subnet ID	Interface ID

For both types of local-use addresses, the Interface ID is an identifier that must be unique in its domain. In most instances, the identifier uses a node's IEEE-802 48–bit address. The Subnet ID identifies a specific subnet in a site. The Subnet ID and the interface ID form a local-use address. Consequently, a large private internet can be constructed without any other address allocation.

Organizations that are not yet connected to the global Internet can use local-use addresses. Local-use addresses enable organizations to operate without the need to request an address prefix from the global Internet address space. If the organization later connects to the Internet, the Subnet ID, Interface ID, and a global prefix can be used to create a global address. For example, the organization can use the Registry ID, Provider ID, and the Subscriber ID to create a global address. This enhancement is a significant improvement over IPv4. IPv4 requires sites that use private (non-global) IPv4 addresses to manually renumber when sites connect to the Internet. IPv6 automatically does the renumbering.

IPv6 Addresses With Embedded IPv4 Addresses

The IPv6 transition mechanisms include a technique for hosts and routers to tunnel IPv6 packets dynamically under IPv4 routing infrastructure. IPv6 nodes that utilize this technique are assigned special IPv6 unicast addresses that carry an IPv4 address in the low-order 32 bits. This type of address is called an *IPv4-compatible IPv6 address*. The address format is shown in the following table.

TABLE 14–5 IPv4–Compatible IPv6 Address Format

80 bits	16 bits	32 bits
0000.....0000	0000	IPv4 Address

A second type of IPv6 address that holds an embedded IPv4 address is also defined. This address is used to represent an IPv4 address within the IPv6 address space. This address is mainly used internally within the implementation of applications, APIs, and

the operating system. This type of address is called an *IPv4-mapped IPv6 address*. The address format is shown in the following table.

TABLE 14–6 IPv4–Mapped IPv6 Address Format

80 bits	16 bits	32 bits
0000.....0000	FFFF	IPv4 Address

Anycast Addresses

An IPv6 anycast address is an address that is assigned to more than one interface. Typically, the address belongs to different nodes. A packet that is sent to an anycast address is routed to the *nearest* interface that has that address, according to the routing protocol's measure of distance.

Anycast addresses can be used as part of a route sequence. Thus, a node can select which of several Internet service providers that the node wants to carry its traffic. This capability is sometimes called *source-selected policies*. You implement this capability by configuring anycast addresses to identify the set of routers that belongs to Internet service providers. For example, you can configure one anycast address per Internet service provider. You can use the anycast addresses as intermediate addresses in an IPv6 routing header. Then the packet is delivered by a particular provider or is delivered by a sequence of providers. You can also use anycast addresses to identify the set of routers that are attached to a particular subnet. You can also use anycast addresses to identify the set of routers that provide entry into a particular routing domain.

You can locate anycast addresses from the unicast address space by using any of the defined unicast address formats. Thus, anycast addresses are syntactically indistinguishable from unicast addresses. When you assign a unicast address to more than one interface, you turn the unicast address into an anycast address. However, you must explicitly configure the nodes to which the address is assigned in order to know that the address is an anycast address.

Multicast Addresses

An IPv6 multicast address is an identifier for a group of interfaces. An interface can belong to any number of multicast groups. The following table shows the multicast address format.

TABLE 14–7 Multicast Address Format

8 bits	4 bits	4 bits	112 bits

TABLE 14–7 Multicast Address Format *(Continued)*

11111111	FLGS	SCOP	Group ID

11111111 at the start of the address identifies the address as a multicast address. FLGS is a set of four flags: 0,0,0,T.

The high-order 3 flags are reserved and must be initialized to zero.

- **T=0** – Indicates a permanently assigned (*well-known*) multicast address, assigned by the global Internet numbering authority.
- **T=1** – Indicates a non-permanently assigned (*transient*) multicast address.

SCOP is a 4-bit multicast scope value used to limit the scope of the multicast group. The following table shows the SCOP values.

TABLE 14–8 SCOP Values

0	Reserved	8	Organization-local scope
1	Node-local scope	9	(unassigned)
2	Link-local scope	A	(unassigned)
3	(unassigned)	B	(unassigned)
4	(unassigned)	C	(unassigned)
5	Site-local scope	D	(unassigned)
6	(unassigned)	E	Global scope
7	(unassigned)	F	Reserved

Group ID identifies the multicast group, either permanent or transient, within the given scope.

IPv6 Routing

Routing in IPv6 is almost identical to IPv4 routing under CIDR. The only difference is 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, 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 (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 capability 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.

In order 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 features. The new features include provider selection and extended addresses.

IPv6 Neighbor Discovery

IPv6 solves a set of problems that are related to the interaction between nodes that are attached to the same link. IPv6 defines mechanisms for solving each of the following problems.

- **Router discovery** – Hosts locate routers that reside on an attached link.
- **Prefix discovery** – Hosts discover the set of address prefixes that define which destinations are attached to the link (sometimes referred to as on-link). Nodes use prefixes to distinguish destinations that reside on a link from those only reachable through a router.
- **Parameter discovery** – A node learns link parameters, such as the link maximum transmission unit (MTU). A node also learns Internet parameters, such as the hop limit value, to place in outgoing packets.
- **Address autoconfiguration** – Nodes automatically configure an address for an interface.
- **Address resolution** – Nodes determine the link-layer address of a neighbor (an on-link destination) with only the destinations's IP address.
- **Next-hop determination** – An algorithm determines mapping for an IP destination address into the IP address of the neighbor to which traffic for the destination should be sent. The next-hop can be a router or the destination.

- **Neighbor unreachability detection** – Nodes determine that a neighbor is no longer reachable. For neighbors that are used as routers, alternate default routers can be tried. For both routers and hosts, address resolution can be performed again.
- **Duplicate address detection** – A node determines that an address that the node wants to use is not already in use by another node.
- **Redirect** – A router informs a host of a better first-hop node to reach a particular destination.

Neighbor discovery defines five different Internet Control Message Protocol (ICMP) packet types: a pair of router solicitation and router advertisement messages, a pair of neighbor solicitation and neighbor advertisements messages, and a redirect message. The messages serve the following purpose:

- **Router solicitation** – When an interface becomes enabled, hosts can send router solicitations. 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** – Sent by a node to determine the link-layer address of a neighbor. Also, sent by a node 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 response to a Neighbor Solicitation message, node can also send unsolicited neighbor advertisements to announce a link-layer address change.
- **Redirect** – Used by routers to inform hosts of a better first hop for a destination, or that the destination is on-link.

Router Advertisement

On multicast-capable links and point-to-point links, each router periodically multicasts 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 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 prefixes that is used for on-link determination. 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. The list 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 that informs the sender that the destination is a neighbor.

Router advertisements (and per-prefix flags) allow routers to inform hosts how to perform address autoconfiguration. For example, routers can specify whether hosts should use stateful (DHCPv6) or autonomous (stateless) address configuration.

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 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 multicasting a neighbor solicitation that asks the target node to return its link-layer address. Neighbor solicitation messages are multicast 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.

Neighbor Solicitation and Unreachability

Neighbor solicitation messages can also be used to determine if more than one node has been 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 and being processed properly by its IP layer. Neighbor unreachability detection uses confirmation from two sources. When possible, upper-layer protocols provide a positive confirmation that a connection is making *forward progress*. That is, data that was sent previously is known to have been delivered correctly. For example, new TCP acknowledgments were received recently. When positive confirmation is not

forthcoming through such hints, a 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.

In addition to addressing the previous general problems, neighbor discovery also handles the following situations.

- **Link-layer address change** – A node that knows its link-layer address has been changed can multicast a few (unsolicited) neighbor advertisement packets. The node can multicast to all nodes to update cached link-layer 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.
- **Inbound load balancing** – Nodes with replicated interfaces might want to load-balance the reception of incoming packets across multiple network interfaces on the same link. Such nodes have multiple link-layer 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-layer addresses.

  Load balancing is handled by allowing routers to omit the source link-layer address from router advertisement packets. Consequently, neighbors must use neighbor solicitation messages to learn link-layer addresses of routers. Returned neighbor advertisement messages can then contain link-layer addresses that differ, depending on who issued the solicitation.
- **Anycast addresses** – Anycast addresses identify one of a set of nodes that provide an equivalent service. Multiple nodes on the same link can be configured to recognize the same anycast address. Neighbor discovery handles anycasts by setting nodes to expect to receive multiple neighbor advertisements for the same target. All advertisements for anycast addresses are tagged as being non-override advertisements. Non-override advertisements invoke specific rules to determine which of potentially multiple advertisements should be used.
- **Proxy advertisements** – A router that accept packets on behalf of a target address that is unable to respond to neighbor solicitations can issue non-override neighbor advertisements. Currently, the use of proxy is not specified. However, proxy advertising can potentially be used to handle cases like mobile nodes that have moved off-link. However, the use of proxy is not intended as a general mechanism to handle nodes that do not implement this protocol.

Comparison With IPv4

The IPv6 neighbor discovery protocol corresponds to a combination of the IPv4 protocols Address Resolution Protocol (ARP), 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 neighbor discovery protocol provides a multitude of improvements over the IPv4 set of protocols.

- Router discovery is part of the base protocol set. Hosts do not need to *snoop* the routing protocols.
- Router advertisements carry link-layer addresses. No additional packet exchange is needed to resolve the router's link-layer address.
- Router advertisements carry prefixes for a link. A separate mechanism is not needed to configure the *netmask*.
- Router advertisements enable address autoconfiguration.
- Routers can 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.
- 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.
- Redirects contain the link-layer address of the new first hop. Separate address resolution is not needed on receiving a redirect.
- Multiple prefixes can be associated with the same link. By default, hosts learn all on-link 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 off-link. Consequently, hosts send the traffic to routers. A router can then issue redirects as appropriate.
- Unlike IPv4, the recipient of an IPv6 redirect assumes that the new next-hop is on-link. In IPv4, a host ignores redirects that specify a next-hop that is not on-link, according to the link's network mask. The IPv6 redirect mechanism is analogous to the XRedirect facility. The redirect mechanism is useful on non-broadcast and shared media links. On these links, it is undesirable or not possible for nodes to check for all prefixes for on-link destinations.
- 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-layer addresses. For instance, mobile nodes can move off-link without losing any connectivity because of stale ARP caches.
- Unlike ARP, neighbor discovery detects half-link failures (using neighbor unreachability detection) and avoids sending traffic to neighbors with which two-way connectivity is absent.
- Unlike in IPv4 router discovery, the router advertisement messages do not contain a preference field. The preference field is not needed to handle routers of different stability. The neighbor unreachability detection detects dead routers and switches

to a working router.

- By using link-local addresses to uniquely identify routers, hosts can maintain the router associations. The ability to identify routers is required for router advertisements and is required for redirect messages. Hosts need to maintain router associations if the site uses new global prefixes.
- 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 Internet Control Message Protocol (ICMP) redirects and can send router advertisement messages.
- By placing address resolution at the ICMP layer, the protocol becomes more media independent than ARP. Consequently, standard IP authentication and security mechanisms can be used.

IPv6 Stateless Address Autoconfiguration

A host performs several steps to autoconfigure its interfaces in IPv6. The autoconfiguration process creates a link-local address. The autoconfiguration process verifies its uniqueness on a link. The process also determines which information should be autoconfigured (addresses, other information, or both). The process determines if the addresses should be obtained through the stateless mechanism, the stateful mechanism, or both mechanisms. This section describes the process for generating a link-local address. This section also describes the process for generating site-local and global addresses by stateless address autoconfiguration. Finally, this section describes the procedure for duplicate address detection.

Stateless Autoconfiguration Requirements

IPv6 defines mechanisms for both stateful address and stateless address autoconfiguration. Stateless autoconfiguration requires no manual configuration of hosts, minimal (if any) configuration of routers, and no additional servers. The stateless mechanism allows a host to generate its own addresses by using a combination of local information and non-local information that is advertised by routers. Routers advertise prefixes that identify the subnet or subnets that are associated with a link. Hosts generate an *interface identifier* that uniquely identifies an interface on a subnet. An address is formed by combining the the prefix and the interface identifier. In the absence of routers, a host can generate only link-local addresses. However, link-local addresses are only sufficient for allowing communication among nodes that are attached to the same link.

Stateful Autoconfiguration Model

In the stateful autoconfiguration model, hosts obtain interface addresses or configuration information and parameters from a server. Servers maintain a database that checks which addresses have been assigned to which hosts. The stateful autoconfiguration protocol allows hosts to obtain addresses and other configuration information from a server. Stateless and stateful autoconfiguration complement each other. For example, a host can use stateless autoconfiguration to configure its own addresses, but use stateful autoconfiguration to obtain other information.

When to Use Stateless and Stateful Approaches

The stateless approach is used when a site is not concerned with the exact addresses that hosts use. However, the addresses must be unique and must be properly routable. The stateful approach is used when a site requires more precise control over exact address assignments. Stateful and stateless address autoconfiguration can be used simultaneously. The site administrator specifies which type of autoconfiguration to use through the setting of appropriate fields in router advertisement messages.

IPv6 addresses are leased to an interface for a fixed (possibly infinite) length of time. Each address has an associated lifetime that indicates how long the address is bound to an interface. When a lifetime expires, the binding (and address) become invalid and the address can be reassigned to another interface elsewhere. To handle the expiration of address bindings gracefully, an address experiences two distinct phases while the address is assigned to an interface. Initially, an address is preferred, meaning that its use in arbitrary communication is unrestricted. Later, an address becomes *deprecated* in anticipation that its current interface binding becomes invalid. When the address is in a deprecated state, the use of the address is discouraged, but not strictly forbidden. New communication (for example, the opening of a new TCP connection) should use a preferred address when possible. A deprecated address should be used only by applications that have been using the address. Applications that cannot switch to another address without a service disruption can use a deprecated address.

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 specified in this document 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 probably generate link-local addresses by using the mechanism that is described in

this document. In addition, routers are expected to pass successfully the duplicate address detection procedure on all addresses prior to assigning the address to an interface.

IPv6 Protocol Overview

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. Autoconfiguration begins when a multicast-capable interface is enabled, for example, during system startup. Nodes (both hosts and routers) begin the autoconfiguration process by generating a link-local address for the interface. A link-local address is formed by appending the interface's identifier to the well-known link-local prefix.

A node must attempt to verify that a tentative link-local address is not already in use by another node on the link. After verification, the link-local address can be assigned to an interface. Specifically, the node sends a neighbor solicitation message that contains the tentative address as the target. If another node is already using that address, the node returns a neighbor advertisement saying that the node is using that address. 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. These parameters can be set by system management.

If a node determines that its tentative link-local address is not unique, autoconfiguration stops and manual configuration of the interface is required. To simplify recovery in this instance, an administrator can supply an alternate interface identifier that overrides the default identifier. Then the autoconfiguration mechanism can be applied by using the new (presumably unique) interface identifier. Alternatively, link-local and other addresses need to be configured manually.

After a node determines that its tentative 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 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. If no routers are present, stateful autoconfiguration is invoked.

Routers send router advertisements periodically. However, the delay between successive advertisements is generally longer than a host that performs

autoconfiguration can wait. To obtain an advertisement quickly, a host sends one or more router solicitations to the all-routers multicast group. Router advertisements contain two flags that indicate what type of stateful autoconfiguration (if any) should be performed. A *managed address configuration* flag indicates whether hosts should use stateful autoconfiguration to obtain addresses. An *other stateful configuration* flag indicates whether hosts should use stateful autoconfiguration to obtain additional information (excluding addresses).

Prefix Information

Router advertisements also contain zero or more prefix information options that contain information that stateless address autoconfiguration uses to generate site-local and global addresses. The stateless address and stateful address autoconfiguration fields in router advertisements are processed independently of one another. A host can use both stateful address and stateless address autoconfiguration simultaneously. One option field that contains prefix information, the *autonomous address-configuration* 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 how long addresses that are created from the prefix remain preferred and valid.

Because routers generate router advertisements periodically, hosts continually receive new advertisements. Hosts process the information that is contained in each advertisement as described previously. Hosts add to the information. Hosts also refresh the information that is received in previous advertisements.

Address Uniqueness

For safety, 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 identifier. Thus, if a node has already verified the uniqueness of a link-local address, additional addresses that are created from the same interface identifier need not be tested individually. In contrast, all addresses that are obtained manually or by stateful address autoconfiguration should be tested individually for uniqueness. 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.

IPv6 Mobility Support

Routing is based on the subnet prefix in a packet's destination IP address. Consequently, packets that are destined for a mobile node, host or router, do not reach the node when the node is not attached to the node's home link. The home link is the link where the node's home IPv6 subnet prefix exists. In order to continue communication, regardless of a node's movement, a mobile node can change its IP address each time the node moves to a new link. However, the mobile node does not maintain transport and higher-layer connections when the node changes location. Consequently, IPv6 mobility support is particularly important when recognizing that mobile computers become a significant population of the Internet in the future.

IPv6 mobility support solves this problem. IPv6 mobility enables a mobile node to move from one link to another without changing the mobile node's IP address. IPv6 mobility assigns an IP address to the mobile node within its home subnet prefix on its home link. This address is known as the node's *home address*.

Thus, packets that are routed to the mobile node's *home address* reach their destination regardless of the mobile node's current point of attachment to the Internet. The mobile node can continue to communicate with other nodes (stationary or mobile) after moving to a new link.

IPv6 mobility solves the problem of transparently routing packets to and from mobile nodes while away from home. IPv6 mobility does not solve all the problems that are related to the use of mobile computers or wireless networks. In particular, IPv6 mobility does not attempt to solve the following problems:

- The ability to handle links with partial reachability, such as typical wireless networks. However, a movement detection procedure addresses some aspects.
- Access control on a link that is being visited by a mobile node.

IPv6 Quality-of-Service Capabilities

A host can use the flow label and the traffic fields in the IPv6 header. A host uses these fields to identify those packets for which the host requests special handling by IPv6 routers. For example, the host can request non-default quality of service or real-time service. This important capability enables the support of applications that require some degree of consistent throughput, delay, or jitter. These types of applications are known as *multi media* or *real-time* applications.

Flow Labels

A source can use the 20-bit flow label field in the IPv6 header. A source can use this field to label those packets for which the source requests special handling by the IPv6 routers. For example, a source can request non-default quality of service or real-time service. This aspect of IPv6 is still experimental and subject to change as the requirements for flow support in the Internet become clearer. Some hosts or routers do not support the functions of the flow label field. These hosts or routers are required to set the field to zero when originating a packet. Hosts or routers forward the field without changes when forwarding a packet. Hosts or routers ignore the field when receiving a packet.

What Is a Flow?

A flow is a sequence of packets that are sent from a particular source to a particular (unicast or multicast) destination. The source also requires special handling by the intervening routers. The nature of the special handling might be conveyed to the routers by a control protocol. The control protocol can be a resource reservation protocol. The special handling also might be conveyed by information within the flow's packets, for example, in a hop-by-hop option.

Active flows from a source to a destination can be multiple and can contain traffic that is not associated with any flow. The combination of a source address and a nonzero flow label uniquely identifies a flow. Packets that do not belong to a flow carry a flow label of zero.

The flow's source node assigns a flow label to a flow. New flow labels must be chosen randomly (in a "pseudo" manner) and uniformly from the range 1 to FFFFF hex. This random allocation makes any set of bits within the flow label field suitable for use as a hash key by routers. The routers can use the hash key to look up the state that is associated with the flow.

Packets Belonging to the Same Flow

All packets that belong to the same flow must be sent with the same source address, same destination address, and same nonzero flow label. If any of those packets include a hop-by-hop options header, then the packets must all be originated with the contents of the hop-by-hop options header. The next header field of the hop-by-hop options header is excluded. If any of those packets include a routing header, then the packets must all be originated with the same contents in all extension headers. The same contents include all extensions before the routing header and the routing header. The next header field in the routing header is excluded. The routers or destinations are permitted, but not required, to verify that these conditions are satisfied. If a violation is detected, the violation should be reported to the source. The violation is reported by

a problem message for an ICMP parameter, Code 0. The violation points to the high-order octet of the flow label field. The high-order octet is offset one octet within the IPv6 packet.

Routers are free to set up the flow-handling state for any flow. Routers do not need explicit flow establishment information from a control protocol, a hop-by-hop option, or other means. For example, when a router receives a packet from a particular source with an unknown, non-zero flow label, a router can process its IPv6 header. The router processes any necessary extension headers in the same way that the router processes extension headers with the flow label field set to zero. The routers also determine the next-hop interface. The routers might also update a hop-by-hop option, advance the pointer and addresses in a routing header, or decide how to queue the packet. The decision to queue the packet is based on the Traffic Class field of the packet. The routers can then choose to *remember* the results of those processing steps. Then the routers can cache the information. The routers use the source address and the flow label as the cache key. Subsequent packets, with the same source address and flow label, can then be handled by referring to the cached information. The routers do not need to examine all those fields. The routers can assume that the fields are unchanged from the first packet that is checked in the flow.

Traffic Class

The nodes that originate a packet must identify different classes or different priorities of IPv6 packets. The nodes use the Traffic Class field in the IPv6 header to make this identification. The routers that forward the packets also use the Traffic Class field for the same purpose.

The following general requirements apply to the Traffic Class field:

- The service interface to the IPv6 service within a node must supply the value of the Traffic Class bits for an upper-layer protocol. The Traffic Class bits must be in packets that are originated by that upper-layer protocol. The default value must be zero for all 8 bits.
- Nodes that support some or all of the Traffic Class bits can change the value of those bits. The nodes can change only the values in packets that the nodes originate, forward, or receive, as required for that specific use. Nodes should ignore and leave unchanged any bits of the Traffic Class field for which the nodes do not support a specific use.
- The Traffic Class bits in a received packet might not be the same value that is sent by the packet's source. Therefore, the upper-layer protocol must not assume that the values are the same.

IPv6 Security Improvements

The current Internet has a number of security problems. The Internet lacks effective privacy and effective authentication mechanisms below the application layer. IPv6 remedies these shortcomings by having two integrated options that provide security services. You can use these two options either individually or together to provide differing levels of security to different users. Different user communities have different security needs.

The first option, an extension header called the IPv6 *Authentication Header* (AH), provides authentication and integrity (without confidentiality) to IPv6 datagrams. The extension is algorithm independent. The extension supports many different authentication techniques. The use of AH is proposed to help ensure interoperability within the worldwide Internet. The use of AH eliminates a significant class of network attacks, including host masquerading attacks. When using source routing with IPv6, the IPv6 authentication header becomes important because of the known risks in IP source routing. Upper-layer protocols and upper-layer services currently lack meaningful protections. However, the placement of the header at the Internet layer helps provide host origin authentication.

The second option, an extension header that is called the IPv6 *Encapsulating Security Payload* (ESP), provides integrity and confidentiality to IPv6 datagrams. Though simpler than some similar security protocols (for example, SP3D, ISO NLSP), ESP remains flexible and is algorithm independent.

IPv6 Authentication Header and IPv6 Encapsulating Security Payload are features of the new Internet Protocol Security (IPsec). For an overview of IPsec, see Chapter 19. For a description of how you implement IPsec, see Chapter 20.

CHAPTER 15

Administering IPv6 (Task)

This chapter shows you how to enable IPv6 and IPv6 routers. This chapter also shows you how to configure IPv6 addresses for DNS, NIS, and NIS+. You also learn how to create tunnels between routers. This chapter also shows you how to run IPv6 additions to commands that display diagnostics. Finally, this chapter shows you how to display IPv6 name service information.

This chapter contains the following information:

Topic	Information
Overview information about IPv6	Chapter 14
Transition information about transitioning from IPv4 to IPv6	Chapter 17
Conceptual information that is related to the procedures in this chapter	Chapter 16

Enabling IPv6 Nodes

This section provides procedures that you might need to configure IPv6 nodes on your network.

Note – The term *node* in this context refers either to a Solaris server or client workstation.

Enabling IPv6 Nodes Task Map

TABLE 15–1 Enabling IPv6 Nodes Task Map

Task	Description	For Instructions, Go to ...
Enable IPv6 on a node	Involves touching `hostname6.`*interface* file, displaying addresses, and entering the addresses in the `/etc/inet/ipnodes` file. (See note.)	"How to Enable IPv6 on a Node" on page 304
Configure a Solaris IPv6 router	Involves adding entries to the `indp.conf` file.	"How to Configure a Solaris IPv6 Router" on page 305
Add IPv6 addresses to NIS and NIS+	Involves adding entries to the `/etc/ipnodes` file.	"How to Add IPv6 Addresses to NIS and NIS+" on page 306
Add IPv6 addresses to DNS	Involves adding AAAA records to the DNS zone and reverse zone file.	"How to Add IPv6 Addresses to DNS" on page 307

Note – You can enable IPv6 on a system when you install the Solaris software. If you answered *yes* to enable IPv6 during the installation process, you can omit the following procedures to enable IPv6.

▼ How to Enable IPv6 on a Node

1. **Become superuser on the system where you want to enable IPv6.**
2. **On a command line, type the following for each interface.**

   # **`touch /etc/hostname6.`***interface*

Interface	Interface name, such as `le0`, `le1`.

3. **Reboot.**

Note – The reboot process sends out router discovery packets. The router responds with a prefix. The response enables the node to configure the interfaces with an IP address. Rebooting also restarts key network daemons in IPv6 mode.

---

4. **On a command line, display the IPv6 addresses.**

```
# ifconfig -a
```

5. **Add the IPv6 address to the appropriate name service as follows:**
   - For NIS and NIS+, see "How to Add IPv6 Addresses to NIS and NIS+" on page 306.
   - For DNS, see "How to Add IPv6 Addresses to DNS" on page 307.

▼ How to Configure a Solaris IPv6 Router

1. **Become superuser on the system that acts as a router.**
2. **Edit the file `/etc/inet/ndpd.conf` with subnet prefixes by adding one or more of the following entries.**

   See the `in.ndpd`(1M) man page for a list of variables and allowable values. For more information about the `ndpd.conf` file, see the `ndpd.conf`(4) man page.

   a. **Add entries that specify router behavior for all interfaces.**

      `ifdefault` *variable value*

   b. **Add entries that specify the default behavior of prefix advertisement.**

      `prefixdefault` *variable value*

   c. **Add sets per interface parameter entries.**

      `if` *interface variable value*

   d. **Add advertisements for each entry for interface prefix information.**

      `prefix` *prefix/length interface variable value*

3. **Reboot the system.**

Note – Neighbor discovery (`in.ndpd`) relays the subnet address prefixes of the hosts to the hosts. Also, the next generation RIP routing protocol (`in.ripngd`) runs automatically.

Example—`ndpd.conf` Router Configuration File

```
# Send router advertisements out all NICs
ifdefault AdvSendAdvertisements on
# Advertise a global prefix and a
# site local prefix on three interfaces.
# 0x9255 = 146.85
prefix 2:0:0:9255::0/64         hme0
prefix fec0:0:0:9255::0/64     hme0
# 0x9256 = 146.86
prefix 2:0:0:9256::0/64     hme1
prefix fec0:0:0:9256::0/64    hme1
# 0x9259 = 146.89
prefix 2:0:0:9259::0/64        hme2
prefix fec0:0:0:9259::0/64    hme2
```

▼ How to Add IPv6 Addresses to NIS and NIS+

A new table has been added for NIS+ named `ipnodes.org_dir`. The table contains both IPv4 and IPv6 addresses for a host. The existing `hosts.org_dir` table, which contains only IPv4 addresses for a host, remains the same to facilitate existing applications. Both the `hosts.org_dir` and `ipnodes.org_dir` tables must be consistent with the IPv4 addresses. See "IPv6 Extensions to Solaris Name Services" on page 336 for an overview.

Administration of the new `ipnodes.org_dir` table is similar to administering the `hosts.org_dir`. The same tools and utilities that are used in administering the previous NIS+ tables are valid for `ipnodes.org_dir`. See *System Administration Guide: Naming and Directory Services* for details on how to manipulate the NIS+ table.

The following procedure merges the entries from `/etc/inet/ipnodes` into the `ipnodes.org_dir` table (in verbose mode). The NIS+ table was probably created by `nistbladm`(1), `nissetup`(1M), or `nisserver`(1M).

- **On a command line, type the following command:**

```
% nisaddent -mv -f /etc/inet/ipnodes ipnodes
```

Use the following procedure to display the `ipnodes.org_dir` table.

- **On a command line, type the following command:**

```
% nisaddent -d ipnodes
```

Two new maps have been added for NIS: `ipnodes.byname` and `ipnodes.byaddr`. These maps contain both IPv4 and IPv6 host name and address associations. The `hosts.byname` and `hosts.byaddr` maps, which contain only IPv4 host name and address associations, remain the same to facilitate existing applications. Administration of the new maps is similar to the maintenance of the `hosts.byname` and `hosts.byaddr` older maps. Again, it is important that when you update the *hosts* maps with IPv4 addresses that the new ipnode maps are also updated with the same information.

Note – Tools that are aware of IPv6 use the new NIS maps and the new NIS+ tables.

▼ How to Add IPv6 Addresses to DNS

1. **Become superuser on system that has DNS.**
2. **Edit the appropriate DNS zone file by adding AAAA records for the IPv6–enabled host, using the following format.**

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

3. **Edit the DNS reverse zone file and add PTR records, using the following format.**

   host-address `IN PTR` *host-name*

   See RFC 1886 for more information about AAAA and PTR records.

Example—DNS Zone File

```
vallejo        IN        AAAA   2::9256:a00:20ff:fe12
IN        AAAA   fec0::9256:a00:20ff:fe12:528
```

Example—DNS 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.
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.c.e.f \
    IN        PTR        vallejo.Eng.apex.COM.
```

Monitoring IPv6

The following commands are modified to accommodate the Solaris implementation of IPv6.

- `ifconfig`(1M)
- `netstat`(1M)
- `snoop`(1M)
- `ping`(1M)
- `traceroute`(1M)

You can use the new additions to conduct diagnostics. For conceptual descriptions of these commands, see "IPv6 Extensions to the `ifconfig` Utility" on page 325 and "IPv6 Extensions to Existing Utilities" on page 332.

Monitoring IPv6 Task Map

TABLE 15–2 Monitoring IPv6 Task Map

Task	Description	For Instructions, Go to ...
Display interface address assignments	Displays all, or just IPv4, or just IPv6 address assignments by using `ifconfig` command.	"How to Display Interface Address Assignments" on page 309
Display network status	Displays all sockets and routing table entries. Displays inet address family for IPv4. Displays inet6 address family for IPv6. Displays statistics for IPv6 or ICMPv6 counters of interfaces by using the `netstat` command.	"How to Display Network Status" on page 310
Control the display output of IPv6 related commands	Controls the output of the `ping`, `netstat`, `ifconfig`, 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 IPv6 Related Commands" on page 313
Monitor only IPv6 network traffic	Displays all IPv6 packets by using the `snoop` command.	"How to Monitor Only IPv6 Network Traffic" on page 314
Probe all multihomed host addresses	Checks all addresses by using the `ping` command.	"How to Probe All Multihomed Host Addresses" on page 315
Trace all routes	Uses the `traceroute` command.	"How to Trace All Routes" on page 315

▼ How to Display Interface Address Assignments

You can use the `ifconfig` command to display all address assignments as well as just IPv4 or IPv6 address assignments.

- **On the command line, type the following command.**

```
% ifconfig [option]
```

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

Example—Displaying Addressing Information for All Interfaces

```
% ifconfig -a
lo0: flags=1000849 mtu 8232 index 1
        inet 120.10.0.1 netmask ff000000
le0: flags=1000843 mtu 1500 index 2
        inet 120.46.86.54 netmask ffffff00 broadcast 120.146.86.255
        ether 8:0:73:56:a8
lo0: flags=2000849 mtu 8252 index 1
        inet6 ::1/128
le0: flags=2000841 mtu 1500 index 2
        ether 8:0:20:56:a8
        inet6 fe80::a00:fe73:56a8/10
le0:1: flags=2080841 mtu 1500 index 2
        inet6 fec0::56:20ff:fe73:56a8/64
le0:2: flags=2080841 mtu 1500 index 2
        inet6 2::56:a00:fe73:56a8/64
```

Example—Displaying Addressing Information for All IPv4 Interfaces

```
% ifconfig -a4
lo0: flags=1000849 mtu 8232 index 1
        inet 120.10.0.1 netmask ff000000
le0: flags=1000843 mtu 1500 index 2
        inet 120.46.86.54 netmask ffffff00 broadcast 120.46.86.255
        ether 8:0:20:56:a8
```

Example—Displaying Addressing Information for All IPv6 Interfaces

```
% ifconfig -a6
lo0: flags=2000849 mtu 8252 index 1
        inet6 ::1/128
```

```
le0: flags=2000841 mtu 1500 index 2
        ether 8:0:20:56:a8
        inet6 fe80::a00:fe73:56a8/10
le0:1: flags=2080841 mtu 1500 index 2
        inet6 fec0::56:20ff:fe73:56a8/64
le0:2: flags=2080841 mtu 1500 index 2
        inet6 2::56:a00:fe73:56a8/64
```

▼ How to Display Network Status

These procedures enable you to display the following structure formats for network data by using the `netstat` command:

- All sockets and routing table entries
- Inet address family for IPv4
- Inet6 address family for IPv6
- Statistics per interface—IPv6/ICMPv6 counters

● **On the command line, type the following command.**

```
% netstat [option]
```

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

Example—Displaying All Sockets and Routing Table Entries

```
% netstat -a
UDP: IPv4
   Local Address         Remote Address     State
-------------------- -------------------- -------
      *.*                                  Unbound
      *.apexrpc                             Idle
      *.*                                  Unbound
            .
            .
UDP: IPv6
   Local Address                    Remote Address                   State
If
--------------------------------- ---------------------------------- -------
      *.*                                                           Unbound
      *.time                                                        Idle
      *.echo                                                        Idle
      *.discard                                                     Idle
      *.daytime                                                     Idle
      *.chargen                                                     Idle

TCP: IPv4
```

```
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
      *.*                  *.*                0      0     0      0 IDLE
      *.apexrpc            *.*                0      0     0      0 LISTEN
      *.*                  *.*                0      0     0      0 IDLE
      *.ftp                *.*                0      0     0      0 LISTEN
localhost.427              *.*                0      0     0      0 LISTEN
      *.telnet             *.*                0      0     0      0 LISTEN
tn.apex.COM.telnet is.Eng.apex.COM         8760      0   8760     0 ESTABLISHED
tn.apex.COM.33528 np.apex.COM.46637        8760      0   8760     0 TIME_WAIT
tn.apex.COM.33529 np.apex.COM.apexrpc      8760      0   8760     0 TIME_WAIT
TCP: IPv6
   Local Address     Remote Address   Swind Send-Q  Rwind Recv-Q   State   If
----------------- -----------------   ----- ------  ----- ------   -----
      *.*                *.*             0      0      0      0   IDLE
      *.ftp              *.*             0      0      0      0   LISTEN
      *.telnet           *.*             0      0      0      0   LISTEN
      *.shell            *.*             0      0      0      0   LISTEN
      *.smtp             *.*             0      0      0      0   LISTEN
        .
        .
 2::56:8.login     something.1023    8640      0   8640      0 ESTABLISHED
 fe80::a:a8.echo   fe80::a:89        8640      0   8640      0 ESTABLISHED
 fe80::a:a8.ftp    fe80::a:90        8640      0   8640      0 ESTABLISHED
```

Example—Displaying Inet Address Family for IPv4

```
% netstat -f inet
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
tn.apex.COM.telnet    is.apex.COM.35388    8760      0  8760   0  ESTABLISHED
tn.apex.COM.1022      alive-v4.nfsd        8760      0  8760   0  ESTABLISHED
tn.apex.COM.1021      sl.apex.COM.nfsd     8760      0  8760   0  ESTABLISHED
.
.
tn.apex.COM.33539     np.apex.COM.apexrpc  8760      0  8760   0  TIME_WAIT
```

Example—Displaying Inet6 Address Family for IPv4

```
% netstat -f inet6
TCP: IPv6
   Local Address        Remote Address     Swind Send-Q Rwind Recv-Q   State      If
-------------------- -------------------- ----- ------ ----- ------ ----------- -----
2::56:a8.login        something.1023       8640      0  8640      0 ESTABLISHED
fe80::a0:a8.echo      fe80::a0:de.35389    8640      0  8640      0 ESTABLISHED
.
.
fe80::a0:a8.ftp-data  fe80::a0:de.35394   25920      0 25920      0 TIME_WAIT
```

Example—Displaying Statistics Per Interface (IPv6/ICMPv6 Counters)

```
% netstat -sa
RAWIP
        rawipInDatagrams    =  1407     rawipInErrors       =     0
        rawipInCksumErrs    =     0     rawipOutDatagrams   =     5
        rawipOutErrors      =     0

UDP
        udpInDatagrams      =  7900     udpInErrors         =     0
        udpOutDatagrams     =  7725     udpOutErrors        =     0

TCP     tcpRtoAlgorithm     =     4     tcpRtoMin           =   200
        tcpRtoMax           = 60000     tcpMaxConn          =    -1
                .
                .
IPv4    ipForwarding        =     2     ipDefaultTTL        =   255
        ipInReceives        =406345     ipInHdrErrors       =     0
        ipInAddrErrors      =     0     ipInCksumErrs       =     0
                .
                .
IPv6 for lo0
        ipv6Forwarding      =     2     ipv6DefaultHopLimit =     0
        ipv6InReceives      =     0     ipv6InHdrErrors     =     0
                .
                .
IPv6 for le0
        ipv6Forwarding      =     2     ipv6DefaultHopLimit =   255
        ipv6InReceives      =   885     ipv6InHdrErrors     =     0
                .
                .
IPv6    ipv6Forwarding      =     2     ipv6DefaultHopLimit =   255
        ipv6InReceives      =   885     ipv6InHdrErrors     =     0
                .
                .
ICMPv4  icmpInMsgs          =   618     icmpInErrors        =     0
        icmpInCksumErrs     =     0     icmpInUnknowns      =     0
        icmpInDestUnreachs  =     5     icmpInTimeExcds     =     0
                .
                .
ICMPv6 for lo0
        icmp6InMsgs         =     0     icmp6InErrors       =     0
        icmp6InDestUnreachs =     0     icmp6InAdminProhibs =     0
                .
                .
ICMPv6 for le0
        icmp6InMsgs         =   796     icmp6InErrors       =     0
        icmp6InDestUnreachs =     0     icmp6InAdminProhibs =     0
        icmp6InTimeExcds    =     0     icmp6InParmProblems =     0
                .
                .
ICMPv6  icmp6InMsgs         =   796     icmp6InErrors       =     0
```

```
        icmp6InDestUnreachs =     0     icmp6InAdminProhibs =     0
                .
                .
IGMP:
       2542 messages received
          0 messages received with too few bytes
          0 messages received with bad checksum
       2542 membership queries received
                .
                .
```

▼ How to Control the Display Output of IPv6 Related Commands

You can control the output of the `netstat` and `ifconfig` commands. Create a file that is named `inet_type` in the `/etc/default` directory. Then specify the value of the DEFAULT_IP variable. For more information about the `inet_type`, see the `inet_type`(4) man page.

1. **Create the `/etc/default/inet_type` file.**
2. **Make one of the following entries, as needed.**
   - To display IPv4 information only, type:

     ```
     DEFAULT_IP=IP_VERSION4
     ```
   - To display both IPv4 and IPv6 information, type:

     ```
     DEFAULT_IP=BOTH
     ```

     or

     ```
     DEFAULT_IP=IP_VERSION6
     ```

Note – The `-4` and `-6` flags in `ifconfig` override the value set in the inet_type file. The `-f` flag in `netstat` also overrides the value that is set in the `inet_type` file.

Examples—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 results:

```
% ifconfig -a
lo0: flags=1000849 mtu 8232 index 1
        inet 120.10.0.1 netmask ff000000
```

```
le0: flags=1000843 mtu 1500 index 2
        inet 120.46.86.54 netmask ffffff00 broadcast 120.46.86.255
        ether 8:0:20:56:a8
lo0: flags=2000849 mtu 8252 index 1
        inet6 ::1/128
le0: flags=2000841 mtu 1500 index 2
        ether 8:0:20:56:a8
        inet6 fe80::a00:fe73:56a8/10
le0:1: flags=2080841 mtu 1500 index 2
        inet6 fec0::56:a00:fe73:56a8/64
le0:2: flags=2080841 mtu 1500 index 2
        inet6 2::56:a00:fe73:56a8/64
```

- When you specify the DEFAULT_IP=IP_VERSION4 variable in the `inet_type` file, you should have the following results:

```
% ifconfig -a
lo0: flags=849 mtu 8232
        inet 120.10.0.1 netmask ff000000
le0: flags=843 mtu 1500
        inet 120.46.86.54 netmask ffffff00 broadcast 120.46.86.255
        ether 8:0:20:56:a8
```

▼ How to Monitor Only IPv6 Network Traffic

In this procedure, you use the `snoop` command to display all IPv6 packets.

1. **Become superuser.**
2. **On the command line, type the following command.**

```
# snoop ip6
```

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

Example—Displaying Only IPv6 Network Traffic

```
# snoop ip6
Using device /dev/le (promiscuous mode)
fe80::a0:a1 -> ff02::9     IPv6  S=fe80::a0:a1 D=ff02::9 LEN=892
fe80::a0:de -> fe80::a0:a8 IPv6  S=fe80::a0:de D=fe80::a0:a8 LEN=104
fe80::a0:a8 -> fe80::a0:de IPv6  S=fe80::a0:a8 D=fe80::a0:de LEN=104
fe80::a0:a1 -> ff02::9     IPv6  S=fe80::a0:a1 D=ff02::9 LEN=892
fe80::a0:de -> fe80::a0:a8 IPv6  S=fe80::a0:de D=fe80::a0:a8 LEN=104
fe80::a0:a8 -> fe80::a0:de IPv6  S=fe80::a0:a8 D=fe80::a0:de LEN=152
fe80::a0:a1 -> ff02::9     IPv6  S=fe80::a0:a1 D=ff02::9 LEN=892
fe80::a0:de -> fe80::a0:a8 IPv6  S=fe80::a0:de D=fe80::a0:a8 LEN=72
fe80::a0:a8 -> fe80::a0:de IPv6  S=fe80::a0:a8 D=fe80::a0:de LEN=72
fe80::a0:a8 -> fe80::a0:de IPv6  S=fe80::a0:a8 D=fe80::a0:de LEN=72
fe80::a0:de -> fe80::a0:a8 IPv6  S=fe80::a0:de D=fe80::a0:a8 LEN=72
```

▼ How to Probe All Multihomed Host Addresses

In this procedure, you use the `ping` command to check all addresses.

- **On the command line, type the following command.**

```
% ping -a ipng11
ipng11 (2::102:a00:fe79:19b0) is alive
ipng11 (fec0::102:a00:fe79:19b0) is alive
ipng11 (190.68.10.75) is alive
```

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

▼ How to Trace All Routes

In this procedure, you use the `traceroute` command to trace all routes.

- **On the command line, type the following command.**

```
% traceroute -a <hostname>
```

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

Example—Tracing All Routes

```
% traceroute -a ipng11
traceroute: Warning: Multiple interfaces found; using 2::56:a0:a8 @ le0:2
traceroute to ipng11 (2::102:a00:fe79:19b0),30 hops max, 60 byte packets
 1  ipng-rout86 (2::56:a00:fe1f:59a1)  35.534 ms  56.998 ms *
 2  2::255:0:c0a8:717  32.659 ms  39.444 ms *
 3  ipng61.Eng.apex.COM (2::103:a00:fe9a:ce7b)  401.518 ms  7.143 ms *
 4  ipng12-00 (2::100:a00:fe7c:cf35)  113.034 ms  7.949 ms *
 5  ipng11 (2::102:a00:fe79:19b0)  66.111 ms *  36.965 ms

traceroute: Warning: Multiple interfaces found; using fec0::56:a8 @ le0:1
traceroute to ipng11 (fec0::10:b0), 30 hops max, 60 byte packets
 1  ipng-rout86 (fec0::56:a00:fe1f:59a1) 96.342 ms  78.282 ms 88.327 ms
 2  ipng8-tun1 (fec0::25:0:0:c0a8:717)  268.614 ms  508.416 ms  438.774 ms
 3  ipng61.Eng.apex.COM (fec0::103:a00:fe9a:ce7b)  6.356 ms * 713.166 ms
 4  ipng12-00 (fec0::100:a00:fe7c:cf35)  7.409 ms * 122.094 ms
 5  ipng11 (fec0::102:a00:fe79:19b0)  10.620 ms * *

traceroute to ipng11.eng.apex.com (190.68.10.75),30 hops max,40 byte packets
 1  rmpj17c-086.Eng.apex.COM (120.46.86.1)  4.360 ms  3.452 ms  3.479 ms
 2  flrmpj17u.Eng.apex.COM (120.46.17.131)  4.062 ms  3.848 ms  3.505 ms
 3  ipng8.Eng.apex.COM (120.68.7.23)  4.773 ms *  4.294 ms
 4  ipng61.Eng.apex.COM (120.68.10.104)  5.128 ms  5.362 ms *
 5  ipng12-20.Eng.apex.COM (120.68.10.62)  7.298 ms  5.444 ms *
```

```
6  ipng11.Eng.apex.COM (120.68.10.75)  8.053 ms  6.394 ms *
```

Configuring IPv6 Over IPv4 Tunnels

This section describes how you configure IPv6 over IPv4 tunnels.

For conceptual descriptions of tunnels, see "Solaris Tunneling Interfaces for IPv6" on page 334 and "Tunneling Mechanism" on page 344.

▼ How to Configure IPv6 Over IPv4 Tunnels

1. **Become superuser.**
2. **Create the file** `/etc/hostname6.ip.tun`*n*. **Use the values 0, 1, 2, and so on, for** *n*. **Then add entries following these steps.**

   a. **Add the tunnel source addresses. Then add the tunnel destination addresses.**

      `tsrc` *IPv4-source-addr* `tdst` *IPv4-destination-addr* `up`

   b. **(Optional) Add a logical interface for the source and destination IPv6 addresses.**

      `addif` *IPv6-source-address* *IPv6-destination-address* `up`

      Omit this step if you want the address autoconfigured for this interface. You do not need to configure link-local addresses for your tunnel. Link-local addresses are configured automatically.

When you finish configuring the tunnels, you must reboot.

Note – You must perform the same steps at the other end of the tunnel for bidirectional communication to occur.

If your system is to be configured as a router, you must also configure your router to advertise over tunneling interfaces before rebooting. See "How to Configure Your Router to Advertise Over Tunneling Interfaces" on page 317.

Example—Entry for IPv6 Configuration File to Autoconfigure IPv6 Addresses

This example shows a tunnel for which all IPv6 addresses are autoconfigured.

```
tsrc 129.146.86.138 tdst 192.168.7.19 up
```

Example—Entry in the IPv6 Configuration File for Manually Configured Addresses

This example shows a tunnel for which global source and global destination addresses are manually configured. The site-local source and site-local destination addresses are also manually configured.

```
tsrc 120.46.86.138 tdst 190.68.7.19 up
addif fec0::1234:a00:fe12:528 fec0::5678:a00:20ff:fe12:1234 up
addif 2::1234:a00:fe12:528 2::5678:a00:20ff:fe12:1234 up
```

▼ How to Configure Your Router to Advertise Over Tunneling Interfaces

Following these steps for each tunnel.

1. **Become superuser.**
2. **Edit the `/etc/inet/ndpd.conf` file. Add entries by using the following steps.**
   a. **Enable router advertisement over the tunneling interface.**

      `if ip.tun`*n* `AdvSendAdvertisements 1`

   b. **Add prefixes as needed.**

      `prefix` *interface-address* `ip.tun`*n*
3. **Reboot.**

Displaying IPv6 Name Service Information

This section provides procedures to display IPv6 name service information.

Displaying IPv6 Name Service Information Task Map

TABLE 15–3 Displaying IPv6 Name Service Information Task Map

Task	Description	For Instructions, Go to ...
Display name service information for IPv6	Displays name service information for IPv6 by using the `nslookup` command.	"How to Display IPv6 Name Service Information" on page 318
Verify that DNS IPv6 PTR records are updated correctly	Displays the PTR records for DNS IPv6 PTR records by using the `nslookup` command. Also uses the `set q=PTR` parameter.	"How to Verify That DNS IPv6 PTR Records Are Updated Correctly" on page 319
Display IPv6 information through NIS	Displays the IPv6 information through NIS by using the `ypmatch` command.	"How to Display IPv6 Information Through NIS" on page 320
Display IPv6 information through NIS	Displays the IPv6 information through NIS+ by using the `nismatch` command.	"How to Display IPv6 Information Through NIS+" on page 320
Display IPv6 information independent of name service	Displays the IPv6 information by using the `getent` command.	"How to Display IPv6 Information Independent of Name Service" on page 321

▼ How to Display IPv6 Name Service Information

In this procedure, you use the `nslookup` command to display IPv6 name service information.

1. **On the command line, type the following command:**

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

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

2. **To see information about a particular host, type the following commands at the angle bracket prompt:**

   >**set q=any**
   >*host-name*

3. **To see only AAAA records, type the following command at the angle bracket prompt:**

   >**set q=AAAA**

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

Example—Using `nslookup` to Display IPv6 Information

```
%  /usr/sbin/nslookup
Default Server:  space1999.Eng.apex.COM
Address:  120.46.168.78
> set q=any
> vallejo
Server:  space1999.Eng.apex.COM
Address:  120.46.168.78

vallejo.ipv6.eng.apex.com      IPv6 address = fec0::9256:a00:fe12:528
vallejo.ipv6.eng.apex.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. **On the command line, type the following command:**

   `% /usr/sbin/nslookup`

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

2. **To see the PTR records, type the following command at the angle bracket prompt:**

   `>set q=PTR`

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

Example—Using `nslook` to Display PTR Records

```
%  /usr/sbin/nslookup
Default Server:  space1999.Eng.apex.COM
Address:  120.46.168.78
> set q=PTR
> 8.2.5.0.2.1.e.f.f.f.0.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.2.0.0.0.ip6.int

8.2.5.0.2.1.e.f.f.f.0.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.2.0.0.0.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.

- **On the command line, type the following command:**

  `%` **`ypmatch`** *host-name* **`ipnodes.`***byname*

  The information about *host-name* displays.

  EXAMPLE 15–1 Example—Using `ypmatch` to Display IPv6 Information Through NIS

  ```
  % ypmatch vallejo ipnodes.byname
  fec0::9256:a00:20ff:fe12:528    vallejo
  2::9256:a00:20ff:fe12:528       vallejo
  ```

▼ How to Display IPv6 Information Through NIS+

In this procedure, you use the `nismatch` command to display IPv6 information through NIS.

- **On the command line, type the following command:**

  `%` **`nismatch`** *host-name* **`ipnodes.`***org-dir*

  The information about *host-name* displays.

  EXAMPLE 15–2 Example—Using `nismatch` to Display IPv6 Information Through NIS+

  ```
  % nismatch vallejo ipnodes.org_dir
  vallejo vallejo fec0::9256:a00:20ff:fe12:528
  vallejo vallejo 2::9256:a00:20ff:fe12:528
  ```

▼ How to Display IPv6 Information Independent of Name Service

- **On the command line, type the following command:**

  `% getent ipnodes` *host-name*

  The information about *host-name* displays.

  EXAMPLE 15–3 Example—Using `getent` to Display IPv6 Information Independent of Name Service

```
% getent ipnodes vallejo
2::56:a00:fe87:9aba        vallejo vallejo
fec0::56:a00:fe87:9aba     vallejo vallejo
```

CHAPTER 16

IPv6 Files and Commands (Reference)

The Solaris implementation of IPv6 consists primarily of changes to the TCP/IP stack, both at the kernel and user level. New IPv6 modules enable tunneling, router discovery, and stateless address autoconfiguration. This chapter describes the concepts that are associated with the Solaris implementation of IPv6.

This chapter contains the following information:

Overview of the Solaris IPv6 Implementation

As a part of the IPv4 to IPv6 transition, IPv6 specifies methods for encapsulating IPv6 packets within IPv4 packets. IPv6 also specifies IPv6 packets that are encapsulated within IPv6 packets. Consequently, a new module, `tun`(7M), which performs the actual packet encapsulation, has been added. This module, which is known as the tunneling module, is plumbed and is configured by using the `ifconfig` utility the

same as any physical interface. This module enables the tunneling module to be pushed between IP device and IP module. Tunneling devices also have entries in the system interface list.

The `ifconfig`(1M) utility is also modified. You use this utility to create the IPv6 stack. This utility also supports new parameters that are described in this chapter.

The `in.ndpd`(1M) daemon is added to perform router discovery and stateless address autoconfiguration.

IPv6 Network Interface Configuration File

IPv6 uses the file `/etc/hostname6.`*interface* at start up to automatically define network interfaces in the same way IPv4 uses `/etc/hostname.`*interface*. A minimum of one `/etc/hostname.*` or `/etc/hostname6.*` file should exist on the local machine. The Solaris installation program creates these files for you. In the file name, replace *interface* with the device name of the primary network interface.

The file name has the following syntax:

`hostname.`*interface*
`hostname6.`*interface*

Interface has the following syntax:

dev`[.`*Module*`[.`*Module* `...]]`*PPA*

Dev	A network interface device. The device can be a physical network interface, such as `le`, `qe`, and so on, or a logical interface, such as a tunnel. See "Solaris Tunneling Interfaces for IPv6" on page 334 for more details.
Module	The list of one or more streams modules to be pushed onto the device when the device is plumbed.
PPA	The physical point of attachment.

The syntax [.[.]] is also accepted.

The following list shows examples of valid file names:

```
hostname6.le0
hostname6.ip.tun0
hostname.ip.tun0
```

IPv6 Interface Configuration File Entry

The autoconfiguration of interfaces in IPv6 enables a node to compute its own link-local address that is based on its link-layer address. Consequently, the interface configuration file for IPv6 might not have an entry. In this instance, the startup scripts configure an interface. The node then "learns" of other addresses and prefixes through the neighbor discovery daemon, `in.ndpd`. If you require static addresses for an interface, use the `ifconfig` utility. Consequently, the address or host name is stored in `/etc/hostname6.`*interface* (or `/etc/hostname.`*interface*. The content is passed to `ifconfig` when the interface is configured.

In this instance, the file contains only one entry. The entry is the host name or IP address that is associated with the network interface. For example, suppose `smc0` is the primary network interface for a machine that is called `ahaggar`. The `/etc/hostname6.*` file for the interface would have the name `/etc/hostname6.smc0`. The file would contain the entry `ahaggar`.

The networking start up script examines the number of interfaces and the existence of the `/etc/inet/ndpd.conf` file to start routing daemons and packet forwarding. See "How to Configure a Solaris IPv6 Router" on page 305.

IPv6 Extensions to the `ifconfig` Utility

The `ifconfig` utility now enables IPv6 interfaces and the tunneling module to be plumbed. The `ifconfig`(1M) utility uses an extended set of ioctls to configure both IPv4 and IPv6 network interfaces. The following table shows the set of options that are added to this utility. See "How to Display Interface Address Assignments" on page 309 for a description of useful diagnostic procedures that use this utility.

TABLE 16–1 New `ifconfig` Utility Options

Option	Description
`index`	Set the interface index.
`tsrc/tdst`	Set tunnel source or destination.
`addif`	Create the next available logical interface.
`removeif`	Delete a logical interface with a specific IP address.
`destination`	Set the point-to-point destination address for an interface.
`set`	Set an address, netmask, or both for an interface.
`subnet`	Set the subnet address of an interface.
`xmit/-xmit`	Enable or disable packet transmission on an interface.

"Enabling IPv6 Nodes" on page 304 provides IPv6 configuration procedures.

Examples—New `ifconfig` Utility Options

The following usage of the `ifconfig` command creates the `hme0:3` logical interface to the 1234::5678/64 IPv6 address. This command enables the interface with the `up` option. The command also reports status. The command disables the interface. Finally, the command deletes the interface.

EXAMPLE 16–1 Examples—Using `addif` and `removeif`

```
# ifconfig hme0 inet6 addif 1234::5678/64 up
Created new logical interface hme0:3

# ifconfig hme0:3 inet6
hme0:3: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        inet6 1234::5678/64

# ifconfig hme0:3 inet6 down

# ifconfig hme0 inet6 removeif 1234::5678
```

The following usage of the `ifconfig` command opens the device that is associated with the physical interface name. The command configures the streams that are needed for TCP/IP to use the device. The command reports the status of the device. The command configures the source and the destination address for the tunnel. Finally, the command reports the new status of the device after the configuration.

EXAMPLE 16–2 Examples—Using `tsrc/tdst` and `index`

```
# ifconfig ip.tun0 inet6 plumb index 13

# ifconfig ip.tun0 inet6
ip.tun0: flags=2200850<POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu
1480 index 13
        inet tunnel src 0.0.0.0
        inet6 fe80::/10 --> ::

# ifconfig ip.tun0 inet6 tsrc 120.46.86.158 tdst 120.46.86.122

# ifconfig ip.tun0 inet6
ip.tun0: flags=2200850<POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu
1480 index 13
        inet tunnel src 120.46.86.158  tunnel dst 120.46.86.122
        inet6 fe80::8192:569e/10 --> fe80::8192:567a
```

Nodes With Multiple Network Interfaces

If a node contains more than one network interface, you must create additional `/etc/hostname.`*interface* files for the additional network interfaces.

IPv4 Behavior

For example, consider the system `timbuktu`, which is shown in Figure 4–1. This system has two network interfaces. This system also functions as a router. The primary network interface `le0` is connected to network `192.9.200`. The IP address of the system is `192.9.200.70`. The host name for the system is `timbuktu`. The Solaris installation program creates the `/etc/hostname.le0 file` for the primary network interface. The installation program also enters the `timbuktu` host name in the file.

The second network interface is `le1`. This interface is connected to network `192.9.201`. Although this interface is physically installed on the `timbuktu` system, the interface must have a separate IP address. Therefore, you have to create manually the `/etc/hostname.le1` file for this interface. The entry in the file is the name of the router, `timbuktu-201`.

IPv6 Behavior

If IPv6 is to be configured, only the interfaces for `/etc/hostname6.le0` and `/etc/hostname6.le1` must exist. Each interface address is configured automatically when the system is started.

IPv6 Daemons

This section describes the following IPv6 daemons:

- `in.ndpd` – Daemon for IPv6 autoconfiguration
- `in.ripngd` – Network routing daemon for IPv6
- `inetd` – Internet services daemon

in.ndpd Daemon

This daemon implements router discovery and auto-address configuration for IPv6. The following table shows the supported options.

TABLE 16–2 in.ndpd Daemon Options

Option	Description
-d	Turns on debugging for all events
-D	Turns on specific debugging
-f	File to read configuration from (instead of default file)
-I	Prints related information for each interface
-n	Does not loop back router advertisements
-r	Ignores received packets
-v	Verbose mode (reports various types of diagnostic messages)
-t	Turns on packet tracing

Parameters control the actions in.ndpd. Those parameters are set in the /etc/inet/ndpd.conf configuration file and the /var/inet/ndpd_state.*interface* startup file (if the parameters exist).

When the /etc/inet/ndpd.conf file exists, the file is parsed and used to configure a node as a router. The following table lists the valid keywords that might appear in this file. When a host is booted, routers might not be immediately available, or advertised packets by the router might be dropped and 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 time that the file is 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 the state files. The in.ndpd daemon automatically maintains the state files.

TABLE 16–3 /etc/inet/ndpd.conf Keywords

Keywords	Description
ifdefault	Specifies router behavior for all interfaces. Use the following syntax to set router parameters and corresponding values: ifdefault [*variable value*]

TABLE 16–3 `/etc/inet/ndpd.conf` Keywords *(Continued)*

Keywords	Description
`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*]

Note – The `ifdefault/prefixdefault` entries must precede the `if` and `prefix` entries in the configuration file.

See the `in.ndpd`(1M) man page and see also the `ndpd.conf`(4) man page for a list of configuration variables and allowable values.

Example—`/etc/inet/ndpd.conf` File

The following example provides a template of commented lines and also shows an example of how the keywords and configuration variables are used.

```
# ifdefault       [variable value]*
# prefixdefault [variable value]*
# if ifname     [variable value]*
# prefix prefix/length ifname
#
#  Per interface configuration variables
#
#DupAddrDetectTransmits
#AdvSendAdvertisements
#MaxRtrAdvInterval
#MinRtrAdvInterval
#AdvManagedFlag
#AdvOtherConfigFlag
#AdvLinkMTU
#AdvReachableTime
#AdvRetransTimer
#AdvCurHopLimit
#AdvDefaultLifetime
#
# Per Prefix:  AdvPrefixList configuration variables
#
#
#AdvValidLifetime
```

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

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

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 qe2 AdvSendAdvertisements 1
prefix 2:0:0:54::/64 qe2
prefix fec0:0:0:54::/64 qe2
```

`in.ripngd` Daemon

The `in.ripngd` daemon implements the RIP next-generation routing protocol for IPv6 routers. RIP next generation defines the IPv6 equivalent of RIP. RIP is a widely used IPv4 routing protocol that is based on the Bellman-Ford distance vector algorithm. The following table shows the supported options.

TABLE 16–4 `in.ripngd` Daemon Options

Option	Description
`-p` *n*	*n* specifies the alternate port number that is used to send/receive RIPNG packets.
`-q`	Suppresses routing information.
`-s`	Forces routing information whether 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` Internet Services Daemon

An IPv6–enabled server is a server that can handle IPv4 or IPv6 addresses. The server uses the same protocol that the corresponding client uses. The

`/etc/inet/inetd.conf` file contains the list of servers that `inetd`(1M) invokes when this daemon receives an Internet request over a socket. Each socket-based Internet server entry is composed of a single line that uses the following syntax:

service_name socket_type proto flags user server_pathname args

See the `inetd.conf`(4) man page for a description of the possible values for each field. In the Solaris operating environment, to specify a service as IPv6-enabled in the `/etc/inet/inetd.conf` file, you must specify the *proto* field as `tcp6` or `udp6`. If the service is IPv4-only, the *proto* field must be specified as `tcp` or `udp`. By specifying a *proto* value of `tcp6` or `udp6` for a service, `inetd` passes the specific daemon an AF_INET6 socket.

The following entry in the `inetd.conf` file depicts a `udp` server (myserver) that can communicate with both IPv4 and IPv6 client applications.

EXAMPLE 16–3 Server Communicating With Both IPv4 and IPv6 Client Applications

```
myserver   dgram   udp6    wait    root    /usr/sbin/myserver    myserver
```

An IPv6–enabled server can inherit an AF_INET (IPv4 only) or an AF_INET6 (IPv6 and IPv4) socket from `inetd`. The *proto* value for the service is specified as either `tcp6` (`udp6`) or `tcp` (`udp`). For these types of servers, you can also specify two `inetd.conf` entries. You can specify *proto* as `tcp`. You can also specify *proto* as `tcp6`.

Note – Because AF_INET6 sockets work with either the IPv4 or IPv6 protocols, specifying a *proto* value of `tcp6` (`udp6`) is sufficient.

See *Programming Interfaces Guide* for details on writing various types of IPv6–enabled servers.

All servers that are provided with Solaris software require only one `inetd` entry that specifies *proto* as `tcp6` or `udp6`. However, the remote shell server (`shell`) and the remote execution server (`exec`) must have an entry for both the `tcp` and `tcp6` *proto* values. The following example shows the `inetd` entries for `rlogin`, `telnet`, `shell`, and `exec`.

EXAMPLE 16–4 `inetd.conf` Entries for Servers Provided With Solaris Software

```
login stream    tcp6 nowait root  /usr/sbin/in.rlogind  in.rlogind
telnet stream    tcp6 nowait root /usr/sbin/in.telnetd in.telnetd
shell    stream    tcp    nowait    root    /usr/sbin/in.rshd    in.rshd
shell    stream    tcp6    nowait    root    /usr/sbin/in.rshd    in.rshd
exec    stream    tcp    nowait    root    /usr/sbin/in.rexecd    in.rexecd
exec    stream    tcp6    nowait    root    /usr/sbin/in.rexecd    in.rexecd
```

TCP Wrappers are a public domain utility that is used to monitor and to filter incoming requests for various network services, such as telnet. If you specify TCP Wrappers as the *server_pathname* for any of these services, you must ensure that *TCP Wrappers* are IPv6 capable. Otherwise, you must specify *proto* as `tcp` or `udp` for those services that are being used with *TCP Wrappers*.

In addition, if you replace a Solaris utility with another implementation, you must verify if the implementation of that service supports IPv6. If the implementation does not support IPv6, then you must specify the *proto* value as either `tcp` or `udp`.

Note – If you specify *proto* as `tcp` or `udp` only, the service uses only IPv4. You need to specify proto as `tcp6` or `udp6` to enable either IPv4 or IPv6 connections. If the service does not support IPv6, then do not specify `tcp6` or `udp6`.

See IPv6 extensions to the Socket API in *Programming Interfaces Guide* for more details on writing IPv6 enabled servers that use sockets.

IPv6 Extensions to Existing Utilities

User-level interface changes also include extensions to the following utilities:

- `netstat`(1M)
- `snoop`(1M)
- `route`(1M)
- `ping`(1M)
- `traceroute`(1M)

The `ifconfig`(1M) utility has also changed. See "IPv6 Extensions to the `ifconfig` Utility" on page 325 for a description.

`netstat`(1M)

In addition to displaying IPv4 network status, `netstat` can display IPv6 network status as well. You can choose which protocol information to display by setting the DEFAULT_IP value in the `/etc/default/inet_type` file and 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 with the `-f` option. For more information on the `inet_type` file, see the `inet_type`(4) man page.

The new `-p` option displays the net-to-media table, which is the ARP table for IPv4 and neighbor cache for IPv6. See the `netstat`(1M) man page for details. See "How to Display Network Status" on page 310 for descriptions of procedures that use this command.

`snoop`(1M)

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. By specifying the `ip` or `ip6` protocol keywords, 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 Only IPv6 Network Traffic" on page 314 for a description of procedures that use this command.

`route`(1M)

This utility now operates on both IPv4 and IPv6 routes. By default, `route` operates on IPv4 routes. If you use the option `-inet6` on the command line immediately after the `route` command, operations are performed on IPv6 routes. See the `route`(1M) man page for details.

`ping`(1M)

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.

Additionally, you can ping all the addresses of a multihomed target host by using the `-a` command-line option. See the `ping`(1M) man page for details. See "How to Probe All Multihomed Host Addresses" on page 315 for a description of a procedure that uses this command.

`traceroute`(1M)

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. See the `traceroute`(1M) man page for details.

Controlling Display Output

You can control how the `netstat` and `ifconfig` commands display output:

- Use keywords that are added to the command line to specify either `inet` or `inet6` addresses.
- Set the configuration variable DEFAULT_IP in the `/etc/default/inet_type` file.

You can set the value of DEFAULT_IP to IP_VERSION4, IP_VERSION6, or BOTH. If you do not create this file by specifying the DEFAULT_IP, then `netstat` and `ifconfig` displays both versions.

Note – The `inet` or `inet6` keyword option overrides the value that is set in the `inet_type` file when you use the `netstat` and `ifconfig` commands.

See "How to Control the Display Output of IPv6 Related Commands" on page 313 for a description of procedures.

Solaris Tunneling Interfaces for IPv6

Tunneling interfaces have the following format:

`ip.tun` *ppa*

ppa is the physical point of attachment.

Note – The Solaris software does not yet support encapsulating packets within IPv6 packets.

At system startup, the tunneling module (`tun`) is pushed (by `ifconfig`) on top of IP to create a virtual interface. The push is accomplished by creating the appropriate `hostsname6.*` file.

For example, to create a tunnel to encapsulate IPv6 packets over an IPv4 network (IPv6 over IPv4), you create the following file name:

```
/etc/hostname6.ip.tun0
```

The content of this file is passed to `ifconfig`(1M) after the interfaces have been plumbed. The content becomes the parameters necessary to configure a point-to-point tunnel.

The following listing is an example of entries in `hostname6.ip.tun0` file.

EXAMPLE 16–5 `hostname6.`*interface* Entries

```
tsrc 120.68.100.23 tdst 120.68.7.19 up
addif 1234:1234::1 5678:5678::2 up
```

In this example, the IPv4 source and destination addresses are used as tokens to autoconfigure IPv6 link-local addresses of the source and destination for the `ip.tun0` interface. Two interfaces are configured, the `ip.tun0` interface, and a logical interface (`ip.tun0:1`) that has the source and destination IPv6 addresses specified by the `addif` command.

As mentioned previously, the contents of these configuration files are passed to `ifconfig` without change when the system is started as multiuser. The previous example is equivalent to the following:

```
# ifconfig ip.tun0 inet6 plumb
# ifconfig ip.tun0 inet6 tsrc 120.68.100.23 tdst 120.68.7.19 up
# ifconfig ip.tun0 inet6 addif 1234:1234::1 5678:5678::2 up
```

The following display shows the output of `ifconfig -a` for this tunnel.

```
ip.tun0: flags=2200850<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480
index 6
        inet tunnel src 120.68.100.23  tunnel dst 120.68.7.19
        inet6 fe80::c0a8:6417/10 --> fe80::c0a8:713
ip.tun0:1: flags=2200850<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480
index 5
        inet6 1234:1234::1/128 --> 5678:5678::2
```

You can configure more logical interfaces by adding lines to the configuration file by using the following syntax:

```
addif IPv6-source IPv6-destination up
```

Note – Either end of the tunnel is an IPv6 router that advertises one or more prefixes over the tunnel. You do not need `addif` commands in the tunnel configuration files. Only `tsrc` and `tdst` might be required because all other addresses are autoconfigured.

In some situations, specific source and destination link-local addresses need to be manually configured for a particular tunnel. Change the first line of the configuration file to include these link-local addresses. The following line is an example:

```
tsrc 120.68.100.23 tdst 120.68.7.19 fe80::1/10 fe80::2 up
```

Notice that the source link-local address has a prefix length of 10. In this example, the `ip.tun0` interface resembles the following:

```
ip.tun0: flags=2200850<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480
index 6
        inet tunnel src 120.68.100.23  tunnel dst 120.68.7.19
        inet6 fe80::1/10 --> fe80::2
```

For specific information about `tun`, see the `tun`(7M) man page. For a general description of tunneling concepts during the transition to IPv6, see "Tunneling Mechanism" on page 344. For a description of a procedure for configuring tunnels, see "How to Configure IPv6 Over IPv4 Tunnels" on page 316.

IPv6 Extensions to Solaris Name Services

This section describes naming changes that were introduced by the implementation of IPv6 in the Solaris 8 release. You can store IPv6 addresses in any of the Solaris naming services (NIS, NIS+, DNS, and files). You can also use NIS and NIS+ over IPv6 RPC transports to retrieve any NIS or NIS+ data.

`/etc/inet/ipnodes` File

The `/etc/inet/ipnodes` file stores both IPv4 and IPv6 addresses. This file serves as a local database that associates the names of hosts with their IPv4 and IPv6 addresses. You should not store host names and their addresses in static files, such as `/etc/inet/ipnodes`. However, for testing purposes, store IPv6 addresses in a file in the same way that IPv4 addresses are stored in `/etc/inet/hosts`. The `ipnodes` file uses the same format convention as the `hosts` file. See "Network Databases"

on page 49 for a description of the hosts file. See ipnodes(4) man page for a description of the ipnodes file.

IPv6–aware utilities use the new /etc/inet/ipnodes database. The existing /etc/hosts database, which contains only IPv4 addresses, remains the same to facilitate existing applications. If the ipnodes database does not exist, IPv6–aware utilities use the existing hosts database.

Note – If you need to add addresses, you must add IPv4 addresses to both the hosts and ipnodes files. You add only IPv6 addresses to the ipnodes file.

Example—/etc/inet/ipnodes File

```
#
# Internet IPv6 host table
# with both IPv4 and IPv6 addresses
#
::1     localhost
2::9255:a00:20ff:fe78:f37c   fripp.guitars.com fripp fripp-v6
fe80::a00:20ff:fe78:f37c     fripp-11.guitars.com frippll
120.46.85.87                    fripp.guitars.com fripp fripp-v4
2::9255:a00:20ff:fe87:9aba   strat.guitars.com strat strat-v6
fe80::a00:20ff:fe87:9aba     strat-11.guitars.com stratll
120.46.85.177                   strat.guitars.com strat strat-v4 loghost
```

Note – You must group host name addresses by the host name, as shown in the previous example.

NIS Extensions for IPv6

Two new maps have been added for NIS: ipnodes.byname and ipnodes.byaddr. Similar to /etc/inet/ipnodes, these maps contain both IPv4 and IPv6 information. The hosts.byname and hosts.byaddr maps contain only IPv4 information. These maps remain the same to facilitate existing applications.

NIS+ Extensions for IPv6

A new table has been added for NIS+ named ipnodes.org_dir. The table contains both IPv4 and IPv6 addresses for a host. The hosts.org_dir table contains only IPv4 addresses for a host. This table remains the same to facilitate existing applications.

DNS Extensions for IPv6

A new resource record that is defined as an AAAA record has been specified by RFC 1886. This AAAA record maps a host name into an 128–bit IPv6 address. The PTR record is still used with IPv6 to map IP addresses into host names. The thirty two 4–bit nibbles of the 128–bit address are reversed for an IPv6 address. Each nibble is converted to its corresponding hexadecimal ASCII value. Then `ip6.int` is appended.

Changes to the `nsswitch.conf` File

In addition to the capability of looking up IPv6 addresses through `/etc/inet/ipnodes`, IPv6 support has been added to the NIS, NIS+, and DNS name services. Consequently, the `nsswitch.conf`(4) file has been modified to support IPv6 lookups. An ipnodes line has been added to the `/etc/nsswitch.conf` file. This addition enables you to perform lookups in the new databases for each of the Solaris Name Services (NIS, NIS+, DNS, and files). The following bold line shows an example of the ipnodes entry:

```
hosts:  files dns nisplus [NOTFOUND=return]
ipnodes: files dns nisplus [NOTFOUND=return]
```

Note – Before changing the `/etc/nsswitch.conf` file to search ipnodes in multiple name services, populate these ipnodes databases with IPv4 and IPv6 addresses. Otherwise, unnecessary delays can result in the resolution of host addresses (including possible boot-timing delays).

The following diagram shows the new relationship between the `nsswitch.conf` file and the new name services databases for applications that use the `gethostbyname`() and `getipnodebyname`() commands. Items in italics are new. The `gethostbyname`() command checks only for IPv4 addresses that are stored in `/etc/inet/hosts`. The `getipnodebyname`() command consults the database that is specified in the `ipnodes` entry in the `nsswitch.conf` file. If the lookup fails, then the command consults the database that is specified in the `hosts` entry in the `nsswitch.conf` file.

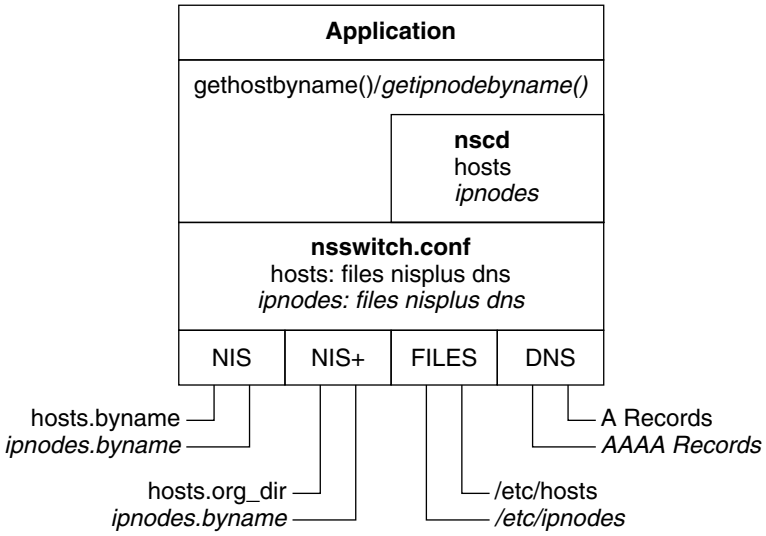


FIGURE 16–1 Relationship Between `nsswitch.conf` and Name Services

For more information on Naming Services, see *System Administration Guide: Naming and Directory Services*.

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 `nismatch` command works with the new NIS+ tables. The `nslookup` command can look up the new AAAA records in DNS. For a description of the changes to the name services see "NIS Extensions for IPv6" on page 337, "NIS+ Extensions for IPv6" on page 337, and "DNS Extensions for IPv6" on page 338.

For a description of procedures that use these commands, see "Displaying IPv6 Name Service Information" on page 318.

NFS and RPC IPv6 Support

NFS software and 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

The Solaris operating environment now supports IPv6–over-ATM a permanent virtual circuits (PVCs) and a static switched virtual circuits (SVC).

CHAPTER 17

Transitioning From IPv4 to IPv6 (Reference)

As hosts and routers are upgraded to support IPv6, the hosts and routers must be able to interoperate over the network with the hosts and router that support only IPv4. This chapter provides an overview of the standardized solutions to transitioning from IPv4 to IPv6. RFC 1933 also provides detailed solutions to the transition problem.

This chapter contains the following information:

Transition Requirements

The transition does not require any global coordination. Your sites and Internet service provider (ISP) can transition at their own pace. Furthermore, an effort has been made to minimize the number of dependencies during the transition. For instance, the transition does not require that routers be upgraded to IPv6 prior to upgrading hosts.

Different sites have different constraints when transitioning. Also, early adopters of IPv6 are likely to have different concerns than production users of IPv6. RFC 1933 defines the transition tools currently available. The rationale for transition is either the lack of IPv4 address space or the required use of new features in IPv6, or both. The IPv6 specification requires 100 per cent compatibility for the existing protocols and existing applications during the transition.

To understand the transition approaches, the following terms have been defined.

- **IPv4-only node** – A host or router that implements only IPv4. An IPv4-only node does not understand IPv6. The installed base of IPv4 hosts and routers that exist before the transition begins are IPv4-only nodes.
- **IPv6/IPv4 node** – A host or router that implements both IPv4 and IPv6, which is also known as *dual-stack*.
- **IPv6-only node** – A host or router that implements IPv6, and does not implement IPv4.
- **IPv6 node** – Any host or router that implements IPv6. IPv6/IPv4 and IPv6-only nodes are both IPv6 nodes.
- **IPv4 node** – Any host or router that implements IPv4. IPv6/IPv4 and IPv4-only nodes are both IPv4 nodes.
- **Site** – Piece of the private topology of the Internet that does not carry transit traffic for anybody and everybody. The site can span a large geographic area. For instance, the private network on a multinational corporation is one site.

Standardized Transition Tools

RFC 1933 defines the following transition mechanisms:

- When you upgrade your hosts and routers to IPv6, the hosts and routers retain their IPv4 capability. Consequently, IPv6 provides compatibility for all IPv4 protocols and applications. These hosts and routers are known as *dual-stack*.
- These hosts and routers use the name service (for example, DNS) to carry information about which nodes are IPv6 capable.
- IPv6 address formats can contain IPv4 addresses.
- You can tunnel IPv6 packets in IPv4 packets as a method of crossing routers that have not been upgraded to IPv6.

Implementing Dual-Stack

The term dual-stack normally refers to a complete duplication of all levels in the protocol stack from applications to the network layer. An example of complete duplication is the OSI and TCP/IP protocols that run on the same system. However, in the context of IPv6 transition, dual-stack means a protocol stack that contains both IPv4 and IPv6. The remainder of the stack is identical. Consequently, the same transport protocols (TCP, UDP, and so on) can run over both IPv4 and IPv6. Also, the same applications can run over both IPv4 and IPv6.

The following figure illustrates dual-stack protocols through the OSI layers.

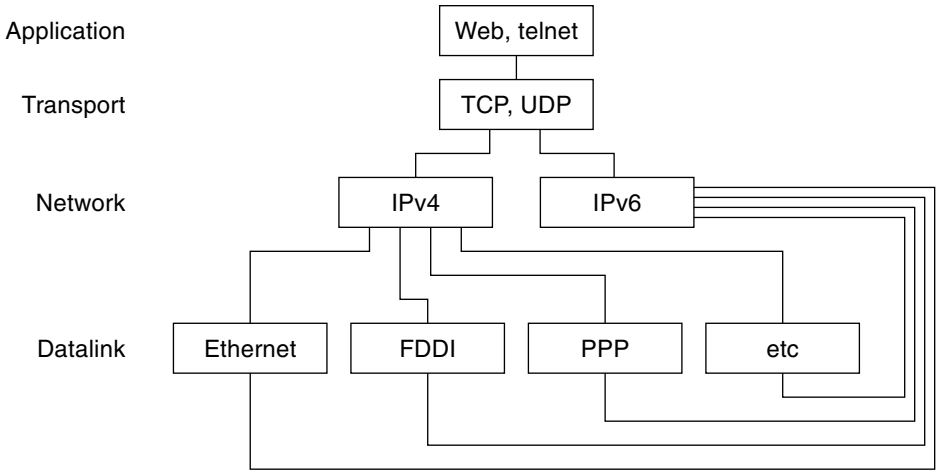


FIGURE 17–1 Dual-Stack Protocols

In the dual-stack method, subsets of both hosts and routers are upgraded to support IPv6, in addition to IPv4. The dual-stack approach ensures that the upgraded nodes can always interoperate with IPv4-only nodes by using IPv4.

Configuring Name Services

A dual node must determine if the peer can support IPv6 or IPv4 in order to check which IP version to use when transmitting. The control of the information that goes in the name service enables a dual node to determine which IP version to use. You define an IPv4 node's IP address and the IPv6 node's IP address in the name service. Thus, a dual node has both addresses in the name service.

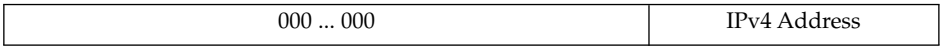
The presence of an IPv6 address in the name service also signifies that the node is reachable by using IPv6. However, the node is only reachable by nodes that obtain information from that name service. For example, placing an IPv6 address in NIS implies that the IPv6 host is reachable by using IPv6. However, the IPv6 host is only reachable by IPv6 and dual nodes that belong to that NIS domain. The placement of an IPv6 address in global DNS requires that the node is reachable from the Internet IPv6 *backbone*. This situation is no different than in IPv4. For example, the mail delivery operation requires that IPv4 addresses exist for nodes that can be reached by using IPv4. The same situation is true for the HTTP proxy operation. When no reachability exists in IPv4, for instance, because of firewalls, the name service must be partitioned into an *inside firewall* and *outside firewall* database. Consequently, the IPv4 addresses are visible only where the IPv4 addresses are reachable.

The protocol that is used to access the name service is independent of the type of address that can be retrieved from the name service. This name service support, coupled with dual-stacks, allows a dual node to use IPv4 when communicating with IPv4-only nodes. Also, this name service support allows a dual node to use IPv6 when communicating with IPv6 nodes. However, the destination must be reachable through an IPv6 route.

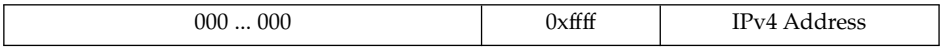
Using IPv4-Compatible Address Formats

In many instances, you can represent a 32-bit IPv4 address as a 128-bit IPv6 address. The transition mechanism defines the following two formats.

- **IPv4–compatible address**



- **IPv4–mapped address**



The compatible format is used to represent an IPv6 node. This format enables you to configure an IPv6 node to use IPv6 without having a *real* IPv6 address. This address format enables you to experiment with different IPv6 deployments because you can use automatic tunneling to cross IPv4–only routers. However, you cannot configure these addresses by using the IPv6 stateless address autoconfiguration mechanism. This mechanism requires existing IPv4 mechanisms such as DHCPv4 or static configuration files.

The mapped address format is used to represent an IPv4 node. The only currently defined use of this address format is part of the socket API. An application can have a common address format for both IPv6 addresses and IPv4 addresses. The common address format can represent an IPv4 address as a 128-bit mapped address. However, IPv4–to-IPv6 protocol translators also allow these addresses to be used.

Tunneling Mechanism

To minimize any dependencies during the transition, all the routers in the path between two IPv6 nodes do not need to support IPv6. This mechanism is called *tunneling*. Basically, IPv6 packets are placed inside IPv4 packets, which are routed

through the IPv4 routers. The following figure illustrates the tunneling mechanism through routers (R) that use IPv4.

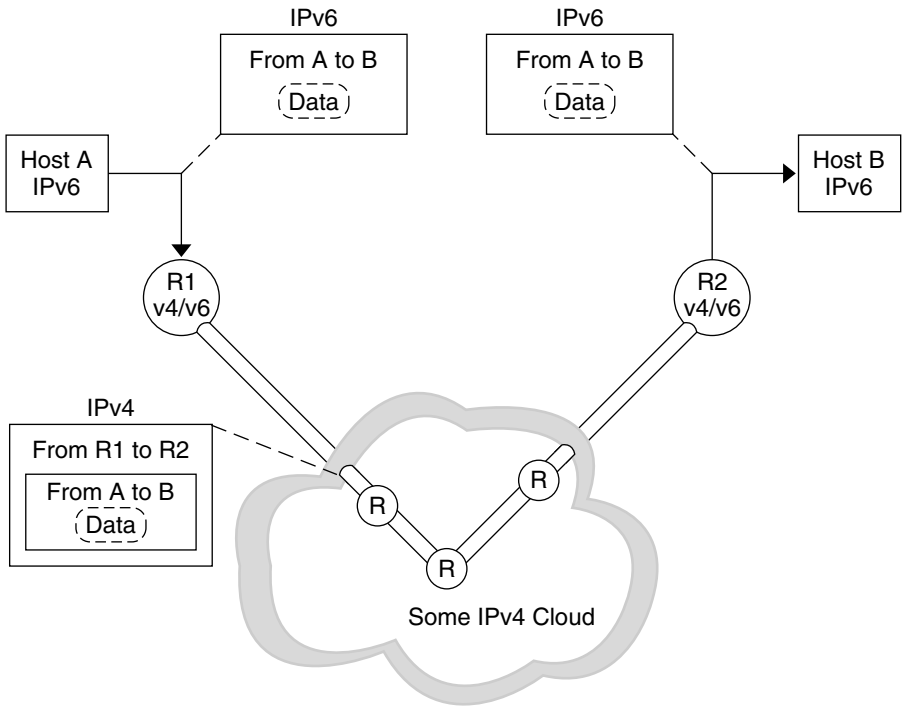


FIGURE 17–2 Tunneling Mechanism

The different uses of tunneling in the transition follow:

- Configured tunnels between two routers (as in the previous figure)
- Automatic tunnels that terminate at the dual hosts

A configured tunnel is currently used in the Internet for other purposes, for example, the MBONE (the IPv4 multicast backbone). Operationally, 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. This kind of tunnel is likely to be used on some parts of the Internet for the foreseeable future.

Automatic Tunnels

The automatic tunnels have a more limited use during early experimental deployment. Automatic tunnels require IPv4–compatible addresses and can be used to connect IPv6 nodes when IPv6 routers are not available. These tunnels can originate

either on a dual host or on a dual router by configuring an automatic tunneling network interface. The tunnels always terminate on the dual host. These tunnels work by dynamically determining the destination IPv4 address (the endpoint of the tunnel) by extracting the address from the IPv4–compatible destination address.

Interaction With Applications

Even on a node that has been upgraded to IPv6, the use of IPv6 is dependent on the applications. An application might not use a networking API that asks the name service for IPv6 addresses. The application might use an API (such as sockets) that requires changes in the application. Also, the provider of the API, such as an implementation of the `java.net` class might not support IPv6 addresses. In either situation, the node only sends and receives IPv4 packets like an IPv4 node would.

The following names have become standard terminology within the Internet community:

- **IPv6–unaware**—This application cannot handle IPv6 addresses. This application cannot communicate with nodes that do not have an IPv4 address.
- **IPv6–aware**—This application can communicate with nodes that do not have an IPv4 address, that is, the application can handle the larger IPv6 addresses. In some situations, the address might be transparent to the application, for example, when the API hides the content and format of the actual address.
- **IPv6–enabled**—This application can, in addition to being IPv6–aware, can use some IPv6–specific feature such as flow labels. The enabled applications can still operate over IPv4, though in a degraded mode.
- **IPv6–required**—This application requires some IPv6–specific feature and cannot operate over IPv4.

IPv4 and IPv6 Interoperability

During the gradual transition phase from IPv4 to IPv6, existing IPv4 applications must continue to work with newer IPv6–enabled applications. Initially, vendors provide host and router platforms that are running a dual-stack, that is, both an IPv4 protocol stack and an IPv6 protocol stack. IPv4 applications continue to run on a dual– stack that is also IPv6 enabled with at least one IPv6 interface. No changes need to be made to these applications (no porting required).

IPv6 applications that run on a dual-stack can also use the IPv4 protocol. IPv6 applications use an IPv4-mapped IPv6 address. Because of the design of IPv6, separate

applications (IPv4 and IPv6) are not needed. For example, you do not need an IPv4 client on a dual host to "talk" with a server on an IPv4-only host. Also, you do not need a separate IPv6 client to talk with an IPv6 server. Implementors need only to port their IPv4 client application to the new IPv6 API. The client can communicate with IPv4–only servers. The client can also communicate with IPv6 servers that run on either a dual host or an IPv6–only host.

The address that the client receives from the name server determines if IPv6 or IPv4 is used. For example, if the name server has an IPv6 address for a server, then the server runs IPv6.

The following table summarizes the interoperability between IPv4 and IPv6 clients and servers. The table assumes that the dual-stack host has both an IPv4 and IPv6 address in the respective name service database.

TABLE 17–1 Client-Server Applications: IPv4 and IPv6 Interoperability

Type of Application (Type of Node)	IPv6-Unaware Server (IPv4-Only Node)	IPv6-Unaware Server (IPv6-Enabled Node)	IPv6-Aware Server (IPv6-Only Node)	IPv6-Aware Server (IPv6-Enabled Node)
IPv6-unaware client (IPv4-only node)	IPv4	IPv4	X	IPv4
IPv6-unaware client (IPv6-enabled node)	IPv4	IPv4	X	IPv4
IPv6-aware client (IPv6-only node)	X	X	IPv6	IPv6
IPv6-aware client (IPv6-enabled node)	IPv4	(IPv4)	IPv6	IPv6

X means that the server cannot communicate with the client.

(IPv4) denotes that the interoperability depends on the address that is chosen by the client. If the client chooses an IPv6 address, the client fails. However, an IPv4 address that is returned to the client as an IPv4–mapped IPv6 address causes an IPv4 datagram to be sent successfully.

In the first phase of IPv6 deployment, most implementations of IPv6 are on dual-stack nodes. Initially, most vendors do not release IPv6–only implementations.

Site Transition Scenarios

Each site and each ISP requires different steps during the transition phase. This section provides some examples of site transition scenarios.

The first step to transition a site to IPv6 is to upgrade the name services to support IPv6 addresses. For DNS, upgrade to a DNS server that supports the new AAAA (quad-A), such as BIND 4.9.4 and later. Two new NIS maps and a new NIS+ table have been introduced for storing IPv6 addresses. The new NIS maps and NIS+ table can be created and administered on any Solaris system. See "IPv6 Extensions to Solaris Name Services" on page 336 for details on the new databases.

After the name service is able to distribute IPv6 addresses, you can start transitioning hosts. You can transition hosts in the following ways:

- Upgrade one host at a time. Use IPv4–compatible addresses and automatic tunneling. No routers need to be upgraded. Use this method for initial *experimental* transition. This method offers only a subset of the IPv6 benefits. This method does not offer stateless address autoconfiguration or IP multicast. You can use this scenario to verify that applications work over IPv6. This scenario also verifies that the application can use IPv6 IP-layer security.
- Upgrade one subnet at a time. Use configured tunnels between the routers. In this scenario, at least one router per subnet is upgraded to dual. The dual routers in the site are tied together by using configured tunnels. Then hosts on those subnets can use all the IPv6 features. As more routers become upgraded in this incremental scheme, you can remove the configured tunnels.
- Upgrade all the routers to dual before any host is upgraded. Though this method appears orderly, the method does not provide any IPv6 benefits until all the routers have been upgraded. This scenario constrains the incremental deployment approach.

Other Transition Mechanisms

The mechanisms that were specified previously handle interoperability between dual nodes and IPv4 nodes, if the dual nodes have an IPv4 address. The mechanisms do not handle interoperability between IPv6-only nodes and IPv4-only nodes. Also, the mechanisms do not handle interoperability between dual nodes that have no IPv4 address and IPv4-only nodes. Most implementations can be made dual. However, a dual implementation requires enough IPv4 address space to assign one address for every node that needs to interoperate with IPv4-only nodes.

Several possibilities enable you to accomplish this interoperability without requiring any new transition mechanisms.

- Use application layer gateways (ALG) that sit at the boundary between the IPv6-only nodes and the remainder of the Internet. Examples of ALGs in use today are HTTP proxies and mail relays.

- Companies are already selling network address translators (NAT) boxes for IPv4 that translate between the private IP addresses (for example, network 10—see RFC 1918) on the *inside* and other IP addresses on the *outside*. These companies will likely upgrade their NAT boxes to also support IPv6–to-IPv4 address translation.

Unfortunately, both ALG and NAT solutions create single points of failure. By using these solutions, the Internet becomes less effective. The IETF is working on a better solution for IPv6-only interoperability with IPv4-only nodes. One proposal is to use header translators with a way to allocate IPv4–compatible addresses on demand. Another proposal is to allocate IPv4–compatible addresses on demand and use IPv4 in IPv6 tunneling to bridge the IPv6-only routers.

The stateless header translator translates between IPv4 and IPv6 header formats if the IPv6 addresses in use can be represented as IPv4 addresses. The addresses must be IPv4-compatible or IPv4-mapped addresses. The support for these translators has been built into the IPv6 protocol. The translation can occur without any information loss, except for encrypted packets. Rarely used features such as source routing can produce information loss.

CHAPTER 18

IP Security Topics

Chapter 19	Provides overview information for IPsec
Chapter 20	Provides step-by-step instructions for setting up IPsec
Chapter 21	Provides an overview of IKE and step-by-step instructions for setting up IKE

CHAPTER 19

IPsec (Overview)

The IP Security Architecture (IPsec) provides cryptographic protection for IP datagrams in IPv4 and IPv6 network packets. The protection can include confidentiality, strong integrity of the data, partial sequence integrity (replay protection), and data authentication. IPsec is performed inside the IP module, and it can be applied with or without the knowledge of an Internet application. When used properly, IPsec is an effective tool in securing network traffic.

This chapter contains the following information:

Introduction to IPsec

IPsec provides security mechanisms that include secure datagram authentication and encryption mechanisms within IP. When you invoke IPsec, it applies the security mechanisms to IP datagrams that you have enabled in the IPsec global policy file. Applications can invoke IPsec to apply security mechanisms to IP datagrams on a per-socket level.

The following figure shows how an IP addressed packet, as part of an IP datagram, proceeds when IPsec has been invoked on an outbound packet. As you can see from the flow diagram, authentication header (AH) and encapsulating security payload

(ESP) entities can be applied to the packet. Subsequent sections describe how you apply these entities, as well as authentication and encryption algorithms.

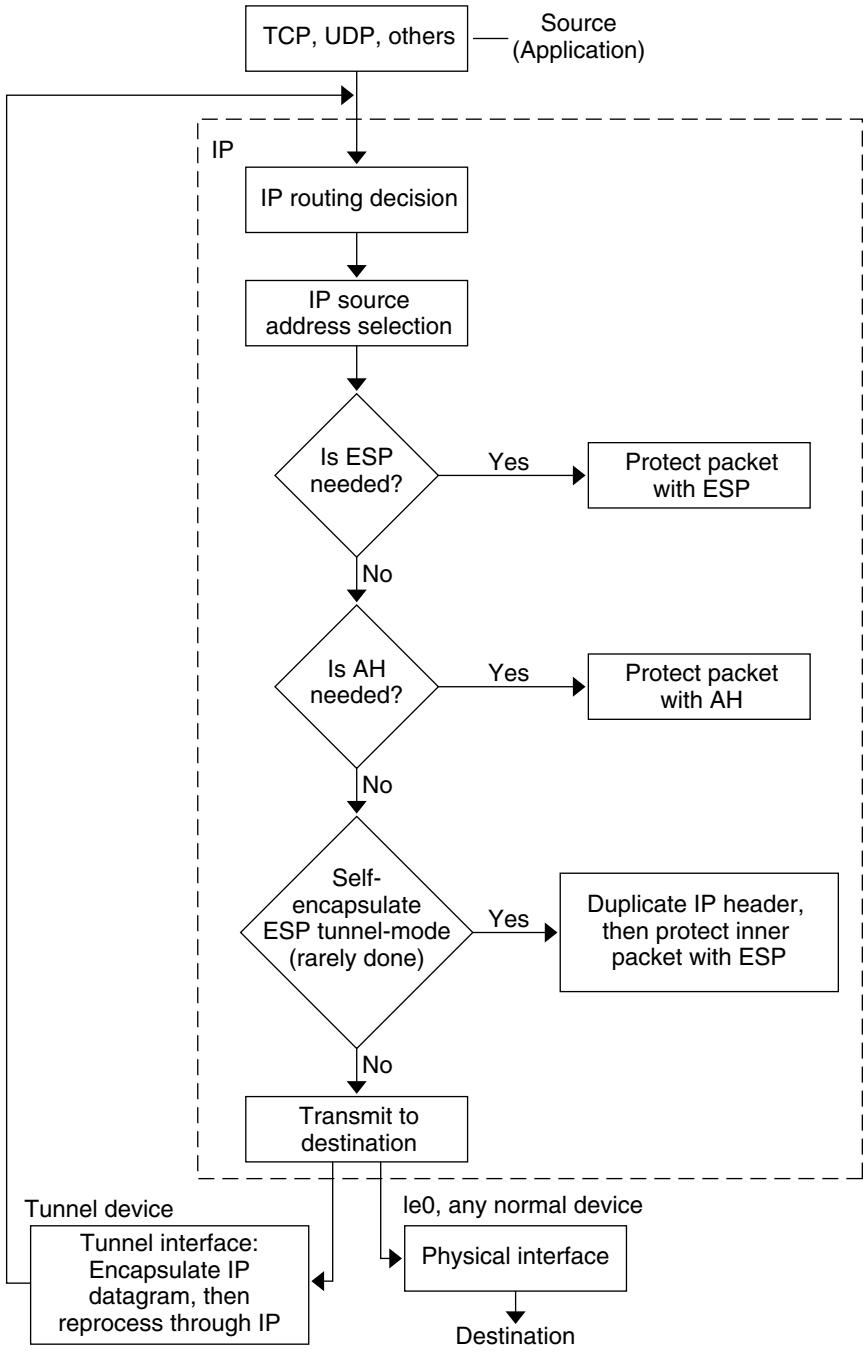


FIGURE 19–1 IPsec Applied to Outbound Packet Process

The following figure shows the IPsec inbound process.

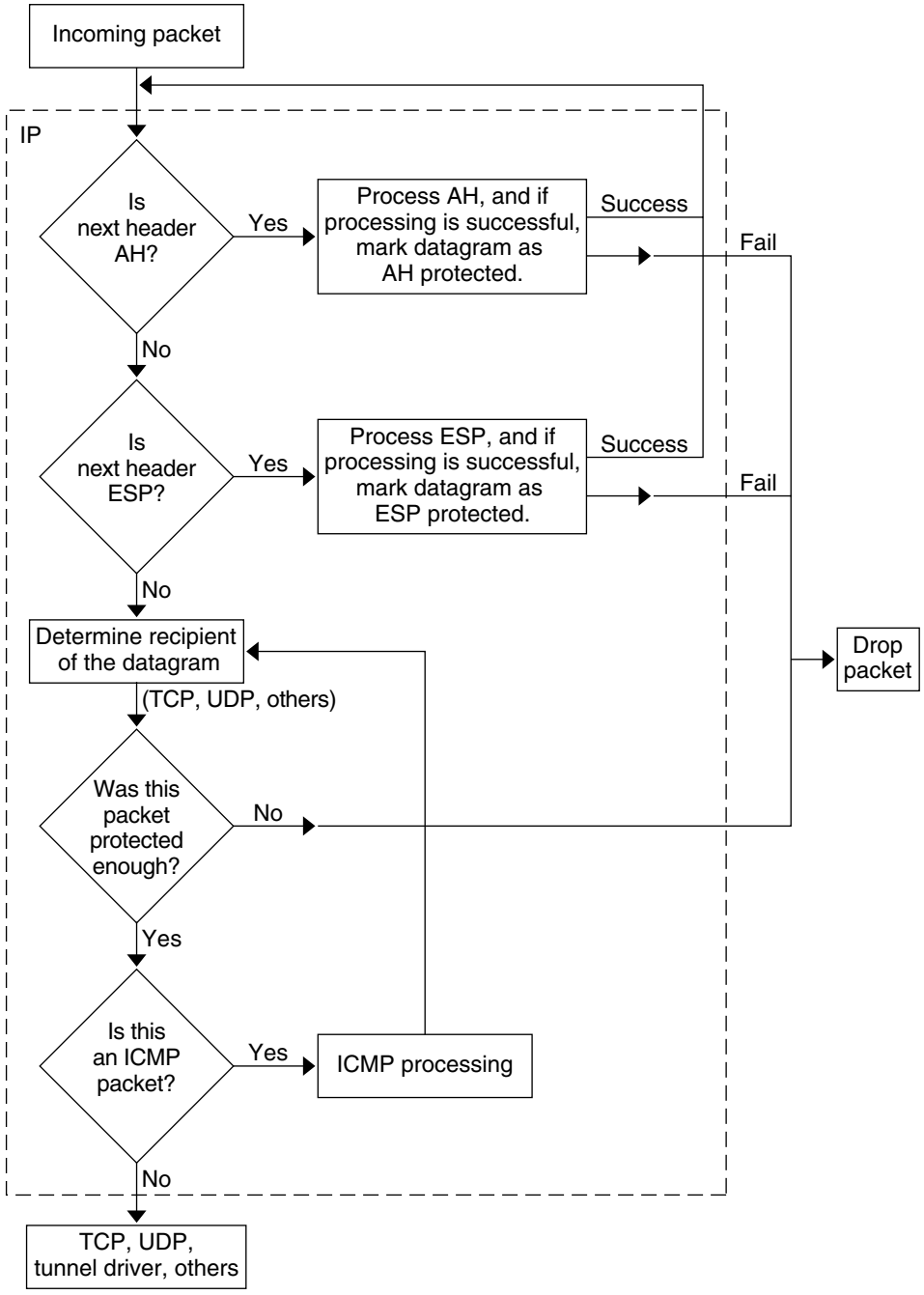


FIGURE 19–2 IPsec Applied to Inbound Packet Process

IPsec Security Associations

An IPsec security associations (SA) specifies security properties that are recognized by communicating hosts. Communicating hosts typically require two SAs to communicate securely. A single SA protects data in one direction — either to a single host or a group (multicast) address. Because most communication is peer-to-peer or client-to-server, two SAs must be present to secure traffic in both directions.

The AH or ESP, destination IP address, and Security Parameter Index (SPI) identifies an IPsec SA. The security parameters index, 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. A socket-based administration engine, the `pf_key`(7P) interface, enables privileged applications to manage the database. The `in.iked`(1M)`in.iked`(1M) daemon provides automatic key management.

Key Management

A security association contains keying information, algorithm choices, endpoint identities, and other parameters. Managing the keying material that SAs require for authentication and for encryption is called key management. The Internet Key Exchange (IKE) protocol handles key management automatically. You can also manage keys manually with the `ipseckey`(1M) command. Currently, SAs on IPv4 packets can use automatic key management, while SAs on IPv6 packets require manual management.

See “IKE Overview” on page 383 for how IKE manages cryptographic keys automatically for IPv4 hosts. See “Keying Utilities” on page 368 for how the administrator can manually manage the cryptographic keys by using the `ipseckey` command.

Protection Mechanisms

IPsec provides two mechanisms for protecting data:

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

Both mechanisms use security associations.

Authentication Header

The authentication header, a new IP header, provides strong integrity, partial sequence integrity (replay protection), and data authentication to IP datagrams. AH protects as much of the IP datagram as it can. AH cannot protect fields that change nondeterministically between sender and receiver. For example, the IP TTL field is not a predictable field and, consequently, not protected by AH. AH is inserted between the IP header and the transport header. The transport header can be TCP, UDP, ICMP, or another IP header when tunnels are being used. See the `tun`(7M) man page for details on tunneling.

Authentication Algorithms and the AH Module

IPsec implements AH as a module that is automatically pushed on top of IP. The `/dev/ipsecah` entry tunes AH with `ndd`(1M). Future authentication algorithms can be loaded on top of AH. Current authentication algorithms include HMAC-MD5 and HMAC-SHA-1. Each authentication algorithm has its own key size and key format properties. See the `authmd5h`(7M) and `authsha1`(7M) man pages for details.

Security Considerations

Replay attacks threaten any AH that does not enable replay protection. An AH does not protect against eavesdropping. Adversaries can still see data protected with AH.

Encapsulating Security Payload

The ESP provides confidentiality over what it encapsulates, as well as the services that AH provides, but only over that which it encapsulates. ESP's authentication services are optional. These services enable you to use ESP and AH together on the same

datagram without redundancy. Because ESP uses encryption-enabling technology, it must conform to U.S. export control laws.

ESP encapsulates its data, so it only protects the data that follows its beginning in the datagram. 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 it needs to. Unlike the authentication header (AH), ESP allows multiple kinds of datagram protection. Using only a single form of datagram protection can make the datagram vulnerable. For example, if you use ESP to provide confidentiality only, the datagram is still vulnerable to replay attacks and cut-and-paste attacks. Similarly, if ESP protects only integrity and does not fully protect against eavesdropping, it could provide weaker protection than AH.

Algorithms and the ESP Module

IPsec ESP implements ESP as a module that is automatically pushed on top of IP. The `/dev/ipsecesp` entry tunes ESP with `ndd`(1M). ESP allows encryption algorithms to be pushed on top of it, in addition to the authentication algorithms that are used in AH. Encryption algorithms include United States Data Encryption Standard (DES), Triple-DES (3DES), Blowfish, and AES. Each encryption algorithm has its own key size and key format properties. Because of export laws in the United States and import laws in other countries, not all encryption algorithms are available outside of the United States.

Security Considerations

An ESP without authentication is vulnerable to cut-and-paste cryptographic attacks and to eavesdropping attacks. When you use ESP without confidentiality, ESP is as vulnerable to replay as AH is.

Authentication and Encryption Algorithms

IPsec uses two types of algorithms, authentication and encryption. The authentication algorithms and the DES encryption algorithms are part of core Solaris installation. If you plan to use other algorithms that are supported for IPsec, you must install the Solaris Encryption Kit, which is provided on a separate CD.

Authentication Algorithms

Authentication algorithms produce an integrity checksum value or *digest* that is based on the data and a key. The authentication algorithm man pages describe the size of both the digest and key. The following table lists the authentication algorithms that are supported in the Solaris operating environment. The table also lists the format of the algorithms when they are used as security options to the IPsec utilities and their man page names.

TABLE 19–1 Supported Authentication Algorithms

Algorithm Name	Security Option Format	Man Page
HMAC-MD5	md5, hmac-md5	`authmd5h`(7M)
HMAC-SHA-1	sha, sha1, hmac-sha, hmac-sha1	`authsha1`(7M)

Encryption Algorithms

Encryption algorithms encrypt data with a key. The algorithms operate on data in units of a *block size*. The encryption algorithm man pages describe the size of both the block size and the key size. By default, the DES-CBC and 3DES-CBC algorithms are installed. You must install the Solaris Encryption Kit to make the AES and Blowfish algorithms available to IPsec. The kit is available on a separate CD that is *not* part of the Solaris 9 installation box. The *Encryption Kit Installation Guide* describes how to install the Solaris Encryption Kit.

The following table lists the encryption algorithms that are supported in the Solaris operating environment. The table also lists the format of the algorithms when they are used as security options to the IPsec utilities, their man page names, and the package that contains them.

TABLE 19–2 Supported Encryption Algorithms

Algorithm Name	Security Option Format	Man Page	Package
DES-CBC	des, des-cbc	`encrdes`(7M)	SUNWcsr, SUNWcarx.u
3DES–CBC or Triple-DES	3des, 3des-cbc	`encr3des`(7M)	SUNWcsr, SUNWcarx.u
Blowfish	blowfish, blowfish-cbc	`encrbfsh`(7M)	SUNWcryr, SUNWcryrx
AES-CBC	aes, aes-cbc	`encraes`(7M)	SUNWcryr, SUNWcryrx

Protection Policy and Enforcement Mechanisms

IPsec separates protection policy and enforcement mechanisms. You can enforce IPsec policies in the following places:

- On a system-wide level
- On a per-socket level

You use the `ipsecconf`(1M) command to configure system-wide policy.

IPsec applies system-wide policy to incoming and outgoing datagrams. You can apply some additional rules to outgoing 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. Resolving a conflict depends on which rule is parsed first. Except when a policy entry states that traffic should bypass all other policy, the traffic is automatically accepted. Outbound datagrams are either sent with protection or without protection. If protection is applied, the algorithms are either specific or non-specific. If policy normally protects a datagram, it can be bypassed by either an exception in system-wide policy, or by requesting a bypass in per-socket policy.

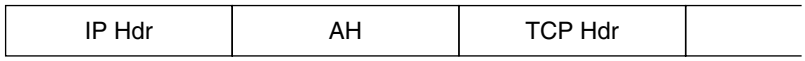
For intra-system traffic, policies are enforced, but actual security mechanisms are not applied. Instead, the outbound policy on an intra-system packet translates into an inbound packet that has had those mechanisms applied.

Transport and Tunnel Modes

When you invoke ESP or AH after the IP header to protect a datagram, you are using transport mode. An example follows. A packet starts off as:

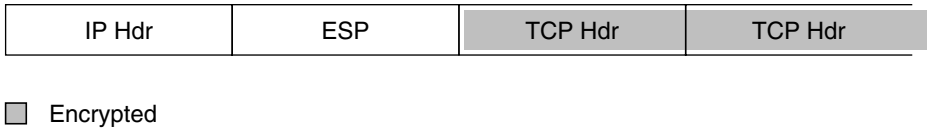
ESP, in transport mode, protects the data as follows:

AH, in transport mode, protects the data as follows:



AH actually covers the data before it appears in the datagram. Consequently, the protection that is provided by AH, even in transport mode, covers some of the IP header.

When an entire datagram is *inside* the protection of an IPsec header, IPsec is protecting the datagram in tunnel mode. Because AH covers most of its preceding IP header, tunnel mode is usually performed only on ESP. The previous example datagram would be protected in tunnel mode as follows:



Often, in tunnel mode, the outer (unprotected) IP header has different source and destination addresses from the inner (protected) IP header. The inner and outer IP headers can match if, for example, an IPsec-aware network program uses self-encapsulation with ESP. Self-encapsulation with ESP protects an IP header option.

The Solaris implementation of IPsec is primarily a transport mode IPsec implementation, which implements the tunnel mode as a special instance of the transport mode. The implementation treats IP-in-IP tunnels as a special transport provider. The `ifconfig`(1M) configuration options to set tunnels are nearly identical to the options available to socket programmers when enabling per-socket IPsec. Also, tunnel mode can be enabled in per-socket IPsec. In per-socket tunnel mode, the inner packet IP header has the same addresses as the outer IP header. See the `ipsec`(7P) man page for details on per-socket policy.

Trusted Tunnels

A configured tunnel is a point-to-point interface. It enables an IP packet to be encapsulated within an IP packet. Configuring a tunnel requires both a tunnel source and tunnel destination. See the `tun`(7M) man page and "Solaris Tunneling Interfaces for IPv6" on page 334 for more information.

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 securely, then you can trust the tunnel. That is, packets that exit the tunnel 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.

Virtual Private Networks

You can use IPsec to construct a Virtual Private Network (VPN). You do this by constructing an Intranet that uses the Internet infrastructure. For example, an organization that uses VPN technology to connect offices with separate networks, can deploy IPsec to secure traffic between the two offices.

The following figure illustrates how two offices use the Internet to form their VPN with IPsec deployed on their network systems.

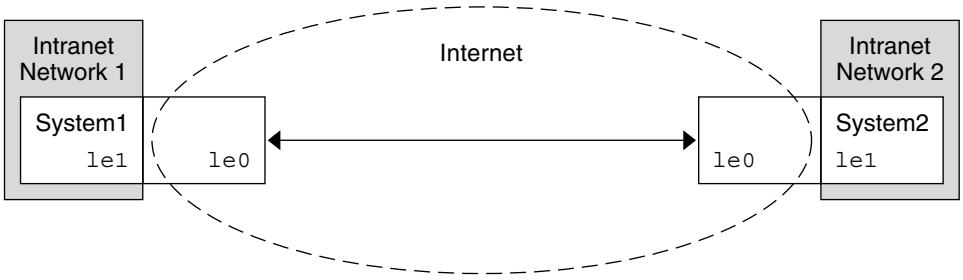


FIGURE 19–3 Virtual Private Network

See "How to Set Up a Virtual Private Network" on page 376 for a description of the setup procedure.

IPsec Utilities and Files

This section describes the IPsec initialization configuration file and various commands that enable you to manage IPsec within your network. For instructions about how to implement IPsec within your network, see "Implementing IPsec Task Map" on page 371.

TABLE 19–3 List of Selected IPsec Files and Commands

IPsec File or Command	Description
`/etc/inet/ipsecinit.conf` file	IPsec policy file. If this file exists, IPsec is activated at boot time.
`ipsecconf` command	IPsec activation command. `ipsecconf` activates IPsec policy when invoked with the `ipsecinit.conf` file as an argument. Useful for viewing and modifying current IPsec policy, and for testing.
`pf_key()` interface	Interface for security association database. Handles manual and automatic key management.
`ipseckey` command	Activation command for keys that are used in IPsec security associations. `ipseckey` provides keying material for IPsec security associations.
`/etc/inet/secret/ipseckeys` file	Keys for IPsec security associations. If the `ipsecinit.conf` exists, this file is automatically read at boot time.
`/etc/inet/ike/config` file	IKE configuration and policy file. If this file exists, the IKE daemon, `in.iked`(1M) starts and loads the `/etc/inet/ike/config` file. See "IKE Utilities and Files" on page 386.

IPsec Policy Command

You use the `ipsecconf`(1M) command to configure the IPsec policy for a host. When you run the command to configure policy, the system creates a temporary file named `ipsecpolicy.conf` to hold the IPsec policy entries. The system immediately uses the file to check all outbound and inbound IP datagrams for policy. Forwarded datagrams are not subjected to policy checks that are added by using this command. See `ifconfig`(1M) and `tun`(7M) for information on how to protect forwarded packets.

You must become superuser to invoke the `ipsecconf` command. The command accepts entries that protect traffic in both directions, and entries that protect traffic in only one direction.

Policy entries that do not specify a direction and contain the patterns `laddr host1` (local address) and `raddr host2` (remote address) protect traffic in both directions for the named host. Thus, you need only one entry for each host. A policy entry of the pattern `saddr host1 daddr host2` (source address to destination address) protects traffic in only one direction, that is, either outbound or inbound. Thus, to protect traffic in both directions, you need to pass the `ipsecconf` command another entry, as in `saddr host2 daddr host1`.

You can see the policies that are configured in the system when you issue the `ipsecconf` command without any arguments. The command displays each entry with an *index* followed by a number. You can use the `-d` option with the index to delete a particular policy in the system. The command displays the entries in the order that they were added, which is not necessarily the order in which the traffic match occurs. To view the order in which the traffic match occurs, use the `-l` option.

The `ipsecpolicy.conf` file is deleted when the system shuts down. To ensure that IPsec policy is active when the machine boots, you can create an IPsec policy file, `/etc/inet/ipsecinit.conf`, that the `inetinit` script reads during startup.

IPsec Policy File

To invoke IPsec security policies when you start the Solaris operating environment, you create an IPsec initialization configuration file with your specific IPsec policy entries. You should name the file `/etc/inet/ipsecinit.conf`. See the `ipsecconf`(1M) man page for details about policy entries and their format. After policies are configured, you can use the `ipsecconf` command to delete a policy temporarily, or to view the existing configuration.

Example—`ipsecinit.conf` File

The Solaris software includes a sample IPsec policy file that you can use as a template to create your own `ipsecinit.conf` file. This sample file is named `ipsecinit.sample` and it contains the following entries:

```
#
#ident    "@(#)ipsecinit.sample    1.6  01/10/18 SMI"
#
# Copyright (c) 1999,2001 by Sun Microsystems, Inc.
# All rights reserved.
#
# This file should be copied to /etc/inet/ipsecinit.conf to enable IPsec
# systemwide policy (and as a side-effect, load IPsec kernel modules).
```

```
# Even if this file has no entries, IPsec will be loaded if
# /etc/inet/ipsecinit.conf exists.
#
# Add entries to protect the traffic using IPsec. The entries in this
# file are currently configured using ipsecconf from inetinit script
# after /usr is mounted.
#
# For example,
#
#     {rport 23} ipsec {encr_algs des encr_auth_algs md5}
#
# Or, in the older (but still usable) syntax
#
#    {dport 23} apply {encr_algs des encr_auth_algs md5 sa shared}
#    {sport 23} permit {encr_algs des encr_auth_algs md5}
#
# will protect the telnet traffic originating from the host with ESP using
# DES and MD5. Also:
#
#     {raddr 10.5.5.0/24} ipsec {auth_algs any}
#
# Or, in the older (but still usable) syntax
#
#    {daddr 10.5.5.0/24} apply {auth_algs any sa shared}
#    {saddr 10.5.5.0/24} permit {auth_algs any}
#
# will protect traffic to or from the 10.5.5.0 subnet with AH
# using any available algorithm.
#
#
# To do basic filtering, a drop rule may be used. For example:
#
#    {lport 23 dir in} drop {}
#    {lport 23 dir out} drop {}
#
# will disallow any remote system from telnetting in.
#
#
# WARNING:    This file is read before default routes are established, and
#        before any naming services have been started. The
#        ipsecconf(1M) command attempts to resolve names, but it will
#        fail unless the machine uses files, or DNS and the DNS server
#        is reachable via routing information before ipsecconf(1M)
#        invocation.  (that is, the DNS server is on-subnet, or DHCP
#        has loaded up the default router already.)
#
#        It is suggested that for this file, use hostnames only if
#        they are in /etc/hosts, or use numeric IP addresses.
#
#        If DNS gets used, the DNS server is implicitly trusted, which
#        could lead to compromise of this machine if the DNS server
#        has been compromised.
#
```

Security Considerations

If, for example, the `/etc/inet/ipsecinit.conf` file is sent from an NFS-mounted file system, an adversary can modify the data contained in the file. The outcome would be a change to the configured policy. Consequently, you should use extreme caution if transmitting a copy of the `ipsecinit.conf` file over a network.

Policy cannot be changed (is "latched") for TCP/UDP sockets on which a `connect`(3SOCKET) or `accept`(3SOCKET) has been issued. Adding new policy entries does not affect the latched sockets. This latching feature might change in the future, so you should not depend on this feature.

Ensure that you set up the policies before starting any communications, because existing connections might be affected by the addition of new policy entries. Similarly, do not change policies in the middle of a communication.

If your source address is a host that can be looked up over the network, and your naming system itself is compromised, then any names that are used are no longer trustworthy.

Security weaknesses often lie in misapplication of tools, not the tools themselves. You should be cautious when using the `ipsecconf` command. Use a console or other hard-connected TTY for the safest mode of operation.

IPsec Security Associations Database

Keying information for IPsec security services is maintained in a security association database (SADB). Security associations protect both inbound and outbound packets. A user process (or possibly multiple cooperating processes) maintains SADBs by sending messages over a special kind of socket. This is analogous to the method that is described in the `route`(7P) man page. Only a superuser can access an SADB.

The operating system might spontaneously emit messages in response to external events, such as a request for a new SA for an outbound datagram, or to report the expiration of an existing SA. You open the channel for passing SADB control messages by using the socket call that is described in the previous section. More than one key socket can be open per system.

Messages include a small base header, followed by a number of extension messages (zero or more). Some messages require additional data. The base message and all extensions must be 8-byte aligned. The GET message serves as an example. This message requires the base header, the SA extension, and the ADDRESS_DST extension. See the `pf_key`(7P) man page for details.

Keying Utilities

The IKE protocol is the automatic keying utility for IPv4 addresses. See Chapter 21 for how to set up IKE. The manual keying utility is the `ipseckey`(1M) command.

You use the `ipseckey` command to manually manipulate the security association databases with the `ipsecah`(7P) and `ipsecesp`(7P) protection mechanisms. You can also use the `ipseckey` command to set up security associations between communicating parties when automated key management is not available. An example is communicating parties that have IPv6 addresses.

While the `ipseckey` command has only a limited number of general options, it supports a rich command language. You can specify that requests should be delivered by means of a programmatic interface specific for manual keying. See the `pf_key`(7P) man page for additional information. When you invoke `ipseckey` with no arguments, it enters an interactive mode that displays a prompt that enables you to make entries. Some commands require an explicit security association (SA) type, while others permit you to specify the SA type and act on all SA types.

Security Considerations

The `ipseckey` command enables a privileged user to enter sensitive cryptographic keying information. If an adversary gains access to this information, the adversary can compromise the security of IPsec traffic. You should consider the following issues when you handle keying material and use the `ipseckey` command:

1. Have you refreshed the keying material? Periodic key refreshment is a fundamental security practice. Changing keys guards against potential weaknesses of the algorithm and keys, and limits the damage of an exposed key.
2. Is the TTY going over a network (interactive mode)?
   - If the TTY is in interactive mode, then 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.
3. Is the file accessed over the network or readable to the world (`-f` option)?
   - An adversary can read a network-mounted file as it is being read. You should avoid using a world-readable file with keying material in it.
   - If your source address is a host that can be looked up over the network, and your naming system is compromised, then any names used are no longer trustworthy.

Security weaknesses often lie in misapplication of tools, not the tools themselves. You should be cautious when using the `ipseckey` command. Use a console or other hard-connected TTY for the safest mode of operation.

IPsec Extensions to Other Utilities

The `ifconfig` command has options to manage IPsec policy on a tunnel interface, and the `snoop` command can parse AH and ESP headers.

`ifconfig` Command

To support IPsec, the following security options have been added to the `ifconfig`(1M) command:

- `auth_algs`
- `encr_auth_algs`
- `encr_algs`

`auth_algs`

This option enables IPsec AH for a tunnel, with the authentication algorithm specified. The `auth_algs` option has the following format:

`auth_algs` *authentication_algorithm*

The algorithm can be either a number or an algorithm name, including the parameter *any*, to express no specific algorithm preference. You must specify all IPsec tunnel properties on the same command line. To disable tunnel security, specify the following option:

```
auth_alg none
```

See Table 19–1 for a list of available authentication algorithms and for pointers to the algorithm man pages.

`encr_auth_algs`

This option enables IPsec ESP for a tunnel, with the authentication algorithm specified. The `encr_auth_algs` option has the following format:

`encr_auth_algs` *authentication_algorithm*

For the algorithm, you can specify either a number or an algorithm name, including the parameter *any*, to express no specific algorithm preference. If you specify an ESP

encryption algorithm, but you do not specify the authentication algorithm, the ESP authentication algorithm value defaults to the parameter, *any*.

See Table 19–1 for a list of available authentication algorithms and for pointers to the algorithm man pages.

`encr_algs`

This option enables IPsec ESP for a tunnel with the encryption algorithm specified. The option has the following format:

`encr_algs` *encryption_algorithm*

For the algorithm, you can specify either a number or an algorithm name. You must specify all IPsec tunnel properties on the same command line. To disable tunnel security, specify the following option:

```
encr_alg none
```

If you specify an ESP authentication algorithm, but not an encryption algorithm, the ESP encryption value defaults to the parameter *null*.

See the `ipsecesp`(7P) man page or Table 19–2 for a list of available encryption algorithms and for pointers to the algorithm man pages.

`snoop` Command

The `snoop` command can now parse AH and ESP headers. Because ESP encrypts its data, `snoop` cannot see encrypted headers that are protected by ESP. AH does not encrypt data, so traffic can still be inspected with `snoop`. The `snoop -V` option shows when AH is in use on a packet. See the `snoop`(1M) man page for more details.

CHAPTER 20

Administering IPsec (Task)

This chapter provides procedures for implementing IPsec on your network.

This chapter contains the following information:

For overview information about IPsec, see Chapter 19. The `ipsecconf`(1M), `ipseckey`(1M), and `ifconfig`(1M) man pages also describe useful procedures in their respective Examples sections.

Implementing IPsec Task Map

TABLE 20–1 Implementing IPsec Task Map

Task	Description	For Instructions, Go To …
Secure traffic between two IPv6 systems	Involves adding addresses to the `/etc/inet/ipnodes` file, entering IPsec policy in the `/etc/inet/ipsecinit.conf` file, manually adding keys with the `ipseckey` command, and invoking the `ipsecinit.conf` file.	"How to Secure Traffic Between Two Systems" on page 372

TABLE 20–1 Implementing IPsec Task Map *(Continued)*

Task	Description	For Instructions, Go To …
Secure a Web server by using IPsec policy	Involves enabling only secure traffic by entering different security requirements for different ports in the `ipsecinit.conf` file, and activating the file.	"How to Secure a Web Server" on page 375
Set up a virtual private network	Involves turning off IP forwarding, turning on IP strict destination multihoming, disabling most network and Internet services, adding security associations, and configuring a secure tunnel. Also involves turning on IP forwarding, configuring a default route, and running the routing protocol.	"How to Set Up a Virtual Private Network" on page 376
Replace current security associations	Involves flushing current security associations and entering new ones on every affected system.	"How to Replace Current Security Associations" on page 380

IPsec Tasks

This section provides procedures that enable you to secure traffic between two systems, secure a Web server by using IPsec policy, and set up a virtual private network. The system names `enigma` and `partym` are examples only. Substitute the names of your systems for the names `enigma` and `partym`.

▼ How to Secure Traffic Between Two Systems

This procedure assumes that each system has two addresses, an IPv4 address and an IPv6 address, and that you are invoking AH protections by using one of the available algorithms. The procedure also assumes that you are sharing security associations. With shared security associations, only one pair of SAs is needed to protect the two systems.

1. **Become superuser on the system console.**

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

2. **On each system, add the addresses and host name for the other system in the** `/etc/inet/ipnodes` **file. The entries for one machine must be contiguous in the file:**

   a. **On a system that is named** `partym`**, type the following:**

   ```
   # Secure communication with enigma
   192.168.66.1 enigma
   fec0::10:20ff:fea0:21f7 enigma
   ```

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

   ```
   # Secure communication with partym
   192.168.55.2  partym
   fec0::9:a00:20ff:fe7b:b667 partym
   ```

   These names are examples only. Use the names of your systems when securing traffic between them.

   This step enables the boot scripts to use the system names without depending on nonexistent naming services.

3. **On each system, edit the** `/etc/inet/ipsecinit.conf` **file to add the IPsec policy entry:**

   a. **On** `enigma`**, type the following:**

   ```
   {laddr enigma raddr partym} ipsec {auth_algs any sa shared}
   ```

   b. **On** `partym` **, type the following:**

   ```
   {laddr partym raddr enigma} ipsec {auth_algs any sa shared}
   ```

4. **On each system, add a pair of security associations between the two systems.**

   On each system, edit a read-only (600 permissions) `/etc/inet/secret/ipseckeys` file, and type the following lines in this file:

   ```
   add ah spi random-number dst local-system authalg an_algorithm_name \
           authkey random-hex-string-of-algorithm-specified-length
   add ah spi random-number dst remote-system authalg an_algorithm_name \
           authkey random-hex-string-of-algorithm-specified-length
   ```

Note – The keys and SPI can and *should* be different for each security association.

5. **Reboot each system.**

```
# /etc/reboot
```

On reboot, the `/etc/inet/secret/ipseckeys` file is read before booting completes. When you change keys, ensure that the `ipseckeys` file is changed on both systems.

Example—Securing Traffic Between IPv4 Addresses

The following example describes how to secure traffic between systems with IPv4 addresses. The example uses automatic key management (IKE) to create security associations. IKE requires less administrative intervention, and scales easily to secure a large amount of traffic.

1. Replace the `/etc/inet/ipnodes` file in Step 2 of "How to Secure Traffic Between Two Systems" on page 372 with the `/etc/hosts` file, as in the following:

   On the system that is named `partym`, add `enigma`:

   ```
   # echo "192.168.66.1 enigma" >> /etc/hosts
   ```

   On the system that is named `enigma`, add `partym` to the `/etc/hosts` file:

   ```
   # echo "192.168.55.2 partym" >> /etc/hosts
   ```

2. Edit the `ipsecinit.conf` file to add the IPsec policy entries.
3. Use the `ike.config`(4) file rather than the `ipseckey` command to add security associations. See "IKE Tasks" on page 391 for the procedures.

   Note – You can also manually create the keys, as described in Step 4 in "How to Secure Traffic Between Two Systems" on page 372.

   ---

4. Reboot.

Example—Securing Traffic Between IPv6 Addresses Without Rebooting

The following example describes how to test secure traffic between systems with IPv6 addresses.

1. Do the "How to Secure Traffic Between Two Systems" on page 372 procedure through Step 4.
2. Instead of rebooting, add the security associations to the database by typing the `ipseckey` command with the `ipseckeys` file as an argument.

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

3. Activate IPsec policy with the `ipsecconf` command:

```
# ipsecconf -a /etc/inet/ipsecinit.conf
```

Note – Read the warning when you execute the command. A socket that is already in use (latched) provides an unsecured back door into the system.

▼ How to Secure a Web Server

A secure Web server requires that any incoming traffic that is not a Web client request pass security checks. The following procedure includes bypasses for Web traffic that is served on the Web server and for DNS client requests from this Web server. All other traffic requires ESP with 3DES and SHA-1 algorithms and uses a shared SA for outbound traffic. Sharing SAs avoids using too many security associations.

1. **Become superuser on the system console.**

   Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

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, it might also need to include port 53 for both TCP and UDP.

3. **Create a read-only file, using the file name of your choice, for example `IPsecWebInitFile`, and type the following lines in this file:**

```
# Web traffic that Web server should bypass.
    {sport 80 ulp tcp} bypass {dir out}
    {dport 80 ulp tcp} bypass {dir in}
    {sport 443 ulp tcp} bypass {dir out}
    {dport 443 ulp tcp} bypass {dir in}

    # Outbound DNS lookups should also be bypassed.
    {dport 53} bypass {dir out}
    {sport 53} bypass {dir in}

    # Require all other traffic to use ESP with 3DES and SHA-1.
    # Use a shared SA for outbound traffic, in order to avoid a
    # large supply of security associations.
    {} permit {encr_algs 3des encr_auth_algs sha}
    {} apply {encr_algs 3des encr_auth_algs sha sa shared}
```

   This configuration enables only secure traffic to access the system, with the bypass exceptions that are described in the previous step.

4. **Read the file you created in the previous step into** `/etc/inet/ipsecinit.conf`**.**

```
# vi  /etc/inet/ipsecinit.conf
 :r IPsecWebInitFile
 :wq!
```

5. **Reboot.**

   The `ipsecconf` command does not affect already-established TCP connections, whose policies are latched. Rebooting ensures that IPsec policy is in effect on all TCP connections. At reboot, the TCP connections latch policy as it is specified in the IPsec policy file.

```
# reboot
```

   The Web server now allows only Web-server traffic, as well as outbound DNS requests and replies. No other services work without enabling IPsec on a remote system. If keying material is handled automatically, the IKE daemon activates IPsec on a remote system with an IPv4 address. On a remote system with an IPv6 address, use the `ipseckey`(1M) command to enable IPsec on the remote system.

▼ How to Set Up a Virtual Private Network

This procedure shows you how to set up a VPN by using the Internet to connect two networks within an organization. The procedure then shows you how to secure the traffic between the networks with IPsec. This procedure assumes that the networks' `le1` interfaces are *inside* the VPN, and the `le0` interfaces are *outside* the VPN on the two systems that implement the VPN link.

The procedure also uses ESP with DES and MD5. The algorithms that are used affect the key lengths, 64 bits (56 bits + 8 bits parity) for DES and 128 bits for MD5. You must perform the following procedure on the two systems that act as the gateway through the Internet. For a description of VPNs, see "Virtual Private Networks" on page 363.

1. **Become superuser on the system console.**

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

---

2. **Turn off IP forwarding:**

```
# ndd -set /dev/ip ip_forwarding 0
```

   Turning off IP forwarding prevents packets from being forwarded from one network to another through this system.

3. **Turn on IP strict destination multihoming:**

```
# ndd -set /dev/ip ip_strict_dst_multihoming 1
```

Turning on IP strict destination multihoming ensures that packets for one of the system's destination addresses arrives on the interface to which that address is assigned.

When you use the `ndd`(1M) command to turn off IP forwarding and turn on IP strict destination, multihoming shuts down the flow of packets except to the system itself, and then only if the packets arrive on the interface that corresponds to the destination IP address.

4. **Disable most (if not all) network services on the Solaris machine by doing the following substeps, as needed:**

Note – The VPN router should allow very few incoming requests. You need to disable all processes that accept incoming traffic (for example, comment out lines in the `inetd.conf` file, kill SNMP, and so on). Alternately, you can use techniques similar to those in "How to Secure a Web Server" on page 375.

---

   a. **If `inetd.conf` has been edited to remove all but essential services, type the following command:**

      ```
      # pkill -HUP inetd
      ```

   b. **If `inetd.conf` has not been edited to remove all but essential services, type the following command on a command line:**

      ```
      # pkill inetd
      ```

   c. **Disable other Internet services, such as SNMP, NFS, and so on, by typing one or more commands such as the following examples, as needed:**

      ```
      # /etc/init.d/nfs.server stop
      # /etc/init.d/sendmail stop
      ```

      Disabling network services prevents IP packets from doing any harm to the system. For example, an SNMP daemon, telnet, or rlogin could be exploited.

5. **On each machine, add a pair of security associations between the two systems.**

   If the systems are using IPv4 addresses, the IKE daemon automatically creates the security associations after you have configured IKE to create them. You can use one of the following procedures to set up IKE for the VPN: "How to Configure IKE With Pre-Shared Keys" on page 392, "How to Configure IKE With Self-Signed Public Certificates" on page 397, or "How to Configure IKE With Public Keys Signed by a Certificate Authority" on page 399.

   If the systems are using IPv6 addresses, you must manually create the security associations by doing the following substeps:

   a. **Enable the `ipseckey` command mode:**

```
# ipseckey
>
```

The > prompt indicates that you are in `ipseckey` command mode.

b. Type the following command:

```
> add esp spi random-number src system1_addr dst system2_addr \
auth_alg md5 encr_alg des \
authkey random-hex-string-of-32-characters \
encrkey random-hex-string-of-16-characters
```

c. Press the Return key to execute the command.

d. Type the following command:

```
> add esp spi random-number src system2_addr dst system1_addr \
auth_alg md5 encr_alg des \
authkey random-hex-string-of-32-characters \
encrkey random-hex-string-of-16-characters
```

Note – The keys and SPI can and *should* be different for each security association.

---

e. Type Control-D or `quit` to exit this mode.

6. Configure a secure tunnel, `ip.tun0` that adds another physical interface from the IP perspective, by performing the following substeps:

a. On System 1, type the following commands:

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 system1-taddr system2-taddr \
tsrc system1-addr tdst system2-addr encr_algs des encr_auth_algs md5

# ifconfig ip.tun0 up
```

b. On System 2, type the following commands:

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 system2-taddr system1-taddr \
tsrc system2-addr tdst system1-addr encr_algs des encr_auth_algs md5

# ifconfig ip.tun0 up
```

7. On each machine, turn on (in this example) `le1:ip_forwarding` and `ip.tun0:ip_forwarding`:

```
# ndd -set /dev/ip le1:ip_forwarding 1

# ndd -set /dev/ip ip.tun0:ip_forwarding 1
```

`ip_forwarding` means that packets that arrive off an interface can be forwarded, *and* packets that leave this interface might have originated on another interface. To successfully forward a packet, both the receiving and transmitting interfaces must have their *ip_forwarding* turned on.

Because `le1` is *inside* the Intranet, and `ip.tun0` connects the two systems through the Internet, `ip_forwarding` must be turned on for these two interfaces.

The `le0` interface still has its `ip_forwarding` turned off. This configuration prevents someone on the *outside* (that is, the Internet) from injecting packets into the protected Intranet.

8. **On each machine, ensure that routing protocols do not advertise the default route within the Intranet:**

   ```
   # ifconfig le0 private
   ```

   While `le0` has `ip_forwarding` turned off, any routing protocol implementation (for example, `in.routed`) might still advertise that `le0` is a valid interface for forwarding packets to its peers inside the Intranet. Setting the interface's *private* flag helps reduce these advertisements.

9. **On each system, manually add a default route over `le0`:**

   ```
   # pkill in.rdisc

   # route add default router-on-le0-subnet
   ```

   Even though `le0` is not part of the Intranet, it does need to reach across the Internet to its peer machine. To do this, Internet routing information is needed. The VPN system looks like a host (as opposed to a router) to the rest of the Internet, so either using a default route or running router discovery is sufficient.

10. **Prevent `in.rdisc` from restarting when the system is rebooted by performing the following substeps:**

    a. **Put the IP address of the default router on the `le0` subnet in the file `/etc/defaultrouter`.**

       This step prevents `in.rdisc` from being started at reboot.

    b. **Prevent routing from occurring early in the boot sequence, and thus reduce vulnerability:**

       ```
       # touch /etc/notrouter
       ```

    c. **Edit the `/etc/hostname.ip.tun0` file and add the following lines.**

       ```
       system1-taddr system2-taddr tsrc system1-addr \
            tdst system2-addr encr_algs des encr_auth_algs md5 up
       ```

    d. **Create an `/etc/rc3.d/S99vpn_setup` file and type the following lines.**

       ```
       ndd -set /dev/ip le1:ip_forwarding 1
       ndd -set /dev/ip ip.tun0:ip_forwarding 1
       ```

```
ifconfig le0 private
in.routed
```

11. **On each machine, run a routing protocol:**

    ```
    # in.routed
    ```

    To prevent an adversary from having time to break your cryptosystem, you need to periodically replace the security associations that you created in Step 5 with new ones. Use the following procedure to replace your current security associations. If you are running an IPv4 network, the IKE module manages the replacement of security associations.

▼ How to Replace Current Security Associations

This procedure enables you to replace current security associations. You should do this procedure periodically so that an adversary has less time to break your cryptosystem.

1. **Become superuser on the system console.**

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

---

2. **On each system, flush the current security associations in `ipseckey` command mode:**

    ```
    # ipseckey

    > flush
    >
    ```

3. **Set new security associations for outbound packets:**

    ```
    > add esp spi new-random-number src local-system dst remote-system \
    auth_alg the_algorithm-name encr_alg the_algorithm-name \
    authkey random-hex-string-of-algorithm-specified-length \
    encrkey random-hex-string-of-algorithm-specified-length
    ```

4. **Press the Return key.**

    This step executes the command and redisplays the `ipseckey` command mode prompt.

5. **Set new security associations for inbound packets:**

    ```
    > add esp spi new-random-number src remote-system dst local-system \
    auth_alg the_algorithm-name encr_alg the_algorithm-name \
    authkey random-hex-string-of-algorithm-specified-length \
    ```

encrkey *random-hex-string-of-algorithm-specified-length*

Note – The keys and SPI can and *should* be different for each security association.

---

6. **Type Control-D or quit to exit this mode.**

Example—Replacing Security Associations in `ipseckeys` Files

The following example refreshes the keys on the systems `partym` and `enigma`, whose traffic was secured in "How to Secure Traffic Between Two Systems" on page 372. The assumption is that both systems are using the SHA1 algorithm for AH, and both systems are using IPv6 addresses.

1. Flush the current keys.
2. Edit the `ipseckeys` file on both systems to replace existing `SPI` and `authkey` values.

   a. Edit the `ipseckeys` file on `partym`:

   ```
   # for inbound packets
   add ah spi 0x55142 dst partym authalg sha1 \
           authkey 012345678921001234abcdeffedcba9876543210
   # for outbound packets
   add ah spi 0x235211 dst enigma authalg sha1 \
           authkey 21001234abcdef9876543210012345678 9fedcba
   ```

   b. Edit the `ipseckeys` file on `enigma`:

   ```
   # for inbound packets
   add ah spi 0x235235 dst enigma authalg sha1 \
           authkey 123456780123456789abcdeffedcba9876543210
   # for outbound packets
   add ah spi 0x123456 dst partym authalg sha1 \
           authkey abcdef98765432100123456789fed12345678bac
   ```

3. To make sure that latched sockets use the new keys, reboot both systems. The `ipseckeys` file is read automatically at boot time.

   ```
   # /usr/sbin/reboot
   ```

   If you are testing, you can place the new keys into the security database on each system without rebooting:

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

CHAPTER 21

Internet Key Exchange

Managing the keying material that IPsec SAs require for secure transmission of IP datagrams is called key management. Automatic key management requires a secure channel of communication for the creation, authentication, and exchange of keys. The Solaris operating environment uses Internet Key Exchange (IKE) to automate key management. IKE easily scales to provide a secure channel for a large volume of traffic. IPsec SAs on IPv4 packets can take advantage of IKE.

This chapter contains the following information:

IKE Overview

The Internet Key Exchange (IKE) daemon, `in.iked`(1M), negotiates and authenticates keying material for security associations in a protected manner. The daemon uses random seeds for keys from internal functions provided by the SunOS™. IKE provides Perfect Forward Secrecy (PFS), that is, the keys that protect data transmission are not used to derive additional keys, and seeds used to create data transmission keys are not reused.

When the IKE daemon discovers a remote host's public encryption key, the local system can then encrypt messages destined for the remote host whose public key it has discovered. The IKE daemon performs its job in two phases called exchanges.

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 ISAKMP (Internet Security Association and Key Management Protocol) Security Association, which is a secure channel for IKE to negotiate keying material for the IP datagrams. Unlike IPsec SAs, the ISAKMP security associations are bidirectional, so only one 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 interfaces that are affected, the algorithms that are used, the authentication method, and if PFS is used. The two authentication methods are pre-shared keys and public key certificates The public key certificates can be self–signed, or they can be issued by a Certificate Authority (CA) from a PKI (Public Key Infrastructure) vendor. Vendors include iPlanet™ Certificate Management System, Entrust, and Verisign.

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 hosts running the IKE daemon. IKE uses the secure channel that was created in Phase 1 to protect the transmission of keying material. The IKE daemon creates the keys from a random number generator (`/dev/random`), refreshes them at a configurable rate, and provides the keying material to algorithms specified in the IPsec policy configuration file.

Negotiating IKE

For two IKE daemons to authenticate each other requires a valid IKE configuration policy file, `ike.config`(4), and keying material. The policy file contains IKE policy entries that determine whether pre-shared keys or public key certificates are to be used to authenticate the Phase 1 exchange.

The key pair `auth_method preshared` indicates that pre-shared keys are used. Values for `auth_method` other than `preshared` are one indication that public key certificates are to be used. Public key certificates can be self-signed, or they can be installed from a PKI vendor.

Using Pre–Shared Keys

Pre-shared keys are created by an administrator on one system, and shared out of band with administrators of communicating systems. The administrator should take care to create large random keys and to protect the file and the out-of-band transmission. The keys are placed in the `/etc/inet/secret/ike.preshared` file on each system. The `ike.preshared`(4) file is for IKE as the `ipseckeys` file is for IPsec. Compromise of the keys in the `ike.preshared` file compromises all keys derived from them.

One system's pre-shared key must be identical to its communicating system's key. The keys are tied to a particular IP address, and are most secure when one administrator controls the communicating systems.

Using 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-Helman method of authenticating and negotiating keys. Public key certificates come in two flavors, self-signed, and certified by a Certificate Authority (CA).

Self-signed public key certificates are created by an administrator. The `ikecert local -ks` command creates the private part of the public-private key pair for the system. The administrator then gets the self-signed certificate output in X.509 format from the communicating system. The communicating 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 hosts.

Self-signed certificates are a halfway point between pre-shared keys and CAs. Unlike pre-shared keys, a self-signed certificate can be used on a mobile machine, or a machine that may be renumbered. To do this, the administrator uses a DNS (`www.example.org`) or EMAIL (`root@domain.org`) alternative name.

Public keys can be delivered by a PKI or a CA vendor. The public keys and their accompanying CAs are installed in the `/etc/inet/ike/publickeys` directory by the administrator. Vendors also issue certificate revocation lists (CRLs). Along with installing the keys and CAs, the administrator is responsible for installing the CRLs in the `/etc/inet/ike/crls` directory.

CAs have the advantage of being certified by an outside vendor, rather than by the administrator of the site. In a sense, CAs are notarized certificates. Like self-signed certificates, they can be used on a mobile machine, or one that may be renumbered. Unlike self-signed certificates, they very easily scale to protecting a large number of communicating systems.

IKE Utilities and Files

This section describes the IKE configuration files and various commands that implement IKE. For instructions about how to implement IKE for your IPv4 network, see "Implementing IKE Task Map" on page 391.

TABLE 21–1 List of IKE Files and Commands

File or Command	Description
`in.iked`(1M) daemon	Internet Key Exchange (IKE) daemon. Activates automated key management.
`ikeadm`(1M)	IKE administration command. For viewing and modifying IKE policy.
`ikecert`(1M)	Certificate database management command. For manipulating local public-key certificate databases.
`/etc/inet/ike/config` file	IKE policy configuration file. Contains the site's rules for matching inbound IKE requests and preparing outbound IKE requests. If this file exists, the `in.iked` daemon starts automatically at boot time.
`/etc/inet/secret/ike.preshared` file	Pre-shared keys file. Contains secret keying material for Phase 1 authentication.
`/etc/inet/secret/ike.privatekeys` file	Private keys directory. Contains the private keys that are part of a public-private key pair.
`/etc/inet/ike/publickeys` directory	Directory to hold public keys and certificate files. By default, includes Sun certificates. Contains the public key part of a public-private key pair.
`/etc/inet/ike/crls` directory	Directory to hold revocation lists for public keys and certificate files.

IKE Daemon

The `in.iked`(1M) daemon automates the management of cryptographic keys for IPsec on a Solaris host. The daemon negotiates with a remote host running the same protocol to provide authenticated keying materials for security associations in a protected manner. The daemon must be running on all hosts that plan to communicate securely. The IKE daemon is automatically loaded at boot time if the IKE configuration policy file `/etc/inet/ike/config` exists.

When the IKE daemon runs, the system authenticates itself to its peer IKE entity (Phase 1). The peer is defined in the IKE policy file, as are the authentication methods. The daemon then establishes the keys for the session (Phase 2). 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 via the `PF_KEY` socket. See the `pf_key`(7P) man page for more information.

Two programs support the IKE daemon. The `ikeadm`(1M) command enables the administrator to view IKE policy and modify it. The `ikecert`(1M) command enables the administrator to view and manage the public-key databases, `ike.privatekeys` and `publickeys`.

IKE Policy File

The IKE configuration policy file, `/etc/inet/ike/config`, provides the keying material for the IKE daemon itself, and for the IPsec SAs that it manages. The IKE daemon itself requires keying material in the Phase 1 exchange. Rules in the `ike/config` file establish the keying material. A valid rule in the policy file contains a label, identifies the hosts or networks that the keying material is for, and specifies the authentication method. See "IKE Tasks" on page 391 for examples of valid policy files. See the `ike.config`(4) man page for examples and descriptions of its parameters.

The IPsec SAs are used on the IP datagrams that are protected according to policies set up in the IPsec configuration policy file, `/etc/inet/ipsecinit.conf`. The IKE policy file determines if PFS is used when creating the IPsec SAs.

The security considerations for the `ike/config` file are similar to those for the `ipsecinit.conf` file. See "Security Considerations" on page 367 for details.

IKE Administration Command

The `ikeadm` command can check the syntax of the IKE configuration file, view aspects of the IKE daemon process, and change the parameters passed to the IKE daemon. The command can also gather statistics and debug IKE processes. See the `ikeadm`(1M) man page for examples and a full description of its options. The privilege level of the running IKE daemon determines what aspects of the IKE daemon can be viewed and modified. There are three levels of privilege.

- 0x0, or base level — At the base level of privilege, you cannot view or modify keying material. The base level is the default level at which the `in.iked` daemon runs.
- 0x1, or modkeys level — At the modkeys level of privilege, you can remove, change, and add pre-shared keys.

- 0x2, or keymat level — At the keymat level of privilege, you can view the actual keying material with the `ikeadm` command.

The security considerations for the `ikeadm` command are similar to those for the `ipseckey` command. See "Security Considerations" on page 368 for details.

Pre-Shared Keys Files

The `/etc/inet/secret/` directory contains the pre-shared keys for ISAKMP SAs and IPsec SAs. The `ike.preshared` file contains the pre-shared keys for ISAKMP SAs, and the `ipseckeys` file contains the pre-shared keys for IPsec SAs, when the administrator creates these keys manually. The `secret` directory is protected at `0700` and its files are protected at `0600`.

- The `ike.preshared` file is created by an administrator when the `ike.config` file requires pre-shared keys. The file contains keying material for ISAKMP SAs, that is, for IKE authentication. Because IKE uses the pre-shared keys to authenticate the Phase 1 exchange, the `ike.preshared` file must be valid before the `in.iked` daemon starts.
- The `ipseckeys` file contains keying material for IPsec SAs. For IPv6 hosts, the administrator manually creates and updates the keys in this file. See "IPsec Tasks" on page 372 for examples of manually managing the file. 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`(1M) command manipulates the local host's public-key databases. Because IKE uses these databases to authenticate the Phase 1 exchange when the `ike.config` file requires public key certificates, the directories must be populated before activating the `in.iked` daemon. Three subcommands handle each of the three databases: `certlocal`, `certdb`, and `certrldb`.

`ikecert certlocal` Command

The `certlocal` subcommand manages the private-key database in the `/etc/inet/secret/ike.privatekeys` directory. Options to the subcommand enable you to add, view, and remove private keys. The command also creates either a self-signed certificate or a certificate request. The `-ks` option creates a self-signed certificate, and the `-kc` option creates a certificate request.

Parameters that you pass to the `certlocal` subcommand when you create a private key must be reflected in the `ike.config` file, as shown in the following table.

TABLE 21–2 Correspondences Between `ike` `certlocal` and `ike.config` Values

`certlocal` **options**	`ike.config` **entry**	**Notes**
`-A` *Subject Alternate Name*	`cert_trust` *Subject Alternate Name*	A nickname that uniquely identifies the certificate. Possible values are IP address, email address, and domain name.
`-D` *X.509 Distinguished Name*	`cert_root` *X.509 Distinguished Name*	The full name of the certificate authority that includes Country, Organization name, Organizational Unit, and Common Name.
`-t dsa-sha1`	`auth_method dss_sig`	Slightly slower than RSA. Is not patented.
`-t rsa-md5` `-t rsa-sha1`	`auth_method rsa_sig`	Slightly faster than DSA. Patent expired in September 2000. The RSA public key must be large enough to encrypt the biggest payload, Typically, an identity payload, such as Distinguished Name, is the biggest.
`-t rsa-md5` `-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 your vendor. The vendor then creates keying material. You use the vendor's keying material as input to the `certdb` and `certrldb` subcommands.

`ikecert certdb` Command

The `certdb` subcommand manages the public-key database, `/etc/inet/ike/publickeys`. Options to the subcommand enable you to add, view, and remove public keys and certificates. The command accepts, as input, certificates that were generated by the `ikecert certlocal -ks` command on a communicating system. See "How to Configure IKE With Self-Signed Public Certificates" on page 397 for the procedure. The command also accepts the certificate that you receive from a PKI or CA as input. See "How to Configure IKE With Public Keys Signed by a Certificate Authority" on page 399 for the procedure.

ikecert certrldb Command

The certrldb subcommand manages the certificate revocation list (CRL) database, /etc/inet/ike/crls. The crls database maintains the revocation lists for public keys. Certificates that are no longer valid are on this list. When PKIs provide you with CRLs, you install them in the CRL database with the ikecert certrldb command. See "How to Update a Certificate Revocation List" on page 401 for the procedure.

/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 /etc/inet/ike directory is protected at 0755. The public-key databases that are stored under it are world-readable (0644). You use the ikecert certdb command to populate the directory.

The files 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 communicating system as input to the command. If you are using certificates from a PKI, you install two pieces of keying material from the vendor into this database — a certificate based on material you sent to the vendor, and a CA from the vendor.

An evaluation copy of the iPlanet CMS, a PKI, is available on the Media Kit in the installation package.

/etc/inet/secret/ike.privatekeys Directory

The ike.privatekeys directory holds private key files that are part of a public-private key pair, keying material for ISAKMP SAs. The directory is protected at 0700. The private key in this database must have a public key counterpart in the publickeys database.The ikecert certlocal command populates this directory. Private keys are not effective until their public key counterparts, self-signed certificates or CAs, are installed in the /etc/inet/ike/publickeys directory.

/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 vendors provide the CRLs for their certificates. You use the ikecert certrldb command to populate the database.

Implementing IKE Task Map

The `ikeadm`(1M), `ikecert`(1M), and `ike.config`(4) man pages contain useful procedures in their respective Examples sections.

TABLE 21–3 Implementing IKE Task Map

Task	Description	For Instructions, Go To …
Configure IKE with pre-shared keys	Involves creating a valid IKE policy file and `ike.preshared` file. IPsec files are also set up before booting the system to use the IKE-generated keys.	"How to Configure IKE With Pre-Shared Keys" on page 392
Refresh pre-shared keys on a running IKE system	Involves checking the IKE privilege level and editing the `ipseckeys` file with fresh keying material on communicating systems.	"How to Refresh Existing Pre-Shared Keys" on page 394
Add pre-shared keys to a running IKE system	Involves checking the IKE privilege level and running the `ikeadm` command with fresh keying material on communicating systems.	"How to Add New Pre-Shared Keys" on page 395
Configure IKE with self-signed public key certificates	Involves creating self–signed certificates with the `ikecert certlocal -ks` command, and adding the public key from a communicating system with the `ikecert certdb` command.	"How to Configure IKE With Self-Signed Public Certificates" on page 397
Configure IKE with a PKI Certificate Authority	Involves sending output from the `ikecert certlocal -kc` command to a PKI, and installing the public key, CA, and CRL from the vendor.	"How to Configure IKE With Public Keys Signed by a Certificate Authority" on page 399
Update the CA revocation lists	Involves installing a PKI vendor's CRL with the `ikecert certrldb` command.	"How to Update a Certificate Revocation List" on page 401

IKE Tasks

This section provides procedures that enable you to automatically manage the keys that secure traffic between two systems with IPv4 addresses. 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.

▼ How to Configure IKE With Pre-Shared Keys

1. **Become superuser on the system console.**

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

---

2. **On each system, create an** `/etc/inet/ike/config` **file with global parameters and rules that permit the IPsec policy in** `ipsecinit.conf` **to succeed. For example,**

```
### ike/config file on enigma, 192.168.66.1

## Global parameters
#
## Phase 1 transform defaults
p1_lifetime_secs 14400
p1_nonce_len 40
#
## Defaults that individual rules can override.
p1_xform { auth_method preshared oakley_group 5 auth_alg sha encr_alg 3des }
p2_pfs 2
#
## The rule to communicate with partym

{ label "Enigma-Partym"
  localid 192.168.66.1
  remoteid 192.168.55.2
  p1_xform
      { auth_method preshared  oakley_group 5  auth_alg md5  encr_alg des }
  p2_pfs 5
    }

### ike/config file on partym, 192.168.55.2
## Global Parameters
#
p1_lifetime_secs 14400
p1_nonce_len 40
#
p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg 3des }
p2_pfs 2

## The rule to communicate with enigma

{ label "Partym-Enigma"
  localid 192.168.55.2
  remoteid 192.168.66.1
  p1_xform
    { auth_method preshared  oakley_group 5  auth_alg md5  encr_alg des }
  p2_pfs 5
```

```
}
```

Note – These machine names are examples only. Use the names and addresses of your machines when securing traffic between them.

3. **On each machine, check the validity of the file:**

```
# /usr/lib/inet/in.iked -c -f /etc/inet/ike/config
```

4. **Generate random keys.**

   On a Solaris system, you can use the `od` command. For example,

```
# od -x </dev/random | head -4
0000000 df97 6d2f 4ef5 2c28 02d5 02aa f9de 481d
0000020 2ae8 b949 67e6 b9b0 dd16 e6d4 b7ea 7278
0000040 ac07 7cc6 99c1 7055 848a 3cf3 4377 980a
0000060 5ad7 5b40 b428 9f3a da20 7daa 65a4 83fe
```

5. **Create the file `/etc/inet/secret/ike.preshared` on each system and put the pre-shared key in each file.**

   The encryption algorithm in this example (see Step 2) is DES, so the pre-shared key must be at least 64 bits. However, a longer key length is a good idea. For example,

```
# ike.preshared on enigma, 192.168.66.1
{ localidtype IP
      localid 192.168.66.1
      remoteidtype IP
      remoteid 192.168.55.2
      # enigma and partym's shared key in hex (128 bits)
      key ac077cc699c17055848a3cf34377980a
    }

# ike.preshared on partym, 192.168.55.2
{ localidtype IP
      localid 192.168.55.2
      remoteidtype IP
      remoteid 192.168.66.1
      # partym and enigma's shared key in hex (128 bits)
      key ac077cc699c17055848a3cf34377980a
    }
```

Note – The pre-shared keys must be identical.

6. **On each system, add the address and host name for the other system in the `/etc/hosts` file. For example,**

   On a system named partym:

```
# Secure communication with enigma
192.168.66.1 enigma
```

On a system named enigma:

```
# Secure communication with partym
192.168.55.2  partym
```

7. **On each system, edit the** `/etc/inet/ipsecinit.conf` **file by adding the following lines:**

   On enigma:

   ```
   {laddr enigma raddr partym} ipsec {auth_algs any sa shared}
   ```

   On partym:

   ```
   {laddr partym raddr enigma} ipsec {auth_algs any sa shared}
   ```

8. **Enable secure communication by rebooting each system.**

   ```
   # /usr/sbin/reboot
   ```

▼ How to Refresh Existing Pre-Shared Keys

This procedure assumes that you want to replace an existing pre-shared key. If you use a strong encryption algorithm, such 3DES, AES, or Blowfish, you may be able to schedule key replacement for when you reboot both machines. This procedure is for machines using an algorithm like DES to secure traffic.

1. **Become superuser on the system console.**

   Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

   ---

2. **Generate random keys and choose one.**

   On a Solaris system, you can use the `od` command.

   ```
   # od -x </dev/random | head -2
   0000000 305e c563 69ca 62c2 ae80 4690 c571 3e18
   0000020 be43 9533 d50f ec49 c7fe cf3c 8f13 91c0
   ```

3. **Edit the** `/etc/inet/secret/ike.preshared` **file on each system, and replace the current key with a new key.**

   For example, on the hosts enigma and partym, you would replace the value of `key` with a new number, like `be439533d50fec49c7fecf3c8f1391c0`.

4. **Check that the** `in.iked` **daemon permits you to change keying material.**

   ```
   # /usr/sbin/ikeadm get priv
   Current privilege level is 0x2, access to keying material enabled
   ```

You can change keying material if the command returns a privilege level of 0x1 or 0x2. Level 0x0 does not permit keying material operations. By default, the `in.iked` daemon runs at the 0x0 level of privilege.

5. **If the `in.iked` daemon permits you to change keying material, read in the new version of the `ike.preshared` file.**

   For example,

   ```
   # ikeadm read preshared
   ```

6. **If the `in.iked` daemon does not permit you to change keying material, kill the daemon and then restart it.**

   When the daemon starts, it reads the new version of the `ike.preshared` file.

   For example,

   ```
   # pkill in.iked
   # /usr/lib/inet/in.iked
   ```

▼ How to Add New Pre-Shared Keys

On a system running the `in.iked` daemon, you can add pre-shared keys for interfaces that you have added to the `ipsecinit.conf` file after the daemon was invoked. This procedure assumes that you have already added the new interfaces to the `/etc/hosts` file and the `/etc/inet/ipsecinit.conf` file on both systems, and that you have not yet read the `ipsecinit.conf` file into each system.

1. **Become superuser on the system console.**

   Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

   ---

2. **Check that the `in.iked` daemon permits you to change keying material.**

   ```
   # /usr/sbin/ikeadm get priv
   Current privilege level is 0x2, access to keying material enabled
   ```

   You can change keying material if the command returns a privilege level of 0x1 or 0x2. Level 0x0 does not permit keying material operations. By default, the `in.iked` daemon runs at the 0x0 level of privilege.

3. **If the `in.iked` daemon does not permit you to change keying material, kill the daemon and then start it with the correct privilege level.**

   For example,

   ```
   # pkill in.iked
   # /usr/lib/inet/in.iked -p 2
   ```

```
Setting privilege level to 2!
```

4. **Generate random keys and choose one.**

   On a Solaris system, you can use the `od` command.

```
# od -x </dev/random | head -2
0000000 2d86 b6f6 eb7a e8a9 3d83 58b2 cd17 4164
0000020 8be4 fea4 b456 933a 46dd 149a 0a10 b2e4
```

5. **Type the `ikeadm` command on each system to add the new keying material.**

   For example, if you are on enigma, and add the key for host nemesis, 192.163.55.8:

```
# ikeadm
ikeadm> add preshared { localidtype ip localid 192.168.66.1
remoteidtype ip remoteid 192.163.55.8 ike_mode main
key 2d86b6f6eb7ae8a93d8358b2cd174164 }
ikeadm: Successfully created new preshared key.
```

   On host nemesis, the administrator would add the identical key, as in:

```
# ikeadm
ikeadm> add preshared { localidtype ip localid 192.163.55.8
remoteidtype ip remoteid 192.168.66.1 ike_mode main
key 2d86b6f6eb7ae8a93d8358b2cd174164 }
ikeadm: Successfully created new preshared key.
```

Note – A message of the form `Error: invalid preshared key definition` indicates that you have mistyped or omitted a parameter to the `add preshared` command. Retype the command correctly to add the key.

---

6. **Exit the `ikeadm` command mode.**

```
ikeadm> exit
#
```

7. **On each system, lower the privilege level of the `in.iked` daemon.**

```
# ikeadm set priv base
```

8. **On each system, activate the `ipsecinit.conf` file to secure the added interfaces.**

```
# ipsecconf -a /etc/inet/ipsecinit.conf
```

Note – Read the warning when you execute the command. A socket that is already in use (latched) provides an unsecured back door into the system.

▼ How to Configure IKE With Self-Signed Public Certificates

1. **Become superuser on the system console.**

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

2. **Use the** `ikecert certlocal -ks` **command to add a self-signed certificate to the** `ike.privatekeys` **database. For example,**

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
-D "C=US, O=ExampleCompany, OU=US-Example, CN=Example" \
-A IP=192.168.10.242
Generating, please wait...
Certificate:
Certificate generated.
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
MIICLTCCAZagAwIBAgIBATANBgkqhkiG9w0BAQQFADBNMQswCQYDVQQGEwJVUzEX
...
6sKTxpg4GP3GkQGcd0r1rhW/3yaWBkDwOdFCqEUyffzU
-----END X509 CERTIFICATE-----
```

3. **Send the certificate to the communicating system's administrator.**

   You can cut-and-paste the certificate into an email, as in:

```
To: root@us.example.com
From: root@un.example.com
Message: -----BEGIN X509 CERTIFICATE-----
MIICLTCCAZagAwIBAgIBATANBgkqhkiG9w0BAQQFADBNMQswCQYDVQQGEwJVUzEX
...
6sKTxpg4GP3GkQGcd0r1rhW/3yaWBkDwOdFCqEUyffzU
-----END X509 CERTIFICATE-----
```

4. **Edit the** `/etc/inet/ike/config` **file to recognize the public keys from a communicating system. For example,**

```
# Explicitly trust the following self-signed certs
# Use the Subject Alternate Name to identify the cert

cert_trust "192.168.10.242"
```

```
cert_trust "192.168.11.241"

## Parameters that may also show up in rules.

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

{
 label "UN-Example to US-Example"
 local_id_type dn
 local_id "C=US, O=ExampleCompany, OU=UN-Example, CN=Example"
 remote_id_type dn
 remote_id "C=US, O=ExampleCompany, OU=US-Example, CN=Example"

 local_addr 192.168.10.242
 remote_addr 192.168.11.241

 p1_xform
  { auth_method rsa_encrypt oakley_group 2  auth_alg md5  encr_alg des }
}
```

5. **Do the following substeps to add the communicating system's public key.**

   a. **Copy the public key from the administrator's email.**

   b. **Type the** `ikecert certdb -a` **command and type <Return>.**

      ```
      # ikecert certdb -a <Return>
      ```

   c. **Paste the public key and type <Return>.**

      ```
      -----BEGIN X509 CERTIFICATE-----
      MIICL...
      ...
      KgDid/nxWPlWQU5vMAiwJXfa0sw/A12w448JVkVmEWaf
      -----END X509 CERTIFICATE----- <Return>
      ```

   d. **End the entry by typing <Control-D>.**

      ```
      <Control-D>
      ```

6. **Verify with the other administrator that the keys have not been tampered with.**

   For example, you can phone the other administrator to compare the values of the public key hash, as in,

```
# ikecert certdb -l
        Certificate Slot Name: 0   Type: if-modn
        Subject Name: <C=US, O=ExampleCo, OU=UN-Example, CN=Example>
        Key Size: 1024
        Public key hash: 2239A6A127F88EE0CB40F7C24A65B818

other system # ikecert certlocal -l
Local ID Slot Name: 1   Type: if-modn
        Key Size: 1024
```

```
Public key hash: 2239A6A127F88EE0CB40F7C24A65B818
```

Note – The public key hash above is different from the one that your systems generate.

▼ How to Configure IKE With Public Keys Signed by a Certificate Authority

1. **Become superuser on the system console.**

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

2. **Use the** `ikecert certlocal -kc` **command to add a trusted root certificate to the** `ike.privatekeys` **database.**

   For example,

```
# ikecert certlocal -kc -m 1024 -t rsa-md5 \
-D "C=US, O=ExampleCompany\, Inc., OU=US-Example, CN=Example" \
-A "DN=C=US, O=ExampleCompany\, Inc., OU=US-Example"
Generating, please wait...
Certificate request generated.
-----BEGIN CERTIFICATE REQUEST-----
MIIByjCCATMCAQAwUzELMAkGA1UEBhMCVVMxHTAbBgNVBAoTFEV4YW1wbGVDb21w
...
lcM+tw0ThRrfuJX9t/Qa1R/KxRlMA3zckO80mO9X
-----END CERTIFICATE REQUEST-----
```

3. **Submit the request to the outside Certificate Authority or PKI.**

   The vendor issues you two certificates and a CRL that you enter into the respective databases.

   - Your publickeys certificate – This certificate is based on the request you submitted to the vendor. The certificate uniquely identifies you.
   - A Certificate Authority – This is the vendor's signature. The CA verifies that your publickeys certificate is legitimate.
   - A Certificate Revocation List – This is the latest list of certificates that the vendor has revoked.

4. **Enter the three certificates as the argument to** `ikecert` **commands.**

   a. **Become superuser on the system console.**

b. Type the `ikecert certdb -a` **command and type <Return>.**

```
# ikecert certdb -a <Return>
```

c. Paste your certificate that you received from the vendor and type <Return>.

```
-----BEGIN X509 CERTIFICATE-----
...
-----END X509 CERTIFICATE-----<Return>
```

d. End the entry by typing <Control-D>.

```
<Control-D>
```

e. Type the `ikecert certdb -a` **command and type <Return>.**

```
# ikecert certdb -a <Return>
```

f. Paste the vendor's CA and type <Return>, then <Control-D> to end the entry.

```
-----BEGIN X509 CERTIFICATE-----
...
-----END X509 CERTIFICATE-----<Return>
<Control-D>
```

g. Type the `ikecert certrldb -a` **command and type <Return>.**

```
# ikecert certrldb -a <Return>
```

h. Paste the vendor's CRL and type <Return>, then <Control-D> to end the entry.

5. Edit the `/etc/inet/ike/config` **file to recognize the vendor.**

Use the name that the vendor tells you to use. For example,

```
# Trusted root cert
# This certificate is from Example PKI
# This is the X.509 distinguished name for the CA that it issues.

cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"

## Parameters that may also show up in rules.

p1_xform { auth_method rsa_sig oakley_group 1 auth_alg sha1 encr_alg 3des }
p2_pfs 2

{
 label "UN-Example to US-Example - Example PKI"
 local_id_type dn
 local_id "C=US, O=ExampleCompany, OU=UN-Example, CN=Example"
 remote_id_type dn
 remote_id "C=US, O=ExampleCompany, OU=US-Example, CN=Example"

 local_addr 192.168.10.242
 remote_addr 192.168.11.241
```

```
 p1_xform
  { auth_method rsa_encrypt oakley_group 2  auth_alg md5  encr_alg des }
}
```

6. **The communicating system does the same operations as you have done.**

   Following the example, the "C=US, O=ExampleCompany, OU=US-Example, CN=Example" system runs the `ikecert` commands as above. Its `ike.config` file uses its local information for local parameters, and your system's information for the remote parameters.

   For example,

```
# Trusted root cert
# This certificate is from Example PKI

cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"

## Parameters that may also show up in rules.

p1_xform { auth_method rsa_sig oakley_group 1 auth_alg sha1 encr_alg 3des }
p2_pfs 2

{
 label "US-Example to UN-Example - Example PKI"
 local_id_type dn
 local_id "C=US, O=ExampleCompany, OU=US-Example, CN=Example"
 remote_id_type dn
 remote_id "C=US, O=ExampleCompany, OU=UN-Example, CN=Example"

 local_addr 192.168.11.241
 remote_addr 192.168.10.242

 p1_xform
  { auth_method rsa_sig oakley_group 2  auth_alg md5  encr_alg des }
}
```

   When the `/etc/hosts` file and `/etc/inet/ipsecinit.conf` file have been modified to include the protected interfaces and the machines are rebooted, the IKE daemon authenticates itself with the public keys and the CA.

Note – The RSA encryption authentication method hides identities in IKE from eavesdroppers, so IKE does not retrieve the peer's certificate. As a result, the method requires that the IKE peers know each other's public keys. Therefore, when you use `auth_method rsa_encrypt` in the `ike.config` file, you must add the peer's certificate to the public-keys database.

▼ How to Update a Certificate Revocation List

1. **Become superuser on the system console.**

Note – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the total security of the system is reduced to the security of the remote login session.

2. **Follow the instructions from the vendor about how to extract the revoked certificates.**
3. **Use the following procedure to add the revoked certificate to the CRL database.**
   a. **Type the** `ikecert certrldb -a` **command and type <Return>.**

      ```
      # ikecert certrldb -a <Return>
      ```

   b. **Paste the revoked certificate from the PKI vendor and type <Return>, then <Control-D> to end the entry.**
4. **Repeat for every CRL in the revocation list.**

CHAPTER 22

Mobile IP Topics

Chapter 23	Provides overview information for Mobile IP
Chapter 24	Provides step-by-step instructions for configuring Mobile IP
Chapter 25	Provides reference information for Mobile IP

CHAPTER 23

Mobile IP (Overview)

Mobile Internet Protocol (IP) enables the transfer of information between mobile computers. Mobile computers include laptops and wireless communications. The mobile computer can change its location to a foreign network. At the foreign network, the mobile computer can still communicate through the home network of the mobile computer. The Solaris implementation of Mobile IP supports only IPv4.

This chapter contains the following information:

Introduction

Current versions of the Internet Protocol (IP) assume that the point at which a computer attaches to the Internet or a network is fixed. IP also assumes that the IP address of the computer identifies the network to which the computer is attached. Datagrams that are sent to a computer are based on the location information that is contained in the IP address. Many Internet Protocols that are in use require that a node's IP address remain unchanged. If any of these applications are active on a Mobile IP computing device, the applications fail. Even HTTP would fail if not for the short-lived nature of its TCP connections. Updating an IP address and refreshing the Web page is not a burden.

If a mobile computer, or *mobile node*, moves to a new network while its IP address is unchanged, the mobile node address does not reflect the new point of attachment. Consequently, routing protocols that exist cannot route datagrams to the mobile node correctly. You must reconfigure the mobile node with a different IP address that represents the new location. Assigning a different IP address is cumbersome. Thus, under the current Internet Protocol, if the mobile node moves without changing its address, it loses routing. If the mobile node does change its address, it loses connections.

Mobile IP solves this problem by allowing the mobile node to use two IP addresses. The first address is a fixed *home address*. The second address is a *care-of address* that changes at each new point of attachment. Mobile IP enables a computer to roam freely on the Internet. Mobile IP also enables a computer to roam freely on an organization's network while still maintaining the same home address. Consequently, communication activities are not disrupted when the user changes the computer's point of attachment. Instead, the network is updated with the new location of the mobile node. See the Glossary for definitions of terms that are associated with Mobile IP.

The following figure illustrates the general Mobile IP topology.

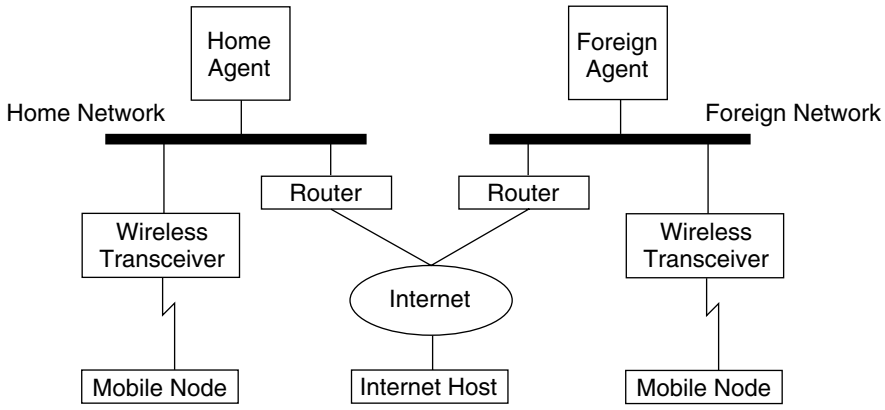


FIGURE 23–1 Mobile IP Topology

By using the previous illustration's Mobile IP topology, the following scenario shows how a datagram moves from one point to another within the Mobile IP framework.

1. The Internet host sends a datagram to the mobile node by using the mobile node's home address (normal IP routing process).
2. If the mobile node is on its home network, the datagram is delivered through the normal IP process to the mobile node. Otherwise, the home agent receives the datagram.

3. If the mobile node is on a foreign network, the home agent forwards the datagram to the foreign agent. The home agent must encapsulate the datagram in an IP-in-IP manner so that the foreign agent's IP address appears in the outer IP header.
4. The foreign agent delivers the datagram to the mobile node.
5. Datagrams from the mobile node to the Internet host are sent by using normal IP routing procedures. If the mobile node is on a foreign network, the packets are delivered to the foreign agent. The foreign agent forwards the datagram to the Internet host.
6. In situations with ingress filtering present, the source address must be topologically correct for the subnet that the datagram is coming from, or a router cannot forward the datagram. If this scenario is the situation on links between the mobile node and the correspondent node, the foreign agent needs to provide reverse tunneling support. Then the foreign agent can deliver every datagram that the mobile node sends to its home agent. The home agent then forwards the datagram through the path that the datagram would have taken had the mobile node resided on the home network. This process guarantees that the source address is correct for all links that the datagram must traverse.

In the instance of wireless communications, the illustrations depict the use of wireless transceivers to transmit the datagrams to the mobile node. Also, all datagrams between the Internet host and the mobile node use the home address of the mobile node. The home address is used even when the mobile node is located on the foreign network. The care-of address is used only for communication with mobility agents. The care-of address is invisible to the Internet host.

Mobile IP Functional Entities

Mobile IP introduces the following new functional entities:

- **Mobile Node (MN)** – Host or router that changes its point of attachment from one network to another network while maintaining all existing communications by using its IP home address.
- **Home Agent (HA)** – Router or server on the home network of a mobile node. The router intercepts datagrams that are destined for the mobile node. The router then delivers the datagrams through the care-of address. The home agent also maintains current information on the location of the mobile node.
- **Foreign Agent (FA)** – Router or server on the foreign network that the mobile node visits. Provides host routing services to the mobile node. The foreign agent might also provide a care-of address to the mobile node while the mobile node is registered.

How Mobile IP Works

Mobile IP enables routing of IP datagrams to mobile nodes. The home address of the mobile node always identifies the mobile node regardless of where the mobile node is attached. When away from home, a care-of address is associated with the mobile node's home address. The care-of address provides information about the current point of attachment of the mobile node. Mobile IP uses a registration mechanism to register the care-of address with a home agent.

The home agent redirects datagrams from the home network to the care-of address. The home agent constructs a new IP header that contains the care-of address of the mobile node as the destination IP address. This new header encapsulates the original IP datagram. Consequently, the home address of the mobile node has no effect on the routing of the encapsulated datagram until the datagram arrives at the care-of address. This type of encapsulation is also called *tunneling*. After the datagram arrives at the care-of address, the datagram is de-encapsulated. Then the datagram is delivered to the mobile node.

The following illustration shows a mobile node that resides on its home network, Network A, before the mobile node moves to a foreign network, Network B. Both

networks support Mobile IP. The mobile node is always associated with the home address of the mobile node, `128.226.3.30`.

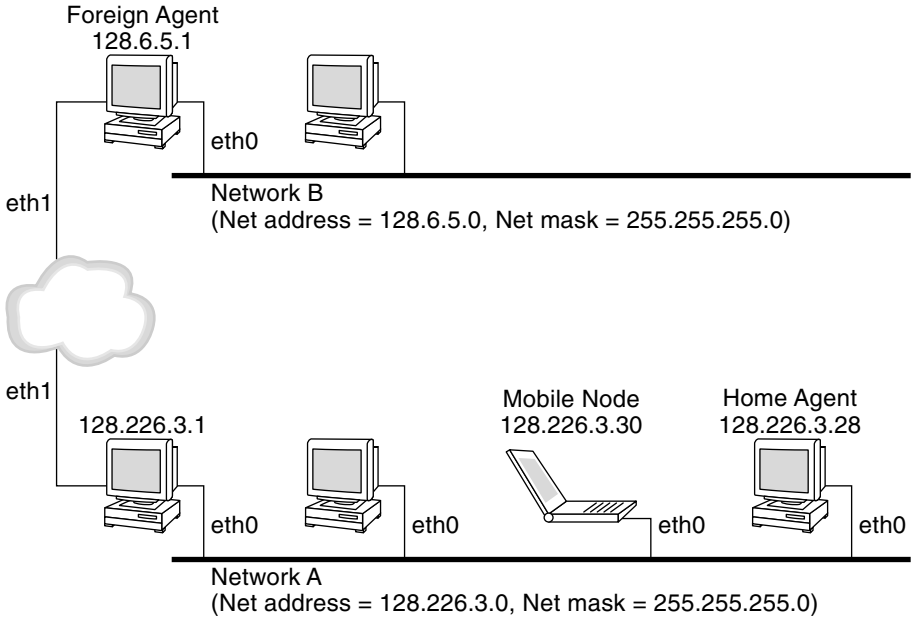


FIGURE 23–2 Mobile Node Residing on Home Network

The following illustration shows a mobile node that has moved to a foreign network, Network B. Datagrams that are destined for the mobile node are intercepted by the home agent on the home network, Network A. The datagrams are encapsulated. Then the datagrams are sent to the foreign agent on Network B. The foreign agent strips off the outer header. Then the foreign agent delivers the datagram to the mobile node that is located on Network B.

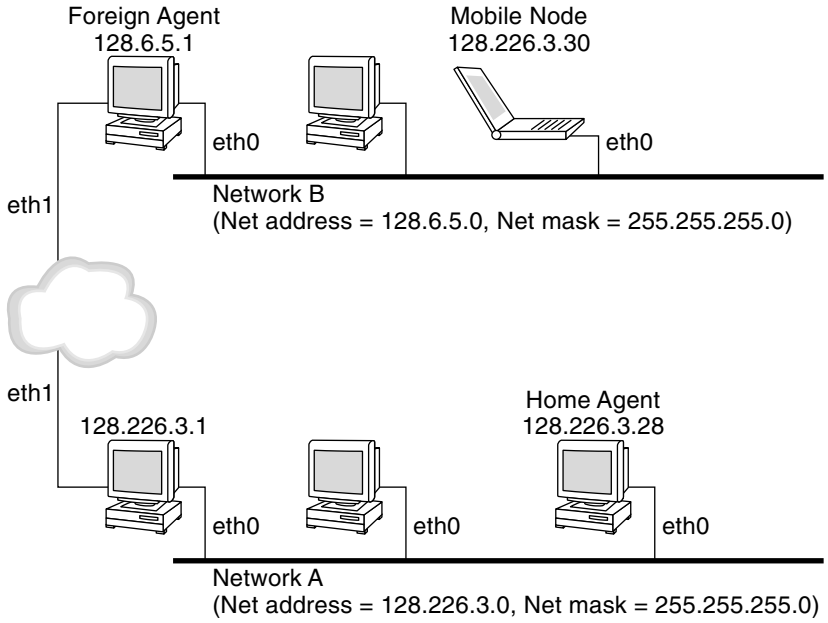


FIGURE 23–3 Mobile Node Moving to a Foreign Network

The care-of address might belong to a foreign agent. The care-of address might be acquired by the mobile node through Dynamic Host Configuration Protocol (DHCP) or Point-to-Point Protocol (PPP). In the latter situation, a mobile node has a co-located care-of address.

Mobility agents (home agents and foreign agents) advertise their presence by using *agent advertisement* messages. Optionally, a mobile node can solicit an agent advertisement message. The mobile node uses any mobility agent that is attached locally through an *agent solicitation* message. A mobile node uses the agent advertisements to determine whether the mobile node is on the home network or a foreign network.

The mobile node uses a special *registration* process to inform the home agent about the current location of the mobile node. The mobile node is always "listening" for mobility agents advertising their presence. The mobile node uses these advertisements to help determine when the mobile node moves to another subnet. When a mobile node determines that the mobile node has moved its location, the mobile node uses the new foreign agent to forward a registration message to the home agent. The mobile node uses the same process when the mobile node moves from one foreign network to another foreign network.

When the mobile node detects that it is located on the home network, the mobile node does not use mobility services. When the mobile node returns to the home network, the mobile node *deregisters* with the home agent.

Agent Discovery

A mobile node uses a method that is known as agent discovery to determine the following information:

- When the node has moved from one network to another
- Whether the network is the home network or a foreign network
- The foreign agent care-of address that is offered by each foreign agent on that network
- Mobility services that are provided by the mobility agent, advertised as flags, and additional extensions in the agent advertisement

Mobility agents transmit *agent advertisements* to advertise services on a network. In the absence of agent advertisements, a mobile node can solicit advertisements. This capability is known as *agent solicitation*. If a mobile node is capable of supporting its own co-located care-of address, the mobile node can use regular router advertisements for the same purposes. See the next section.

Agent Advertisement

Mobile nodes use agent advertisements to determine the current point of attachment to the Internet or to an organization's network. An agent advertisement is an Internet Control Message Protocol (ICMP) router advertisement that has been extended to carry also a mobility agent advertisement extension.

A foreign agent (FA) can be too busy to serve additional mobile nodes. However, a foreign agent must continue to send agent advertisements. Then the mobile node, which is already registered with a foreign agent, knows that the mobile node has not moved out of range of the foreign agent. The mobile node also knows that the foreign agent has not failed. A mobile node that is registered with a foreign agent from which it no longer receives agent advertisements, probably knows that the mobile node can no longer contact that foreign agent.

Agent Advertisement Over Dynamic Interfaces

You can configure the implementation of the foreign agent to send advertisements over dynamically created interfaces. You have options to enable or disable limited unsolicited advertisements over the advertising interfaces. Dynamically created interfaces are defined as only those interfaces that are configured after the mipagent starts. Advertisement over dynamic interfaces is useful for applications that are to support transient mobility interfaces. Moreover, by limiting unsolicited advertisement, network bandwidth might be saved.

Agent Solicitation

Every mobile node should implement agent solicitation. The mobile node uses the same procedures, defaults, and constants for agent solicitation that are specified for solicitation messages of ICMP routers.

The rate that a mobile node sends solicitations is limited by the mobile node. The mobile node can send three initial solicitations at a maximum rate of one per second while the mobile node searches for an agent. After the mobile node registers with an agent, the rate that solicitations are sent is reduced to limit the overhead on the local network.

Care-of Addresses

Mobile IP provides the following alternative modes for the acquisition of a care-of address:

- A foreign agent provides a *foreign agent care-of address*, which is advertised to the mobile node through agent advertisement messages. The care-of address is usually the IP address of the foreign agent that sends the advertisements. The foreign agent is the endpoint of the tunnel. When the foreign agent receives datagrams through a tunnel, the foreign agent de-encapsulates the datagrams. Then the foreign agent delivers the inner datagram to the mobile node. Consequently, many mobile nodes can share the same care-of address. Bandwidth is important on wireless links. Wireless links are good candidates from which foreign agents can provide Mobile IP services to higher-bandwidth wired links.
- A mobile node acquires a *co-located care-of address* as a local IP address through some external means. The mobile node then associates with one of its own network interfaces. The mobile node might acquire the address through DHCP as a temporary address. The address might also be owned by the mobile node as a long-term address. However, the mobile node can only use the address while

visiting the subnet to which this care-of address belongs. When using a co-located care-of address, the mobile node serves as the endpoint of the tunnel. The mobile node performs de-encapsulation of the datagrams that are tunneled to the mobile node.

A Co-located care-of address enables a mobile node to function without a foreign agent. Consequently, a mobile node can use a co-located care-of address in networks that have not deployed a foreign agent.

If a mobile node is using a co-located care-of address, the mobile node must be located on the link that is identified by the network prefix of the care-of address. Otherwise, datagrams that are destined to the care-of address cannot be delivered.

Mobile IP With Reverse Tunneling

The previous description of Mobile IP assumes that the routing within the Internet is independent of the source address of the datagram. However, intermediate routers might check for a topologically correct source address. If an intermediate router does check, the mobile node needs to set up a reverse tunnel. By setting up a reverse tunnel from the care-of address to the home agent, you ensure a topologically correct source address for the IP data packet. Reverse tunnel support is advertised by foreign agents and home agents. A mobile node can request a *reverse tunnel* between the foreign agent and the home agent when the mobile node registers. A reverse tunnel is a tunnel that starts at the care-of address of the mobile node and terminates at the home agent. The following illustration shows the Mobile IP topology that uses a reverse tunnel.

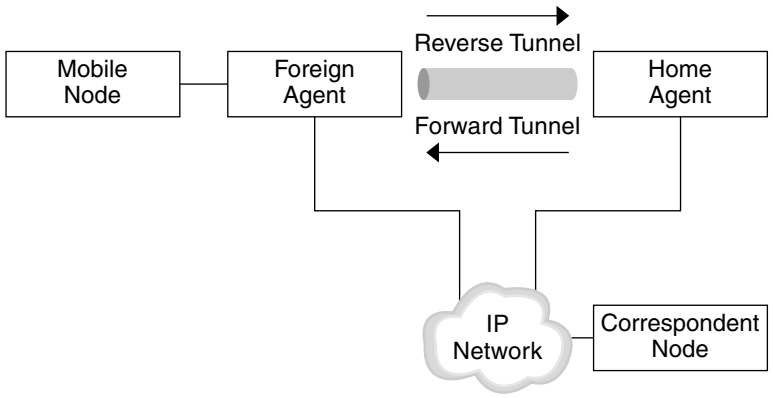


FIGURE 23–4 Mobile IP With a Reverse Tunnel

Limited Private Addresses Support

Mobile nodes that have private addresses that are not globally routable through the Internet require reverse tunnels. Solaris Mobile IP supports mobile nodes that are privately addressed. See “Overview of the Solaris Mobile IP Implementation” on page 441 for the functions that Solaris Mobile IP does not support.

Enterprises employ private addresses when external connectivity is not required. Private addresses are not routable through the Internet. When a mobile node has a private address, the mobile node can only communicate with a correspondent node by having its datagrams reverse-tunneled to its home agent. The home agent then delivers the datagram to the correspondent node in whatever manner the datagram is normally delivered when the mobile node is at home. The following illustration shows a network topology with two mobile nodes that are privately addressed. The two mobile nodes use the same care-of address when registered to the same foreign agent.

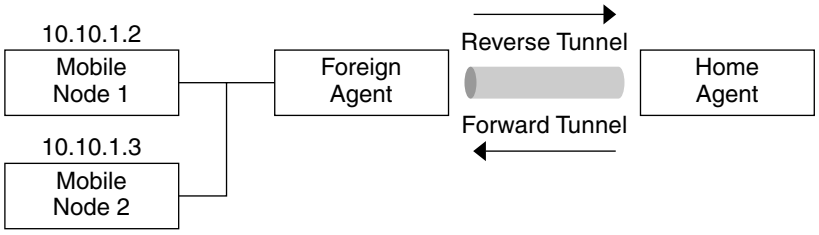


FIGURE 23–5 Privately Addressed Mobile Nodes Residing on the Same Foreign Network

The care-of address and the home agent address must be globally routable addresses if these addresses belong to different domains that are connected by a public Internet.

The same foreign network can include two mobile nodes that are privately addressed with the same IP address. However, each mobile node must have a different home agent. Also, each mobile node must be on different advertising subnets of a single foreign agent. The following illustration shows a network topology that depicts this situation.

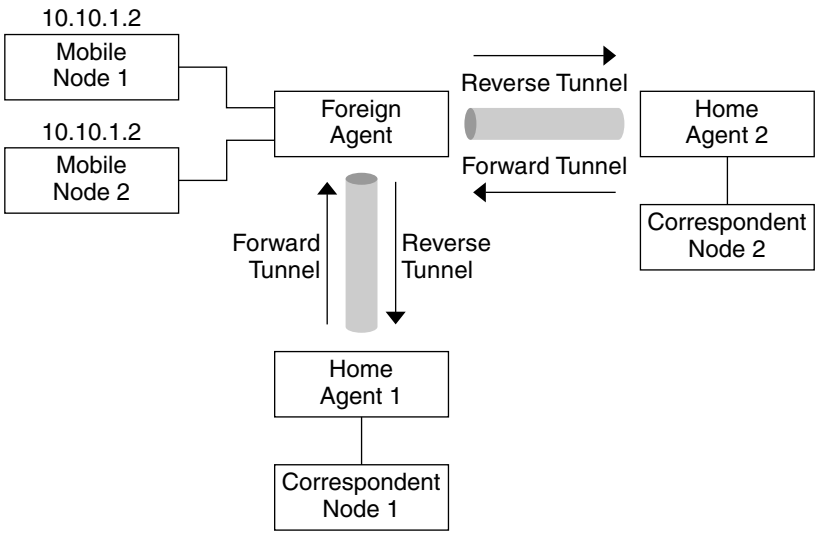


FIGURE 23–6 Privately Addressed Mobile Nodes Residing on Different Foreign Networks

Mobile IP Registration

Mobile nodes detect when they have moved from one subnet to another through the use of agent advertisements. When the mobile node receives an agent advertisement that indicates that the mobile node has changed locations, the mobile node registers through a foreign agent. Even though the mobile node might have acquired its own co-located care-of address, this feature is provided to enable sites to restrict access to mobility services.

Mobile IP registration provides a flexible mechanism for mobile nodes to communicate the current reachability information to the home agent. The registration process enables mobile nodes to perform the following tasks:

- Request forwarding services when visiting a foreign network
- Inform the home agent of the current care-of address
- Renew a registration that is about to expire
- Deregister when the mobile node returns home
- Request a reverse tunnel

Registration messages exchange information between a mobile node, a foreign agent, and the home agent. Registration creates or modifies a mobility binding at the home agent. Registration associates the home address of the mobile node with the care-of address of the mobile node for the specified lifetime.

The registration process also enables mobile nodes to do the following functions:

- Register with multiple foreign agents
- Deregister specific care-of addresses while retaining other mobility bindings
- Discover the address of a home agent if the mobile node is not configured with this information

Mobile IP defines the following registration processes for a mobile node:

- If a mobile node registers a foreign agent care-of address, the mobile node is informing the home agent that it is reachable through that foreign agent.
- If a mobile node receives an agent advertisement that requires the mobile node to register through a foreign agent, the mobile node can still attempt to obtain a co-located care-of address. The mobile node can also register with that foreign agent or any other foreign agent on that link.
- If a mobile node uses a co-located care-of address, the mobile node registers directly with the home agent.
- If a mobile node returns to the home network, the mobile node deregisters with the home agent.

These registration processes involve the exchange of registration requests and registration reply messages. When the mobile node registers by using a foreign agent, the registration process takes the following steps, which the subsequent illustration shows:

1. The mobile node sends a registration request to the prospective foreign agent to begin the registration process.
2. The foreign agent processes the registration request and then relays the request to the home agent.
3. The home agent sends a registration reply to the foreign agent to grant or deny the request.
4. The foreign agent processes the registration reply and then relays the reply to the mobile node to inform the mobile node of the disposition of the request.

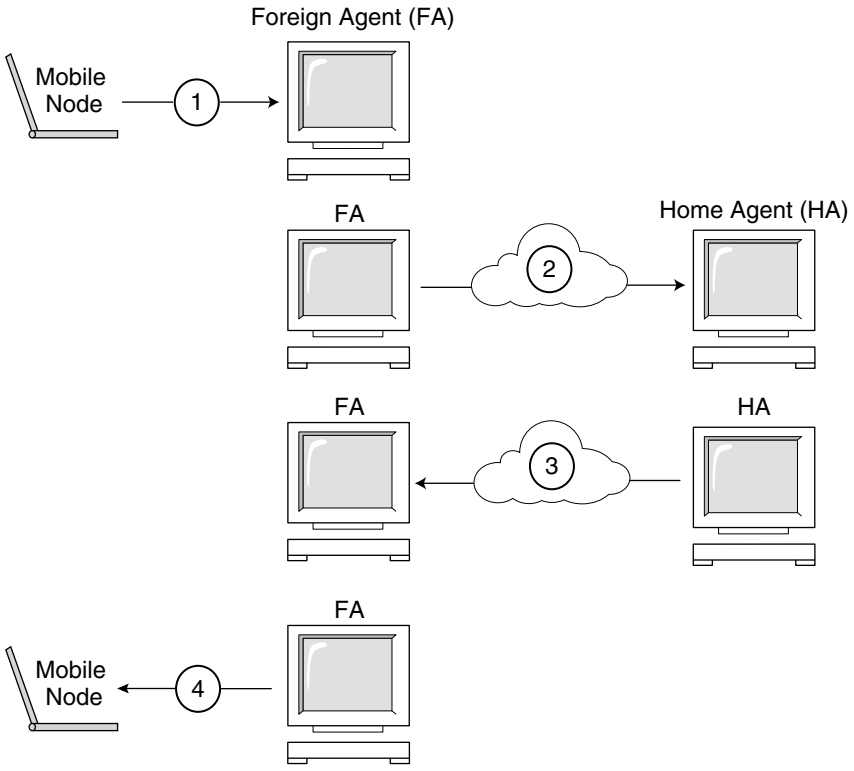


FIGURE 23–7 Mobile IP Registration Process

When the mobile node registers directly with the home agent, the registration process requires only the following steps:

- The mobile node sends a registration request to the home agent.
- The home agent sends a registration reply to the mobile node that grants or denies the request.

Also, either the foreign agent or the home agent might require a reverse tunnel. If the foreign agent supports reverse tunneling, the mobile node uses the registration process to request a reverse tunnel. The mobile node sets the reverse tunnel flag in the registration request to request a reverse tunnel.

Network Access Identifier (NAI)

AAA servers, in use within the Internet, provide authentication and authorization services for dial-up computers. These services are likely to be equally valuable for mobile nodes that use Mobile IP when the nodes attempt to connect to foreign

domains with AAA servers. AAA servers use Network Access Identifier (NAI) to identify clients. A mobile node can identify itself by including the NAI in the Mobile IP registration request.

Because the NAI is typically used to identify the mobile node uniquely, the home address of the mobile node is not always necessary to provide that function. Thus, a mobile node can authenticate itself. Consequently, a mobile node can be authorized for connection to the foreign domain without even having a home address. To request that a home address be assigned, a message that contains the mobile node NAI extension can set the home address field to zero in the registration request.

Mobile IP Message Authentication

Each mobile node, foreign agent, and home agent supports a mobility security association between the various Mobile IP components. The security association is indexed by the security parameter index (SPI) and IP address. In the instance of the mobile node, this address is the home address of the mobile node. Registration messages between a mobile node and the home agent are authenticated with the Mobile-home authentication extension. In addition to Mobile-home authentication, which is mandatory, you can use the optional Mobile-foreign agent and Home-foreign agent authentications.

Mobile Node Registration Request

A mobile node uses a *registration request* message to register with the home agent. Thus, the home agent can create or modify a mobility binding for that mobile node (for example, with a new lifetime). The foreign agent can relay the registration request to the home agent. However, if the mobile node is registering a co-located care-of address, then the mobile node can send the registration request directly to the home agent. If the foreign agent advertises that registration messages must be sent to the foreign agent, then the mobile node must send the registration request to the foreign agent.

Registration Reply Message

A mobility agent returns a *registration reply* message to a mobile node that has sent a registration request message. If the mobile node requests service from a foreign agent, that foreign agent receives the reply from the home agent. Subsequently, the foreign agent relays the reply to the mobile node. The reply message contains the necessary codes to inform the mobile node and the foreign agent about the status of the registration request. The message also contains the lifetime that is granted by the

home agent. The lifetime can be smaller than the original request. The registration reply can also contain a dynamic home address assignment.

Foreign Agent Considerations

The foreign agent plays a mostly passive role in Mobile IP registration. The foreign agent adds all mobile nodes that are registered to the visitor table. The foreign agent relays registration requests between mobile nodes and home agents. Also, when the foreign agent provides the care-of address, the foreign agent de-encapsulates datagrams for delivery to the mobile node. The foreign agent also sends periodic agent advertisement messages to advertise the presence of the foreign agent.

If home and foreign agents support reverse tunnels, and the mobile node requests a reverse tunnel, the foreign agent then tunnels all the packets from the mobile node to the home agent. The home agent then sends the packets to the correspondent node. This process is the reverse of the home agent tunneling all of the mobile node's packets to the foreign agent for delivery to the mobile node. A foreign agent that supports reverse tunnels advertises that the reverse tunnel is supported for registration. Because of the local policy, the foreign agent can deny a registration request when the reverse tunnel flag is not set. The foreign agent can only distinguish multiple mobile nodes with the same (private) IP address when these mobile nodes are visiting different interfaces on the foreign agent. In the forward tunnel situation, the foreign agent distinguishes between multiple mobile nodes that share the same private addresses by looking at the incoming tunnel interface. The incoming tunnel interface maps to a unique home agent address.

Home Agent Considerations

Home agents play an active role in the registration process. The home agent receives registration requests from the mobile node. The registration request might be relayed by the foreign agent. The home agent updates its record of the mobility bindings for this mobile node. The home agent issues a suitable registration reply in response to each registration request. The home agent also forwards packets to the mobile node when the mobile node is away from the home network.

A home agent might not have to have a physical subnet configured for mobile nodes. However, the home agent must recognize the home address of the mobile node through the `mipagent.conf` file or some other mechanism when the home agent grants registration.

A home agent can support private addressed mobile nodes by configuring the private addressed mobile nodes in the `mipagent.conf` file. The home addresses that are used by the home agent must be unique.

Dynamic Home Agent Discovery

In some situations, the mobile node might not know the home agent address when the mobile node attempts to register. If the mobile node does not know the home agent address, the mobile node can use dynamic home agent address resolution to learn the address. In this situation, the mobile node sets the home agent field of the registration request to the subnet-directed broadcast address of its home network. Each home agent that receives a registration request with a broadcast destination address rejects the mobile node's registration by returning a rejection registration reply. By doing so, the mobile node can use the home agent's unicast IP address that is indicated in the rejection reply when the mobile node next attempts registration.

Routing Datagrams to and From Mobile Nodes

This section describes how mobile nodes, home agents, and foreign agents cooperate to route datagrams for mobile nodes that are connected to a foreign network. See "Overview of the Solaris Mobile IP Implementation" on page 441 for Mobile IP functions that are supported in the Solaris operating environment.

Encapsulation Types

Home agents and foreign agents use one of the available encapsulation methods to support datagrams that use a tunnel. Defined encapsulation methods are IP-in-IP Encapsulation, Minimal Encapsulation, and Generic Routing Encapsulation. Foreign and home agent, or indirect co-located mobile node and home agent cases, must support the same encapsulation method. All Mobile IP entities are required to support IP-in-IP Encapsulation.

Unicast Datagram Routing

When registered on a foreign network, the mobile node uses the following rules to choose a default router:

- If the mobile node is registered and uses a foreign agent care-of address, the process is straight forward. The mobile node chooses its default router from among the router addresses that are advertised in the ICMP router advertisement portion of that agent advertisement. The mobile node can also consider the IP source

address of the agent advertisement as another possible choice for the IP address of a default router.

- The mobile node might be registered directly with the home agent by using a co-located care-of address. Then the mobile node chooses its default router from among those routers that are advertised in any ICMP router advertisement message that it receives. The network prefix of the chosen default router must match the network prefix of the care-of address of the mobile node that is externally obtained. The address might match the IP source address of the agent advertisement under the network prefix. Then the mobile node can also consider that IP source address as another possible choice for the IP address of a default router.
- If the mobile node is registered, a foreign agent that supports reverse tunnels routes unicast datagrams from the mobile node to the home agent through the reverse tunnel. If the mobile node is registered with a foreign agent that provides reverse tunnel support, the mobile node must use that foreign agent as its default router.

Broadcast Datagrams

When a home agent receives a broadcast or multicast datagram, the home agent only forwards the datagram to mobile nodes that have specifically requested that they receive them. How the home agent forwards broadcast and multicast datagrams to mobile nodes depends primarily on two factors. Either that mobile node is using a foreign-agent provided care-of address, or the mobile node is using its own co-located care-of address. The former means that the datagram must be double encapsulated. The first IP header identifies the mobile node for which the datagram is to be delivered. Remember, this first IP header is not present in the broadcast or multicast datagram. The second IP header identifies the care-of address, and is the usual tunnel header. In the latter instance, the mobile node is de-capsulating its own datagrams, and the datagram needs only to be sent through the regular tunnel.

Multicast Datagram Routing

To begin receiving multicast traffic when a mobile node is visiting a foreign subnet, a mobile node can join a multicast group in any of the following ways:

- If the mobile node is using a co-located care-of address, the mobile node can use this address as the source IP address of any Internet Group Management Protocol (IGMP) join messages. However, a multicast router must be present on the visited subnet.
- If the mobile node wants to join the ICMP group from its home subnet, the mobile node must use a reverse tunnel to send IGMP join messages to the home agent. However, the mobile node's home agent must be a multicast router. The home

agent then forwards multicast datagrams through the tunnel to the mobile node.

- If the mobile node is using a co-located care-of address, the mobile node can use this address as the source IP address of any IGMP join messages. However, a multicast router must be present on the visited subnet. Once the mobile node has joined, the mobile node can participate by sending its own multicast packets directly on the visited network.
- Send directly on the visited network.
- Send through a tunnel to the home agent.

Multicast routing depends on the IP source address. A mobile node that is sending a multicast datagram must send the datagram from a valid source address on that link. This means a mobile node that is sending multicast datagrams directly on the visited network must use a co-located care-of address as the IP source address. Also, the mobile node must have joined the multicast group that is associated with the address. Similarly, a mobile node that joined a multicast group while on its home subnet before roaming, or joined the multicast group while roaming through a reverse tunnel to its home agent, must use its home address as the IP source address of the multicast datagram. Thus, the mobile node must have these datagrams reverse-tunneled to its home subnet as well, either through itself by using its co-located care-of address, or through a foreign agent reverse tunnel.

While it seems more efficient for a mobile node to always join from the subnet that the mobile node is visiting, it is still a mobile node. Consequently, the mobile node would have to repeat the join every time the mobile node switches subnets. The more efficient way is for the mobile node to join through its home agent, and not have to carry this overhead. Also, multicast sessions might be present that are only available from the home subnet. Other considerations might also force the mobile node to participate in a specific way.

Security Considerations

In many situations, mobile computers use wireless links to connect to the network. Wireless links are particularly vulnerable to passive eavesdropping, active replay attacks, and other active attacks.

Because Mobile IP recognizes its inability to reduce or eliminate this vulnerability, Mobile IP uses a form of authentication to protect Mobile IP registration messages from these types of attack. The default algorithm that is used is MD5, with a key size of 128 bits. The default operational mode requires that this 128–bit key precede and succeed the data to be hashed. The foreign agent uses MD5 to support authentication. The foreign agent also uses key sizes of 128 bits or greater, with manual key

distribution. Mobile IP can support more authentication algorithms, algorithm modes, key distribution methods, and key sizes.

These methods do prevent Mobile IP registration messages from being altered. However, Mobile IP also uses a form of replay protection to alert Mobile IP entities when they receive duplicates of previous Mobile IP registration messages. If this protection method were not used, the mobile node and its home agent might become unsynchronized when either one receives a registration message. Hence, Mobile IP updates its state. For example, a home agent receives a duplicate deregistration message while the mobile node is registered through a foreign agent. Replay protection is ensured either by a method known as nonces, or timestamps. Nonces and timestamps are exchanged by home agents and mobile nodes within the Mobile IP registration messages. Nonces and timestamps are protected from change by the authentication mechanism described previously. Consequently, if a home agent or mobile node sees a duplicate message, the duplicate message can be thrown away.

The use of tunnels can be a significant vulnerability, especially if registration is not authenticated. Also, the Address Resolution Protocol (ARP) is not authenticated, and can potentially be used to steal another host's traffic.

Use of IPsec With Mobile IP

In general, as home and foreign agents are fixed entities, they can use IPsec authentication or encryption to protect both Mobile IP registration messages and forward and reverse tunnel traffic. This process works completely independently of Mobile IP, and only depends on the workstation's ability to perform IPsec functions. Mobile nodes can also use IPsec authentication to protect their registration traffic. If the mobile node registers through a foreign agent, in general the mobile node cannot use IPsec encryption. The reason that the mobile node cannot use IPsec encryption is because the foreign agent must be able to check the information in the registration packet. While IPsec encryption could be used when a foreign agent is not needed, the issue of co-location makes this difficult to achieve. IPsec is an IP-level security relationship. Consequently, a home agent would have to know the mobile node's co-located address without prior information or registration messages. Several protocols can obviate the need for this information, but are beyond the scope of this document. For more information about IPsec, see Chapter 19 or Chapter 20.

CHAPTER 24

Administering Mobile IP (Task)

This chapter provides procedures for modifying, adding, deleting, and displaying parameters in the Mobile IP configuration file. This chapter also shows you how to display mobility agent status.

This chapter contains the following information:

Configuring the Mobile IP Configuration File

When you configure the `mipagent.conf` file for the first time, you need to perform the following tasks:

1. Depending on your organization's host's requirements, determine what functionality your Mobile IP agent can provide:
   - Foreign agent functionality only
   - Home agent functionality only
   - Both foreign and home agent functionality
2. Create the `/etc/inet/mipagent.conf` file and enter the settings you require by using the procedures that are described in this section. You can also copy one of the

following files to `/etc/inet/mipagent.conf` and modify it according to your requirements:

- For foreign agent functionality, copy `/etc/inet/mipagent.conf.fa-sample`.
- For home agent functionality , copy `/etc/inet/mipagent.conf.ha-sample`.
- For both foreign agent and home agent functionality, copy `/etc/inet/mipagent.conf-sample`.

3. You can reboot your system to invoke the boot script that starts the `mipagent` daemon. You can also start `mipagent` by typing the following command on a command line:

```
# /etc/inet.d/mipagent start
```

Configuring the Mobile IP Configuration File Task Map

The following table provides a brief description of the tasks that are described in this section.

TABLE 24–1 Configuring the Mobile IP Configuration File Task Map

Task	Description	For Instructions, Go to ...
Creating the Mobile IP configuration file	Involves creating the `/etc/inet/mipagent.conf` file or copying one of the sample files	"How to Create the Mobile IP Configuration File" on page 427
Configuring the `General` section	Involves typing the version number into the `General` section of the Mobile IP configuration file	"How to Configure the `General` Section" on page 427
Configuring the `Advertisements` section	Involves adding labels and values or changing them in the `Advertisements` section of the Mobile IP configuration file	"How to Configure the `Advertisements` Section" on page 428
Configuring the `GlobalSecurityParameters` section	Involves adding labels and values or changing them in the `GlobalSecurityParameters` section of the Mobile IP configuration file	"How to Configure the `GlobalSecurityParameters` Section" on page 428
Configuring the `Pool` section	Involves adding labels and values or changing them in the `Pool` section of the Mobile IP configuration file	"How to Configure the `Pool` Section" on page 428

TABLE 24–1 Configuring the Mobile IP Configuration File Task Map *(Continued)*

Task	Description	For Instructions, Go to ...
Configuring the `SPI` section	Involves adding labels and values or changing them in the `SPI` section of the Mobile IP configuration file	"How to Configure the `SPI` Section" on page 429
Configuring the `Address` section	Involves adding labels and values or changing them in the `Address` section of the Mobile IP configuration file	"How to Configure the `Address` Section" on page 429

▼ How to Create the Mobile IP Configuration File

1. **Become superuser on the system where you want to enable Mobile IP.**
2. **Depending on your preference, do one of the following substeps.**
   - In the `/etc/inet` directory, create an empty file named `mipagent.conf`.
   - From the following list, copy the sample file that provides the functionality you want to the file `/etc/inet/mipagent.conf`.
     - `/etc/inet/mipagent.conf.fa-sample`
     - `/etc/inet/mipagent.conf.ha-sample`
     - `/etc/inet/mipagent.conf-sample`
3. **Add or change configuration parameters in the `/etc/inet/mipagent.conf` file to conform to your configuration requirements. The remaining procedures in this section describe the steps that you perform.**

▼ How to Configure the `General` Section

If you copied one of the sample files, you can omit this procedure because the sample file contains this entry.

- **Edit the `/etc/inet/mipagent.conf` file and add the following lines.**

```
[General]
     Version = 1.0
```

Note – The `/etc/inet/mipagent.conf` file must contain the preceding entry.

"`General` Section" on page 447 provides descriptions of the labels and values that are used in this section.

▼ How to Configure the `Advertisements` Section

- **Edit the `/etc/inet/mipagent.conf` file and add or change the following lines by using the values that are required for your configuration.**

```
[Advertisements Interface-name]
     HomeAgent = <yes/no>
     ForeignAgent = <yes/no>
     PrefixFlags = <yes/no>
     AdvertiseOnBcast = <yes/no>
     RegLifetime = n
     AdvLifetime = n
     AdvFrequency = n
     ReverseTunnel = <yes/no/FA/HA/both>
     ReverseTunnelRequired = <yes/no/FA/HA>
```

Note – You must include a different `Advertisements` section for each interface on the local host that provides Mobile IP services.

"`Advertisements` Section" on page 447 provides descriptions of the labels and values that are used in this section.

▼ How to Configure the `GlobalSecurityParameters` Section

- **Edit the `/etc/inet/mipagent.conf` file and add or change the following lines by using the values that are required for your configuration.**

```
[GlobalSecurityParameters]
     MaxClockSkew = n
     HA-FAauth = <yes/no>
     MN-FAauth = <yes/no>
     Challenge = <yes/no>
     KeyDistribution = files
```

"`GlobalSecurityParameters` Section" on page 449 provides descriptions of the labels and values that are used in this section.

▼ How to Configure the `Pool` Section

- **Edit the `/etc/inet/mipagent.conf` file and add or change the following lines by using the values that are required for your configuration.**

```
[Pool Pool-identifier]
     BaseAddress = IP-address
```

```
Size = size
```

"Pool Section" on page 450 provides descriptions of the labels and values that are used in this section.

▼ How to Configure the SPI Section

- **Edit the /etc/inet/mipagent.conf file and add or change the following lines by using the values that are required for your configuration.**

```
[SPI SPI-identifier]
     ReplayMethod = <none/timestamps>
     Key = key
```

Note – You must include a different SPI section for each security context that is deployed.

"SPI Section" on page 451 provides descriptions of the labels and values that are used in this section.

▼ How to Configure the Address Section

- **Edit the /etc/inet/mipagent.conf file and add or change the following lines by using the values that are required for your configuration.**
  - **For a mobile node, use the following:**

```
[Address address]
     Type = node
     SPI = SPI-identifier
```

  - **For an agent, use the following:**

```
[Address address]
     Type = agent
     SPI = SPI-identifier
     IPsecRequest = action {properties} [: action {properties}]
     IPsecReply = action {properties} [: action {properties}]
     IPsecTunnel = action {properties} [: action {properties}]
```

action and *{properties}* are any action and associated properties that are defined in the ipsec(7P) man page.

Note – The SPI that is configured previously corresponds to the MD5 protection mechanism that is required by RFC 2002. The SPI that is configured previously does not correspond to the SPI that is used by IPsec. For more information about IPsec, see Chapter 19 and Chapter 20. Also see the `ipsec`(7P) man page.

- **For mobile node that is identified by its NAI, use the following:**

```
[Address NAI]
     Type = Node
     SPI = SPI-identifier
     Pool = Pool-identifier
```

- **For default mobile node, use the following:**

```
[Address Node-Default]
     Type = Node
     SPI = SPI-identifier
     Pool = Pool-identifier
```

"`Address` Section" on page 452 provides descriptions of the labels and values that are used in this section.

Modifying the Mobile IP Configuration File

This section shows you how to modify the Mobile IP configuration file by using the `mipagentconfig`(1M) command. The section also shows you how to display the current settings of parameter destinations.

"Configuring the Mobility IP Agent" on page 456 provides a conceptual description of the `mipagentconfig`(1M) command's usage. You can also review the `mipagentconfig`(1M) man page.

Modifying the Mobile IP Configuration File Task Map

TABLE 24–2 Modifying the Mobile IP Configuration File Task Map

Task	Description	For Instructions, Go to ...
Modifying the `General` section	Uses the `mipagentconfig change` command to change the value of a label in the `General` section of the Mobile IP configuration file	"How to Modify the `General` Section" on page 432
Modifying the `Advertisements` section	Uses the `mipagentconfig change` command to change the value of a label in the `Advertisements` section of the Mobile IP configuration file	"How to Modify the `Advertisements` Section" on page 432
Modifying the `GlobalSecurityParameters` section	Uses the `mipagentconfig change` command to change the value of a label in the `GlobalSecurityParameters` section of the Mobile IP configuration file	"How to Modify the `GlobalSecurityParameters` Section" on page 433
Modifying the `Pool` section	Uses the `mipagentconfig change` command to change the value of a label in the `Pool` section of the Mobile IP configuration file	"How to Modify the `Pool` Section" on page 433
Modifying the `SPI` section	Uses the `mipagentconfig change` command to change the value of a label in the `SPI` section of the Mobile IP configuration file	"How to Modify the `SPI` Section" on page 433
Modifying the `Address` section	Uses the `mipagentconfig change` command to change the value of a label in the `Address` section of the Mobile IP configuration file	"How to Modify the `Address` Section" on page 434
Adding or deleting parameters	Uses the `mipagentconfig add` or `delete` commands to add new parameters, labels, and values or to delete existing ones in any of the sections of the Mobile IP configuration file	"How to Add or Delete Configuration File Parameters" on page 435
Displaying the current settings of parameter destinations	Uses the `mipagentconfig get` command to display current settings of any section of the Mobile IP configuration file	"How to Display Current Parameter Settings in the Configuration File" on page 436

▼ How to Modify the `General` Section

1. **Become superuser on the system where you want to enable Mobile IP.**
2. **On a command line, type the following command for each label that you want to modify in the `General` section.**

   ```
   # mipagentconfig change <label> <value>
   ```

   The following example shows how you might change the version number (in the future) in the configuration file's `General` section.

   EXAMPLE 24–1 Changing Parameters in the `General` Section

   ```
   # mipagentconfig change version 2
   ```

▼ How to Modify the `Advertisements` Section

1. **Become superuser on the system where you want to enable Mobile IP.**
2. **On a command line, type the following command for each label that you want to modify in the `Advertisements` section.**

   ```
   # mipagentconfig change adv device-name <label> <value>
   ```

   For example, if you are changing the agent's advertised lifetime to 300 seconds for device `le0`, use the following command.

   ```
   # mipagentconfig change adv le0 AdvLifetime 300
   ```

   The following example shows how you might change other parameters in the configuration file's `Advertisements` section.

   EXAMPLE 24–2 Changing Parameters in the `Advertisements` Section

   ```
   # mipagentconfig change adv le0 HomeAgent yes
   # mipagentconfig change adv le0 ForeignAgent no
   # mipagentconfig change adv le0 PrefixFlags no
   # mipagentconfig change adv le0 RegLifetime 300
   # mipagentconfig change adv le0 AdvFrequency 4
   # mipagentconfig change adv le0 ReverseTunnel yes
   ```

▼ How to Modify the `GlobalSecurityParameters` Section

1. **Become superuser on the system where you want to enable Mobile IP.**
2. **On a command line, type the following command for each label that you want to modify in the `GlobalSecurityParameters` section.**

   ```
   # mipagentconfig change <label> <value>
   ```

   For example, if you are enabling home agent and foreign agent authentication, use the following command.

   ```
   # mipagentconfig change HA-FAauth yes
   ```

   The following example shows how you might change other parameters in the configuration file's `GlobalSecurityParameters` section.

   EXAMPLE 24–3 Changing Parameters in the `GlobalSecurityParameters` Section

   ```
   # mipagentconfig change MaxClockSkew 200
   # mipagentconfig change MN-FAauth yes
   # mipagentconfig change Challenge yes
   # mipagentconfig change KeyDistribution files
   ```

▼ How to Modify the `Pool` Section

1. **Become superuser on the system where you want to enable Mobile IP.**
2. **On a command line, type the following command for each label that you want to modify in the `Pool` section.**

   ```
   # mipagentconfig change Pool Pool-identifier <label> <value>
   ```

   For example, if you are changing the base address to `192.168.1.1` and size to `100` of Pool `10`, use the following commands.

   EXAMPLE 24–4 Changing Parameters in the `Pool` Section

   ```
   # mipagentconfig change Pool 10 BaseAddress 192.168.1.1
   # mipagentconfig change Pool 10 Size 100
   ```

▼ How to Modify the `SPI` Section

1. **Become superuser on the system where you want to enable Mobile IP.**
2. **On a command line, type the following command for each label that you want to**

modify in the `SPI` section.

```
# mipagentconfig change SPI SPI-identifier <label> <value>
```

For example, if you are changing the key for `SPI 257` to `5af2aee39ff0b332`, use the following command.

```
# mipagentconfig change SPI 257 Key 5af2aee39ff0b332
```

The following example shows how to change the `ReplayMethod` label in the configuration file's SPI section.

EXAMPLE 24–5 Changing Parameters in the `SPI` Section

```
# mipagentconfig change SPI 257 ReplayMethod timestamps
```

▼ How to Modify the `Address` Section

1. **Become superuser on the system where you want to enable Mobile IP.**
2. **On a command line, type the following command for each label that you want to modify in the `Address` section.**

   ```
   # mipagentconfig change addr [NAI | IPaddr | node-default] <label> <value>
   ```

   See "`Address` Section" on page 452 for a description of the three configuration methods (NAI, IP address, and node-default).

   For example, if you are changing the SPI of IP address `10.1.1.1` to `258`, use the following command.

   ```
   # mipagentconfig change addr 10.1.1.1 SPI 258
   ```

   The following example shows how you can change other parameters that provided in the sample configuration file's `Address` section.

   EXAMPLE 24–6 Changing Parameters in the `Address` Section

   ```
   # mipagentconfig change addr 10.1.1.1 Type agent
   # mipagentconfig change addr 10.1.1.1 SPI 259
   # mipagentconfig change addr mobilenode@abc.com Type node
   # mipagentconfig change addr mobilenode@abc.com SPI 258
   # mipagentconfig change addr mobilenode@abc.com Pool 2
   # mipagentconfig change addr node-default SPI 259
   # mipagentconfig change addr node-default Pool 3
   # mipagentconfig change addr 10.68.30.36 Type agent
   # mipagentconfig change addr 10.68.30.36 SPI 260
   # mipagentconfig change IPsecRequest apply {auth_algs md5 sa shared}
   ```

▼ How to Add or Delete Configuration File Parameters

1. **Become superuser on the system where you want to enable Mobile IP.**
2. **On a command line, type the appropriate command for each label that you want to add or delete for the designated section.**

   For the `General` section use the following:

   ```
   # mipagentconfig [add | delete] <label> <value>
   ```

   For the `Advertisements` section use the following:

   ```
   # mipagentconfig [add | delete] adv device-name <label> <value>
   ```

   Note – You can add an interface by typing the following:

   ```
   # mipagentconfig add adv device-name
   ```

   In this instance, default values are assigned to the interface (for both foreign agent and home agent).

   For the `GlobalSecurityParameters` section use the following:

   ```
   # mipagentconfig [add | delete] <label> <value>
   ```

   For the `Pool` section use the following:

   ```
   # mipagentconfig [add | delete] Pool Pool-identifier <label> <value>
   ```

   For the `SPI` section use the following:

   ```
   # mipagentconfig [add | delete] SPI SPI-identifier <label> <value>
   ```

   For the `Address` section use the following:

   ```
   # mipagentconfig [add | delete] addr [NAI | IPaddr | node-default] \
   <label> <value>
   ```

   Note – Do not create identical Advertisements, Pool, SPI, and Address sections.

   For example, to create a new address pool, Pool 11, that has a base address of `192.167.1.1` and a size of `100`, use the following commands.

   EXAMPLE 24–7 Adding a New Pool and Parameters

   ```
   # mipagentconfig add Pool 11 BaseAddress 192.167.1.1
   # mipagentconfig add Pool 11 size 100
   ```

Or you might want to delete a particular security parameter. The following example shows you how to delete `SPI 257`.

EXAMPLE 24–8 Deleting an SPI

```
# mipagentconfig delete SPI 257
```

▼ How to Display Current Parameter Settings in the Configuration File

You can use the `mipagentconfig get` command to display current settings that are associated with parameter destinations.

1. **Become superuser on the system where you are enabling Mobile IP.**
2. **On a command line, type the following command for each parameter for which you want to display settings.**

```
# mipagentconfig get [<parameter> | <label>]
```

For example, if you are displaying the advertisement settings for the `le0` device, use the following command.

```
# mipagentconfig get adv le0
```

This command causes the following results to be displayed (for example).

```
[Advertisements le0]
   HomeAgent = yes
   ForeignAgent = yes
```

The following example shows the results of using the `mipagentconfig get` command with other parameter destinations.

EXAMPLE 24–9 Using the `mipagentconfig get` Command

```
# mipagentconfig get MaxClockSkew
      [GlobalSecurityParameters]
         MaxClockSkew=300

# mipagentconfig get HA-FAauth
      [GlobalSecurityParameters]
         HA-FAauth=no

# mipagentconfig get MN-FAauth
      [GlobalSecurityParameters]
         MN-FAauth=no

# mipagentconfig get Challenge
      [GlobalSecurityParameters]
```

EXAMPLE 24–9 Using the `mipagentconfig get` Command *(Continued)*

```
         Challenge=no

# mipagentconfig get Pool 10
      [Pool 10]
         BaseAddress=192.168.1.1
         Size=100

# mipagentconfig get SPI 257
      [SPI 257]
         Key=11111111111111111111111111111111
         ReplayMethod=none

# mipagentconfig get SPI 258
      [SPI 258]
         Key=15111111111111111111111111111111
         ReplayMethod=none

# mipagentconfig get addr 10.1.1.1
      [Address 10.1.1.1]
         SPI=258
         Type=agent

# mipagentconfig get addr 192.168.1.200
      [Address 192.168.1.200]
         SPI=257
         Type=node

# mipagentconfig get addr 10.1.1.1
      [Address 10.1.1.1]
         Type=agent
         SPI=258
         IPsecRequest = apply {auth_algs md5 sa shared}
         IPsecReply = permit {auth_algs md5}
         IPsecTunnel = apply {encr_algs 3des sa shared}
```

Displaying Mobility Agent Status

You can use the `mipagentstat` command to display a foreign agent's visitors list and a home agent's binding table. "Mobile IP Mobility Agent Status" on page 457 provides a conceptual description of the `mipagentstat` command. You can also review the `mipagentstat`(1M) man page.

▼ How to Display Mobility Agent Status

1. **Become superuser on the system where you are enabling Mobile IP.**
2. **On a command line, type the following command.**

```
# mipagentstat <option>
```

You can use the following options:

`-f`	Shows the list of active mobile nodes in the foreign agent's visitor list
`-h`	Shows the list of active mobile nodes in the home agent's binding table
`-p`	Shows the list of security associations with an agent's mobility agent peers

For example, to show the visitor list for all mobile nodes that are registered with the foreign agent, use the following command.

```
# mipagentstat -f
```

This command causes the following results to be displayed (for example).

```
Mobile Node     Home Agent     Time (s)     Time (s)  Flags
                               Granted      Remaining
--------------- -------------- ------------ --------- -----
foobar.xyz.com  ha1.xyz.com    600          125       .....T.
10.1.5.23       10.1.5.1       1000         10        .....T.
```

To show the foreign agent security associations, use the following command.

```
# mipagentstat -p
```

This command causes the following results to be displayed (for example).

```
Foreign                 ..... Security Association(s).....
Agent                   Requests Replies  FTunnel  RTunnel
----------------------  -------- -------- -------- --------
forn-agent.eng.sun.com  AH       AH       ESP      ESP
```

To show the home agent security associations, use the following command.

```
# mipagentstat -fp
```

This command causes the following results to be displayed (for example).

```
Home                    ..... Security Association(s) .....
Agent                   Requests Replies  FTunnel  RTunnel
----------------------  -------- -------- -------- --------
home-agent.eng.sun.com  AH       AH       ESP      ESP
ha1.xyz.com             AH,ESP   AH       AH,ESP   AH,ESP
```

Displaying Mobility Routes on a Foreign Agent

You can use the `netstat` command to display additional information about source-specific routes that are created by forward and reverse tunnels. See the `netstat`(1M) man page for more information about this command.

▼ How to Display Mobility Routes on a Foreign Agent

1. **Become superuser on the system where you are enabling Mobile IP.**
2. **On a command line, type the following command.**

```
# netstat -rn
```

The following example shows the routes for a foreign agent that uses a reverse tunnel.

```
Routing Table:   IPv4 Source-Specific
Destination      In If     Source      Gateway Flags  Use  Out If
--------------  ------- ------------ --------- -----  ---- -------
10.6.32.11      ip.tun1      --      10.6.32.97  UH      0 hme1
    --          hme1    10.6.32.11       --      U       0 ip.tun1
```

The first line indicates that the destination IP address `10.6.32.11` and the incoming interface `ip.tun1` select `hme1` as the interface that forwards the packets. The next line indicates that any packet originating from interface `hme1` and source address `10.6.32.11` must be forwarded to `ip.tun1`. This is an example of a reverse-tunnel route.

CHAPTER 25

Mobile IP Files and Commands (Reference)

This chapter describes the components that are provided with the Solaris implementation of Mobile IP. To use Mobile IP, you must first configure the Mobile IP configuration file by using the parameters and commands that are described in the following sections.

This chapter contains the following information:

Overview of the Solaris Mobile IP Implementation

The mobility agent software incorporates home agent and foreign agent functionality. The Solaris Mobile IP software does not provide a client mobile node. Only the agent functionality is provided. Each network with mobility support should have at least one static (non-mobile) host running this software. The following RFC functions are supported in the Solaris implementation of Mobile IP:

RFC 1918 Address Allocation for Private Internets

RFC 2002 (Agent only) IP Mobility Support

RFC 2003 IP Encapsulation Within IP

RFC 2794 Mobile IP Network Access Identifier Extension for IPv4

RFC 3012 Mobile IP Challenge/Response Extensions

RFC 3024 Reverse Tunneling for Mobile IP

The base Mobile IP protocol (RFC 2002) does not address the problem of scalable key distribution and treats key distribution as an orthogonal issue. The Solaris Mobile IP software utilizes only manually configured keys, specified in a configuration file.

The functionality in the following IETF drafts is also supported in the Solaris implementation of Mobile IP:

- **draft-ietf-mobileip-rfc2002–bis-03.txt** – IP Mobility Support for IPv4, revised
- **draft-ietf-mobileip-vendor-ext-09.txt** – Mobile IP Vendor/Organization-Specific Extensions

The following RFC functions are not supported in the Solaris implementation of Mobile IP:

RFC 1700 General Routing Encapsulation

RFC 1701 General Routing Encapsulation

RFC 2004 Minimal Encapsulation Within IP

The following functions are not supported in the Solaris implementation of Mobile IP:

- The forwarding of multicast or broadcast traffic by the home agent to the foreign agent for a mobile node that is visiting a foreign network
- The routing of broadcast and multicast datagrams through reverse tunnels
- Private care-of addresses or private home agent addresses

See `mipagent`(1M) man page for additional information.

Mobile IP Configuration File

The `mipagent` command reads configuration information from the `/etc/inet/mipagent.conf` configuration file at startup. Mobile IP uses the `/etc/inet/mipagent.conf` configuration file to initialize the Mobile IP mobility agent. When configured and deployed, the mobility agent issues periodic router advertisements and responds to router discovery solicitation messages as well as Mobile IP registration messages.

See the `mipagent.conf`(4) man page for a description of file attributes and the `mipagent`(1M) man page for a description of its usage.

Configuration File Format

The Mobile IP configuration file consists of sections. Each section has a unique name and is enclosed in square brackets. Each section contains one or more labels. You assign values to the labels by using the following format:

```
[Section_name]
     Label-name = Value-assigned
```

"Configuration File Sections and Labels" on page 447 describes the section names, labels, and possible values.

Sample Configuration Files

The default Solaris installation provides the following sample configuration files in the `/etc/inet` directory:

- `mipagent.conf-sample` – Contains a sample configuration for a Mobile IP agent that provides both foreign and home agent functionality.
- `mipagent.conf.fa-sample` – Contains a sample configuration for a Mobile IP agent that provides only foreign agent functionality.
- `mipagent.conf.ha-sample` – Contains a sample configuration for a Mobile IP agent that provides only home agent functionality.

These sample configuration files contain mobile node address and security settings. Before you can implement Mobile IP, you must create a configuration file with the name `mipagent.conf` and place it in the `/etc/inet` directory. This file contains the configuration settings that satisfy your Mobile IP implementation requirements. You can also choose one of the sample configuration files, modify it with your addresses and security settings, and copy it to `/etc/inet/mipagent.conf`.

"How to Create the Mobile IP Configuration File" on page 427 shows the procedures to perform.

`mipagent.conf-sample` File

The following listing shows the sections, labels, and values that are contained in the `mipagent.conf-sample` file. "Configuration File Sections and Labels" on page 447 describes the syntax, sections, labels, and values.

```
[General]
   Version = 1.0    # version number for the configuration file. (required)

[Advertisements hme0]
   HomeAgent = yes
   ForeignAgent = yes
```

```
   PrefixFlags = yes
   AdvertiseOnBcast = yes
   RegLifetime = 200
   AdvLifetime = 200
   AdvFrequency = 5
   ReverseTunnel = no
   ReverseTunnelRequired = no

[GlobalSecurityParameters]
   MaxClockSkew = 300
   HA-FAauth = yes
   MN-FAauth = yes
   Challenge = no
   KeyDistribution = files

[Pool 1]
   BaseAddress = 10.68.30.7
   Size = 4

[SPI 257]
   ReplayMethod = none
   Key = 11111111111111111111111111111111

[SPI 258]
   ReplayMethod = none
   Key = 15111111111111111111111111111111

[Address 10.1.1.1]
   Type = node
   SPI = 258

[Address mobilenode@sun.com]
   Type = node
   SPI = 257
   Pool = 1

[Address Node-Default]
   Type = node
   SPI = 258
   Pool = 1

[Address 10.68.30.36]
   Type = agent
   SPI = 257
[Address 10.68.30.36]
   Type = agent
   SPI = 257
   IPsecRequest = apply {auth_algs md5 sa shared}
   IPsecReply = permit {auth_algs md5}
   IPsecTunnel =  apply {encr_algs 3des sa shared}
```

Example–`mipagent.conf.fa-sample` File

The following listing shows the sections, labels, and values that are contained in the `mipagent.conf.fa-sample` file. “Configuration File Sections and Labels” on page 447 describes the syntax, sections, labels, and values. The `mipagent.conf.fa-sample` file shows a configuration that provides only foreign agent functionality. This sample file does not contain a `Pool` section, because pools are used only by a home agent. Otherwise, this file is the same as the `mipagent.conf-sample` file.

```
[General]
   Version = 1.0    # version number for the configuration file. (required)

[Advertisements hme0]
   HomeAgent = no
   ForeignAgent = yes
   PrefixFlags = yes
   AdvertiseOnBcast = yes
   RegLifetime = 200
   AdvLifetime = 200
   AdvFrequency = 5
   ReverseTunnel = yes
   ReverseTunnelRequired = no

[GlobalSecurityParameters]
   MaxClockSkew = 300
   HA-FAauth = yes
   MN-FAauth = yes
   Challenge = no
   KeyDistribution = files

[SPI 257]
   ReplayMethod = none
   Key = 11111111111111111111111111111111

[SPI 258]
   ReplayMethod = none
   Key = 15111111111111111111111111111111

[Address 10.1.1.1]
   Type = node
   SPI = 258

[Address 10.68.30.36]
   Type = agent
   SPI = 257

[Address 10.68.30.36]
   Type = agent
   SPI = 257
   IPsecRequest = apply {auth_algs md5 sa shared}
   IPsecReply = permit {auth_algs md5}
   IPsecTunnel = apply {encr_algs 3des sa shared}
```

Example–`mipagent.conf.ha-sample` File

The following listing shows the sections, labels, and values that are contained in the `mipagent.conf.ha-sample` file. "Configuration File Sections and Labels" on page 447 describes the syntax, sections, labels, and values. The `mipagent.conf.ha-sample` file shows a configuration that provides only home agent functionality. Otherwise, this file is the same as the `mipagent.conf-sample` file.

```
[General]
   Version = 1.0    # version number for the configuration file. (required)

[Advertisements hme0]
   HomeAgent = yes
   ForeignAgent = no
   PrefixFlags = yes
   AdvertiseOnBcast = yes
   RegLifetime = 200
   AdvLifetime = 200
   AdvFrequency = 5
   ReverseTunnel = yes
   ReverseTunnelRequired = no

[GlobalSecurityParameters]
   MaxClockSkew = 300
   HA-FAauth = yes
   MN-FAauth = yes
   Challenge = no
   KeyDistribution = files

[Pool 1]
   BaseAddress = 10.68.30.7
   Size = 4

[SPI 257]
   ReplayMethod = none
   Key = 11111111111111111111111111111111

[SPI 258]
   ReplayMethod = none
   Key = 15111111111111111111111111111111

[Address 10.1.1.1]
   Type = node
   SPI = 258

[Address mobilenode@sun.com]
   Type = node
   SPI = 257
   Pool = 1

[Address Node-Default]
   Type = node
```

```
   SPI = 258
   Pool = 1

[Address 10.68.30.36]
    Type = agent
    SPI = 257
    IPsecRequest = apply {auth_algs md5 sa shared}
    IPsecReply = permit {auth_algs md5}
    IPsecTunnel = apply {encr_algs 3des sa shared}
```

Configuration File Sections and Labels

The Mobile IP configuration file contains the following sections:

- `General` (Required)
- `Advertisements` (Required)
- `GlobalSecurityParameters` (Optional)
- `Pool` (Optional)
- `SPI` (Optional)
- `Address` (Optional)

The `General` and `GlobalSecurityParameters` sections contain information relevant to the operation of the Mobile IP agent and can appear only once in the configuration file.

`General` Section

The `General` section contains only one label: the version number of the configuration file. The `General` section has the following syntax:

```
[General]
     Version = 1.0
```

`Advertisements` Section

The `Advertisements` section contains the `HomeAgent` and `ForeignAgent` labels, as well as other labels. You must include a different `Advertisements` section for each interface on the local host that provides Mobile IP services. The `Advertisements` section has the following syntax:

```
[Advertisements Interface-name]
     HomeAgent = <yes/no>
     ForeignAgent = <yes/no>
     .
     .
```

Typically, your system has a single interface (`le0`, `hme0`, and so on) and supports both home agent and foreign agent operations. If this is the situation for the example `hme0`, then the `yes` value is assigned to both the `HomeAgent` and `ForeignAgent` labels as follows:

```
[Advertisements hme0]
     HomeAgent = yes
     ForeignAgent = yes
     .
     .
```

For advertisement over dynamic interfaces, use '*' for the device id part. For example, *Interface-name* ppp* actually implies all ppp interfaces that are configured after `mipagent` has been started. All the attributes in the advertisement section of a dynamic interface type remain the same.

The following table describes the labels and values that you can use in the `Advertisements` section.

TABLE 25–1 `Advertisements` Section Labels and Values

Label	Value	Description
`HomeAgent`	`yes` or `no`	Determines if `mipagent` provides home agent functionality.
`ForeignAgent`	`yes` or `no`	Determines if `mipagent` provides foreign agent functionality.
`PrefixFlags`	`yes` or `no`	Specifies if advertisements include the optional prefix-length extension.
`AdvertiseOnBcast`	`yes` or `no`	If yes, advertisements are sent on 255.255.255.255, rather than 224.0.0.1.
`RegLifetime`	`n`	The maximum lifetime value that is accepted in registration requests, in seconds.
`AdvLifetime`	`n`	The maximum length of time that the advertisement is considered valid in the absence of further advertisements, in seconds.
`AdvFrequency`	`n`	Time between two consecutive advertisements, in seconds.

TABLE 25–1 `Advertisements` Section Labels and Values *(Continued)*

Label	Value	Description
`ReverseTunnel`	`yes` or `no` `FA` or `HA` or `both`	Determines if mipagent provides reverse-tunnel functionality. The value `yes` means that both the foreign agent and home agent support reverse tunneling. The value `no` means that the interface does not support reverse tunneling. The value `FA` means that the foreign agent supports reverse tunneling. The value `HA` means that the home agent supports reverse tunneling. The value `both` means that both the foreign agent and home agent support reverse tunneling.
`ReverseTunnelRequired`	`yes` or `no`	Determines if `mipagent` requires reverse tunnel functionality. Consequently, determines if a mobile node must request a reverse tunnel during registration. The value `yes` means that both the foreign agent and home agent require a reverse tunnel. The value `no` means that the interface does not require a reverse tunnel. The value `FA` means that the foreign agent requires a reverse tunnel. The value `HA` means that the home agent requires a reverse tunnel.
`AdvInitCount`	`n`	Determines initial number of unsolicited advertisements. The default value is 1. This value is meaningful only if `AdvLimitUnsolicited` is `yes`.
`AdvLimitUnsolicited`	`yes` or `no`	Enables or disables a limited number of unsolicited advertisements over the mobility interface.

`GlobalSecurityParameters` Section

The `GlobalSecurityParameters` section contains the `maxClockSkew`, `HA-FAauth`, `MN-FAauth`, `Challenge`, and `KeyDistribution` labels. This section defines the security parameters. The `GlobalSecurityParameters` section has the following syntax:

```
[GlobalSecurityParameters]
     MaxClockSkew = n
     HA-FAauth = <yes/no>
     MN-FAauth = <yes/no>
     Challenge = <yes/no>
```

```
KeyDistribution = files
```

The Mobile IP protocol provides message replay protection by allowing timestamps to be present in the messages. If the clocks differ, the home agent returns an error to the mobile node with the current time and the mobile node can re-register by using the current time. You use the `MaxClockSkew` label to configure the maximum number of seconds that differ between the home agent and the mobile node's clocks. The default value is 300 seconds.

The `HA-FAauth` and `MN-FAauth` labels enable or disable the requirement for home-foreign and mobile-foreign authentication, respectively. The default value is disabled. You use the `challenge` label so that the foreign agent issues challenges to the mobile node in its advertisements. The label is used for replay protection. The default value is disabled here, also.

The following table describes the labels and values that you can use in the `GlobalSecurityParameters` section.

TABLE 25–2 `GlobalSecurityParameters` Section Labels and Values

Label	Value	Description
`MaxClockSkew`	`n`	The number of seconds that `mipagent` accepts as a difference between its own local time and the time that is found in registration requests.
`HA-FAauth`	`yes` or `no`	Specifies if HA-FA authentication extensions must be present in registration requests and replies.
`MN-FAauth`	`yes` or `no`	Specifies if MN-FA authentication extensions must be present in registration requests and replies.
`Challenge`	`yes` or `no`	Specifies if the foreign agent includes challenges in its mobility advertisements.
`KeyDistribution`	`files`	Must be set to files.

`Pool` Section

Mobile nodes can be assigned dynamic addresses by the home agent. The dynamic address assignment is done within the `mipagent` independently of DHCP. You can create an address pool that can be used by mobile nodes by requesting a home address. Address pools are configured through the `Pool` section in the configuration file.

The `Pool` section contains the `BaseAddress` and `Size` labels. The `Pool` section has the following syntax:

```
[Pool Pool-identifier]
     BaseAddress = IP-address
```

```
     Size = size
```

Note – If you use a `Pool` identifier, then it must also exist in the mobile node's `Address` section.

You use the `Pool` section to define address pools that can be assigned to the mobile nodes. You use the `BaseAddress` label to set the first IP address in the pool. You use the `Size` to specify the number of addresses available in the pool.

For example, if IP Addresses `192.168.1.1` through `192.168.1.100` are reserved in pool 10, the `Pool` section has the following entry:

```
[Pool 10]
     BaseAddress = 192.168.1.1
     Size = 100
```

Note – Address ranges should not encompass the broadcast address. For example, you should not assign `BaseAddress = 192.168.1.200` and `Size = 60`, because this range encompasses the broadcast address `192.168.1.255`.

The following table describes the labels and values that are used in the `Pool` section.

TABLE 25–3 `Pool` Section Labels and Values

Label	Value	Description
`BaseAddress`	`n.n.n.n`	First address in the address pool
`Size`	`n`	Number of addresses in the pool

`SPI` Section

Because the Mobile IP protocol requires message authentication, you must identify the security context by using a Security Parameter Index (SPI). You define the security context in the `SPI` section. You must include a different `SPI` section for each security context that is defined. A numerical ID identifies the security context. The Mobile IP protocol reserves the first 256 SPIs. Therefore, you should use only SPI values greater than 256. The SPI section contains security-related information, such as shared secrets and replay protection.

The `SPI` section also contains the `ReplayMethod` and `Key` labels. This section defines the security contexts. The `SPI` section has the following syntax:

```
[SPI SPI-identifier]
     ReplayMethod = <none/timestamps>
     Key = key
```

Two communicating peers must share the same SPI identifier. You must configure them with the same key and replay method. You specify the key as a string of hex digits. The maximum length is 16 bytes. For example, if the key is 16 bytes long, and contains the hex values `0` through `f`, the key string might resemble the following:

```
Key = 0102030405060708090a0b0c0d0e0f10
```

Keys must have an even number of digits, corresponding to the two digits per byte representation.

The following table describes the labels and values that you can use in the SPI section.

TABLE 25–4 `SPI` Section Labels and Values

Label	Value	Description
`ReplayMethod`	`none` or `timestamps`	Specifies the type of replay authentication used for the SPI
`Key`	`x`	Authentication key in hexadecimal

`Address` Section

The Solaris implementation of Mobile IP enables you to configure mobile nodes in one of three methods. Each method is configured in the `Address` section. The first method follows the traditional Mobile IP protocol, and requires that each mobile node have a home address. The second method enables a mobile node to be identified through its Network Access Identifier (NAI). The last method enables you to configure a *default* mobile node, which can be used by any mobile node that has the proper SPI value and related keying material.

Mobile Node

The `Address` section for a mobile node contains the `Type` and `SPI` labels that define the address type and SPI identifier. The `Address` section has the following syntax:

```
[Address address]
     Type = node
     SPI = SPI-identifier
```

You must include an `Address` section in a home agent's configuration file for each mobile node that is supported.

If Mobile IP message authentication is required between the foreign and home agent, you must include an `Address` section for each peer with which an agent needs to communicate.

The SPI value that you configure must represent an SPI section that is present in the configuration file.

You can also configure private addresses for a mobile node.

The following table describes the labels and values that you can use in the `Address` section for a mobile node.

TABLE 25–5 `Address` Section Labels and Values—Mobile Node

Label	Value	Description
`Type`	`node`	Specifies that the entry is for a mobile node
`SPI`	`n`	Specifies the SPI value for the associated entry

Mobility Agent

The `Address` section for a mobility agent contains the `Type` and `SPI` labels that define the address type and SPI identifier. This section also contains IPsec request, reply, and tunnel labels. The `Address` section has the following syntax:

```
[Address address]
     Type = agent
     SPI = SPI-identifier
     IPsecRequest = action {properties} [: action {properties}]
     IPsecReply = action {properties} [: action {properties}]
     IPsecTunnel = action {properties} [: action {properties}]
```

You must include an `Address` section in a home agent's configuration file for each mobility agent that is supported.

If Mobile IP message authentication is required between the foreign and home agent, you must include an `Address` section for each peer with which an agent needs to communicate.

The SPI value that you configure must represent an SPI section that is present in the configuration file.

The following table describes the labels and values that you can use in the `Address` section for a mobility agent.

TABLE 25–6 `Address` Section Labels and Values—Mobility Agent

Label	Value	Description
`Type`	`agent`	Specifies that the entry is for a mobility agent

TABLE 25–6 `Address` Section Labels and Values—Mobility Agent *(Continued)*

Label	Value	Description
`SPI`	`n`	Specifies the SPI value for the associated entry
`IPsecRequest`	`apply` or `permit` (see following note)	IPsec properties to invoke for registration requests to and from this mobility agent peer
`IPsecReply`	`apply` or `permit` (see following note)	IPsec properties to invoke for registration replies to and from this mobility agent peer
`IPsecTunnel`	`apply` or `permit` (see following note)	IPsec properties to invoke for tunnel traffic to and from this mobility agent peer

Note – The `apply` values correspond to outbound datagrams. The `permit` values correspond to inbound datagrams. Therefore, `IPsecRequest apply` values and `IPsecReply permit` values are used by the foreign agent to send and receive registration datagrams. The `IPsecRequest permit` values and the `IPsecReply apply` values are used by the home agent to receive and send registration datagrams.

Mobile Node Identified by its NAI

The `Address` section for a mobile node that is identified by its NAI contains the `Type`, `SPI`, and `Pool` labels. The NAI parameter enables you to identify mobile nodes through their NAI. The `Address` section, using the NAI parameter, has the following syntax:

```
[Address NAI]
     Type = Node
     SPI = SPI-identifier
     Pool = Pool-identifier
```

In order to use pools, you identify mobile nodes through their NAI. The `Address` section permits you to configure an NAI, as opposed to a home address. An NAI uses the format `user@domain` format. You use the `Pool` label to specify which address pool to use in order to allocate the home address to the mobile node.

The following table describes the labels and values that you can use in the `Address` section for a mobile node that is identified by its NAI.

TABLE 25–7 `Address` Section Labels and Values—Mobile Node Identified by Its NAI

Label	Value	Description
`Type`	`node`	Specifies entry for a mobile node
`SPI`	`n`	Specifies SPI value for the associated entry
`Pool`	`n`	Allocates the pool from which an address is assigned to a mobile node

You must have corresponding `SPI` and `Pool` sections for the `SPI` and `Pool` labels that are defined in an `Address` section with a mobile node that is identified by its NAI, as shown in the following illustration.

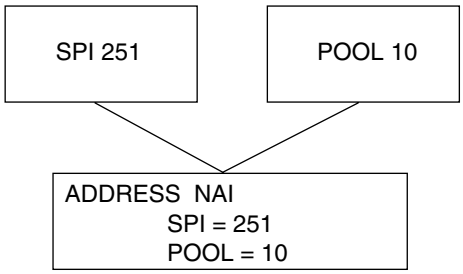


FIGURE 25–1 Corresponding SPI and Pool Sections for Address Section With Mobile Node Identified by Its NAI

Default Mobile Node

The `Address` section for a default mobile node contains the `Type`, `SPI`, and `Pool` labels. The `Node-Default` parameter enables you to permit all mobile nodes to get service if they have the correct SPI (defined in this section). The `Address` section, using the `Node-Default` parameter, has the following syntax:

```
[Address Node-Default]
     Type = Node
     SPI = SPI-identifier
     Pool = Pool-identifier
```

The `Node-Default` enables you to reduce the size of the configuration file. Otherwise, each mobile node requires its own section. However, the `Node-Default` does pose a security risk. If a mobile node is no longer trusted for any reason, you need to update the security information on all trusted mobile nodes. This task can be very tedious. However, you can use the `Node-Default` in networks that consider security risks unimportant.

The following table describes the labels and values that you can use in the `Address` section for a default mobile node.

TABLE 25–8 `Address` Section Labels and Values—Default Mobile Node

Label	Value	Description
`Type`	`node`	Specifies entry for a mobile node
`SPI`	`n`	Specifies SPI value for the associated entry
`Pool`	`n`	Allocates the pool from which an address is assigned to a mobile node

You must have corresponding `SPI` and `Pool` sections for the `SPI` and `Pool` labels that are defined in the `Address` section with a default mobile node, as shown in the following illustration.

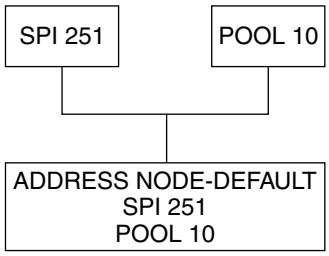


FIGURE 25–2 Corresponding SPI and Pool Sections for Address Section With a Default Mobile Node

Configuring the Mobility IP Agent

You can use the `mipagentconfig` command to configure the mobility agent. This command enables you to create or modify any parameter in the `/etc/inet/mipagent.conf` configuration file. Specifically, you can change any setting, and add or delete mobility clients, pools, and SPIs. The `mipagentconfig` command has the following syntax:

```
# mipagentconfig <command> <parameter> <value>
```

The following table describes the commands that you can use with `mipagentconfig` to create or modify parameters in the `/etc/inet/mipagent.conf` configuration file.

TABLE 25–9 `mipagentconfig` Commands

Command	Description
`add`	Used to add advertisement parameters, security parameters, SPIs, and addresses to the configuration file
`change`	Used to change advertisement parameters, security parameters, SPIs, and addresses in the configuration file
`delete`	Used to delete advertisement parameters, security parameters, SPIs, and addresses from the configuration file
`get`	Used to display current settings in the configuration file

See the `mipagentconfig`(1M) man page for a description of command parameters and acceptable values. "Modifying the Mobile IP Configuration File" on page 430 provides procedures that use the `mipagentconfig` command.

Mobile IP Mobility Agent Status

You can use the `mipagentstat` command to display a foreign agent's visitors list and a home agent's binding table. You can also display the security associations with an agent's mobility agent peers. To display the foreign agent visitor list, you use the `mipagentstat` command's `-f` option. To display the home agent binding table, you use the `mipagentstat` command's `-h` option. To display the security associations with an agent's mobility agent peers, you use the `mipagentstat` command's `-p` option. The following examples show typical output when using the `mipagentstat` command with these options.

EXAMPLE 25–1 Foreign Agent Visitor List

```
Mobile Node     Home Agent     Time (s)     Time (s)  Flags
                               Granted      Remaining
--------------- -------------- ------------ --------- -----
foobar.xyz.com  ha1.xyz.com    600          125       .....T.
10.1.5.23       10.1.5.1       1000         10        .....T.
```

EXAMPLE 25–2 Home Agent Binding Table

```
Mobile Node     Home Agent     Time (s)     Time (s)  Flags
                               Granted      Remaining
--------------- -------------- ------------ --------- -----
foobar.xyz.com  fa1.tuv.com    600          125       .....T.
10.1.5.23       123.2.5.12     1000         10        .....T.
```

EXAMPLE 25–3 Mobility Agent Peer Security Association Table

```
Foreign                  ..... Security Association(s).....
Agent                    Requests Replies  FTunnel  RTunnel
----------------------   -------- -------- -------- --------
forn-agent.eng.sun.com   AH       AH       ESP      ESP

Home                     ..... Security Association(s) .....
Agent                    Requests Replies  FTunnel  RTunnel
----------------------   -------- -------- -------- --------
home-agent.eng.sun.com   AH       AH       ESP      ESP
ha1.xyz.com              AH,ESP   AH       AH,ESP   AH,ESP
```

See `mipagentstat`(1M) command for more information about the command's options. "Displaying Mobility Agent Status" on page 437 provides procedures that use the `mipagentstat` command.

Mobile IP State Information

On shutdown, the `mipagent` daemon stores internal state information in `/var/inet/mipagent_state`. This occurs only when the `mipagent` provides services as a home agent. This state information includes the list of mobile nodes that are being supported as a home agent, their current care-of addresses, and remaining registration lifetimes. This state information also includes the security association configuration with mobility agent peers. If the `mipagent` program is terminated (for maintenance) and restarted, `mipagent_state` is used to re-create as much of the mobility agent's internal state as possible in an effort to minimize service disruption for mobile nodes that might be visiting other networks. If `mipagent_state` exists, it is read immediately after `mipagent.conf` every time `mipagent` is started or restarted.

`netstat` Extensions for Mobile IP

Mobile IP extensions have been added to the `netstat`(1M) command to identify Mobile IP forwarding routes. Specifically, you can use the `netstat`(1M) command to display a new routing table that is called "Source-Specific." See the `netstat`(1M) man page for more information.

The following example shows the output of `netstat` when you use the `-nr` flags.

EXAMPLE 25–4 Output From `netstat` Command

```
Routing Table:   IPv4 Source-Specific
Destination      In If     Source      Gateway Flags  Use  Out If
--------------  ------- ------------ --------- -----  ---- -------
10.6.32.11      ip.tun1      --      10.6.32.97  UH      0 hme1
    --          hme1    10.6.32.11       --      U       0 ip.tun1
```

The example shows the routes for a foreign agent that uses a reverse tunnel. The first line indicates that the destination IP address `10.6.32.11` and the incoming interface `ip.tun1` select `hme1` as the interface that forwards the packets. The next line indicates that any packet that originates from interface `hme1` and source address `10.6.32.11` must be forwarded to `ip.tun1`.

`snoop` Extensions for Mobile IP

Mobile IP extensions have been added to the `snoop`(1M) command to identify Mobile IP traffic on the link. See the `snoop`(1M) man page for more information.

The following example shows the output of `snoop` that runs on the mobile node, `mip-mn2`.

EXAMPLE 25–5 Output From `snoop` Command

```
mip-mn2# snoop
Using device /dev/hme (promiscuous mode)
  mip-fa2 -> 224.0.0.1    ICMP Router advertisement (Lifetime 200s [1]:
{mip-fa2-80 2147483648}), (Mobility Agent Extension), (Prefix Lengths),
(Padding)
  mip-mn2 -> mip-fa2   Mobile IP reg rqst
  mip-fa2 -> mip-mn2   Mobile IP reg reply (OK code 0)
```

This example shows that the mobile node received one of the periodically sent mobility agent advertisements from the foreign agent, `mip-fa2`. Then `mip-mn2` sent a registration request to `mip-fa2`, and in response, received a registration reply. The registration reply indicates that the mobile node successfully registered with its home agent.

The `snoop`(1M) command also supports IPsec extensions. Consequently, you can show how registration and tunnel packets are being protected.

CHAPTER 26

IP Network Multipathing Topics

Chapter 27	Provides overview information for IP Network Multipathing
Chapter 28	Provides step-by-step instructions for configuring IP Network Multipathing

CHAPTER 27

IP Network Multipathing (Overview)

IP Network Multipathing provides both load spreading and failover when you have multiple network interface cards that are connected to the same IP link (for example, Ethernet).

This chapter contains the following information:

Introduction

IP Network Multipathing provides your system with the following capabilities:

- Recovery from single-point failures with network adapters
- Increased traffic throughput

If a failure occurs in the network adapter, and if you have an alternate adapter connected to the same IP link, the system switches all the network accesses automatically from the failed adapter to the alternate adapter. This process ensures

uninterrupted access to the network. Also, when you have multiple network adapters that are connected to the same IP link, you achieve increased traffic throughput by spreading the traffic across multiple network adapters.

Note – Other IP-related documents, such as RFC 2460, use the term *link* instead of *IP link*. This document uses the term *IP link* to avoid confusion with IEEE 802. In IEEE 802, *link* refers to a single wire from an Ethernet NIC to an Ethernet switch.

See IP link definition in the Glossary or refer to Table 27–1.

IP Network Multipathing Features

The Solaris implementation of IP Network Multipathing provides the following features:

- **Failure Detection** – Ability to detect when a network adapter has failed and automatic switching (*failover*) of the network access to an alternate network adapter. This assumes that you have configured an alternate network adapter. See "Detecting Physical Interface Failures" on page 466 for more information.
- **Repair Detection** – Ability to detect when a network adapter that failed previously has been repaired and automatically switching back (*failback*) of the network access to an alternate network adapter. This assumes that you have enabled failbacks. See "Detecting Physical Interface Repairs" on page 468 for more information.
- **Outbound Load Spreading** – Outbound network packets are spread across multiple network adapters without affecting the ordering of packets in order to achieve higher throughput. Load spreading occurs only when the network traffic is flowing to multiple destinations that use multiple connections.

Communication Failures

Communication failures can occur in the following ways:

1. Transmit/receive path of the NIC can stop transmitting packets.
2. Attachment of the NIC to the link is down.
3. Port on the switch does not transmit/receive packets.
4. Physical interface in a group not present at system boot.

5. Host on the other end is not responding or the router that is forwarding the packets is not responding.

The Solaris implementation of IP Network Multipathing addresses the first four types of communication failures.

IP Network Multipathing Components

The following table identifies and describes the components that compose IP Network Multipathing.

TABLE 27–1 IP Network Multipathing Components

Component	Description
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/prefixes are assigned to an IP link. A subnet number/prefix cannot be assigned to more than one IP link. In ATM LANE, an IP link is a single emulated LAN. When using ARP, the scope of the ARP protocol is a single IP link.
Network Interface Card (NIC)	Network adapter that is either internal or a separate card that serves as an interface to a link.
Physical interface	A node's attachment to a link. This attachment is often implemented as a device driver plus a network adapter. Some network adapters can have multiple points of attachment, for example, qfe. The usage of *Network adapter* in this document refers to a "Single Point of Attachment."
Physical interface group	The set of physical interfaces on a system that are connected to the same link. They are identified by assigning the same (non-null) character string name to all the physical interfaces in the group.
Physical interface group name	A name that is assigned to a physical interface that identifies the group. The name is local to a system. Multiple physical interfaces, sharing the same group name, form a physical interface group.
Failure detection	The process of detecting when a NIC or the path from the NIC to some layer 3 device no longer works.

TABLE 27–1 IP Network Multipathing Components *(Continued)*

Component	Description
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.
Failover	The process of switching network access from a failed interface to a good physical interface. Network access includes IPv4 unicast, multicast, and broadcast traffic, as well as IPv6 unicast and multicast traffic.
Failback	The process of switching back network access to an interface that is detected to have been repaired.
Standby Interface	A physical interface that is not used to carry data traffic unless some other physical interface in the group has failed.

Solaris Network Multipathing

The following components implement Solaris network multipathing:

- Multipathing daemon – `in.mpathd`(1M)
- `ip`(7P)

The `in.mpathd` daemon detects failures and implements various policies for failover and failback. After `in.mpathd` detects a failure or repair, `in.mpathd` sends an ioctl to do the failover or failback. IP, which implements the ioctl, does the network access failover transparently and automatically.

Caution – Do not use Alternate Pathing while using IP Network Multipathing on the same set of NICs. Likewise, you should not use IP Network Multipathing while you are using Alternate Pathing. You can use Alternate Pathing and IP Network Multipathing at the same time on different sets of NICs.

Detecting Physical Interface Failures

The `in.mpathd` daemon can detect interface failure and repair by two methods. In the first method, the daemon sends and receives ICMP echo probes through the interface. In the second method, the daemon monitors the RUNNING flag on the interface. The link state on some models of network interface cards is reflected by the RUNNING flag. Consequently, when the link fails, the failure is detected much sooner.

An interface is considered to have failed if either of the previous two methods indicates failure. An interface is considered repaired only if both methods indicate that the interface is repaired.

The `in.mpathd` daemon sends ICMP echo probes to the targets that are connected to the link on all the interfaces that belong to a group to detect failures and repair. After you add an interface to a multipathing group and assign a test address, the daemon sends probes to detect failures on all the interfaces of the multipathing group. "How to Configure a Multipathing Interface Group With Two Interfaces" on page 482 describes the steps you perform to configure test address and groups.

Because `in.mpathd` determines which targets to probe dynamically, you cannot configure the targets. Routers that are connected to the link are chosen as targets for probing. If no routers exist on the link, arbitrary hosts on the link are chosen. A multicast packet that is sent to the "all hosts" multicast address (`224.0.0.1` in IPv4 and `ff02::1` in IPv6) determines the arbitrary hosts. The first few hosts that respond to the echo packets are chosen as targets for probing. If `in.mpathd` cannot find routers or hosts that responded to ICMP echo packets, `in.mpathd` cannot detect failures.

To ensure that each NIC in the group functions properly, `in.mpathd` probes all the targets separately through all the interfaces in the multipathing group. If no replies are made to five consecutive probes, `in.mpathd` considers the interface to have failed. The probing rate depends on the failure detection time (FDT). The default value for failure detection time is 10 seconds. The `in.mpathd`(1M) man page describes how to change the failure detection time. For a failure detection time of 10 seconds, the probing rate is approximately one probe every two seconds.

The failure detection time only applies to the ICMP echo probe method of detecting failures. If link failure results in the clearing of the RUNNING flag for an interface, the `in.mpathd` daemon responds immediately to the change in the flag status.

After a failure is detected, failover of all network access occurs from the failed interface to another functional interface in the group. If you have configured a standby interface, `in.mpathd` chooses the standby interface for failover of IP addresses, broadcasts, and multicast memberships. If you have not configured a standby interface, `in.mpathd` chooses the interface with the least number of IP addresses.

Physical interfaces in the same group that are not present at system boot represent a special instance of failure detection. The startup script `/etc/init.d/network` detects these types of failure. This type of failure displays error messages similar to the following:

```
moving addresses from failed IPv4 interfaces: hme0 (moved to hme1)
moving addresses from failed IPv6 interfaces: hme0 (moved to hme1)
```

Note – In this special instance of failure detection, only static IP addresses that are specified in host name files are moved to a different physical interface within the same multipathing group.

This type of failure can be automatically repaired by a failback. The RCM DR Post-attach feature for IP Network Multipathing automates the DR attachment of a NIC. When a NIC is DR attached, the interface is plumbed and configured. If the interface was removed prior to a reboot, the IP multipathing Reboot-safe feature recovers the IP address. The IP address is transferred to the replaced NIC. The replaced NIC is added to the original IP multipathing interface group. See "How to Recover a Physical Interface That Was Not Present at System Boot" on page 492.

Detecting Physical Interface Repairs

The `in.mpathd` daemon considers an interface repaired if the daemon receives responses to 10 consecutive probe packets, and the RUNNING flag is set on the interface.

When an interface fails, all addresses are moved to another functional interface in the group. Because `in.mpathd` needs an address for probing so that it can detect repairs, you must configure a test IP address that cannot move during the failover. Moreover, you should not allow a normal application to use this test address, because the failover of network access cannot occur for these addresses. "How to Configure a Multipathing Interface Group With Two Interfaces" on page 482 describes the steps that you perform. If `in.mpathd` detects a repair, failback of all network access occurs to the repaired interface.

As noted in "Detecting Physical Interface Failures" on page 466, automatic failback is supported for physical interfaces that are not present at system boot. See "How to Recover a Physical Interface That Was Not Present at System Boot" on page 492.

Group Failures

A group failure is when all the network interface cards appear to fail at the same time. `in.mpathd` does not do any failovers for a group failure. Also, no failover occurs when all the targets fail at the same time. In this instance, `in.mpathd` flushes all of its current targets and discovers new targets (see "Detecting Physical Interface Failures" on page 466).

Note – Group failures were previously known as link failures.

Administering Multipathing Groups With Multiple Physical Interfaces

This section describes how you enable IP Network Multipathing. To use the IP Network Multipathing feature, you should have more than one physical interface connected to the same IP link. For example, the same Ethernet switch or the same IP subnet, configured under the same multipathing group, works. If you have just one physical interface, refer to "Administering Multipathing Groups With a Single Physical Interface" on page 476.

Multipathing groups are identified by non-null names. For example, math-link, bio-link, and chem-link make good names. The names typically represent where these groups are connected. When failure is detected in one of the network adapters in the multipathing group, all network access is failed over from the failed adapter to the working adapter in the group. The failover of network access includes IPv4 unicast, broadcast, and multicast traffic, as well as IPv6 unicast and multicast traffic. For IP network multipathing to function properly, the following conditions must exist for the network adapters that are part of the same multipathing group:

1. You must push and configure the same set of STREAMS modules on all network adapters in the multipathing group.
2. If you have plumbed IPv4 on one network adapter, then you must plumb IPv4 on all network adapters in the multipathing group.
3. If you have plumbed IPv6 on one network adapter, then you must plumb IPv6 on all network adapters in the multipathing group.
4. All Ethernet network adapters in the system should have unique MAC address in the case of ethernets. This is achieved by setting the local-mac-address? to TRUE in the openboot PROM for SPARC platforms. Nothing needs to be done for IA (x86) platforms.
5. All network adapters of the multipathing group must be connected to the same IP link.
6. The multipathing group should not contain dissimilar interfaces. The interfaces that are grouped together should be of the same interface type that is defined in `/usr/include/net/if_types.h`. For example, you cannot combine Ethernet with Token ring, and you cannot combine a Token bus with asynchronous transfer mode (ATM).

7. When you use IP network multipathing with ATMs, you must configure the ATM for LAN emulation (multipathing over classical IP instances is not currently supported).

Note – The fourth condition concerns all interfaces in the system, not just those belonging to the multipathing group.

For the adapters that do not come with factory-set unique MAC addresses, you can manually configure a MAC address for each adapter as a workaround. For example, use the `ifconfig ether` command in a startup script file.

Note – The MAC addresses configured manually cannot be maintained across system reboot. You are responsible for choosing unique MAC addresses. IP Network Multipathing might behave unpredictably if the MAC addresses of adapters are not unique.

Grouping Physical Interfaces

You use the `ifconfig` command to configure groups. This command uses a new `group` parameter that requires a group name and places both the IPv4 and IPv6 instances of the interface in that group. The `group` parameter has the following syntax:

`ifconfig` *interface-name* `group` *group-name*

Note – Avoid using spaces in group names. The `ifconfig` status display does not show spaces. Consequently, if you have created two similar group names, but one of them contains a space, these group names look alike in the status display. However, they are different group names. This difference might be confusing.

Placing the IPv4 instance under a particular group automatically places the IPv6 instance under the same group. Also, you can place a second interface, connected to the same subnet, in the same group by using the same command. See "How to Configure a Multipathing Interface Group With Two Interfaces" on page 482.

You can remove an interface from a multipathing group by using a null string with the group sub-command. See "How to Remove an Interface From a Group" on page 488.

To place an interface in a new group when it is already part of some multipathing group, you do not need to remove it from any existing group. Placing the interface in a new group automatically removes it from any existing group. See "How to Move an Interface From an Existing Group to a Different Group" on page 489.

You can have any number of network adapters that you can configure in the same multipathing group. You cannot use the group parameter with logical interfaces. For example, you can use the parameter with `hme0`, but not with `hme0:1`.

You must connect all the interfaces in the multipathing group to the same IP link, because when an interface fails, the failover operation moves all the IP addresses from the failed interface to an interface in the group that is functional. For routers to continue routing packets to the addresses that have been switched to the functional interface, the functional interface must be connected to the same IP link.

Configuring Test Addresses

You must configure all physical interfaces of a multipathing group with a test address. You need test addresses to detect failures and repairs. If a test address is not configured, it is not chosen for failover. Only `in.mpathd` uses test addresses. Normal applications should not use this address. This address does not fail over when the interface fails. In IPv4, you should configure the test address in such a way that normal applications do not use the test address. See "How to Configure a Multipathing Interface Group With Two Interfaces" on page 482.

This section describes test address configuration concepts for the following Internet protocols:

- IPv4
- IPv6

IPv4 Test Addresses

The `in.mpathd` multipathing daemon requires a test IP address for detecting failures and repairs. You must use a routeable address for this IP address. That is, the subnet prefix of the address must be known to any routers present on the link. You use the `ifconfig` command's new `-failover` option to configure a test address. Use the following syntax to configure a test address:

```
# ifconfig interface-name addif ip-address <other-parameters> -failover up
```

For <other-parameters>, use the parameters that are required by your configuration. See the `ifconfig`(1M) man page for descriptions. "How to Configure a Multipathing Interface Group With Two Interfaces" on page 482 shows the steps you perform for an IPv4 test address.

For example, to add a new logical interface with an address of `19.16.85.21`, the netmask and broadcast address set to the default value, and also configure the interface with a test address, type the following:

```
# ifconfig hme0 addif 19.16.85.21 netmask + broadcast + -failover up
```

Note – You must mark an IPv4 test address as `deprecated` to prevent applications from using the test address. See "How to Configure a Multipathing Interface Group With Two Interfaces" on page 482.

Use `failover` without the dash to turn on the failover attribute of the address.

Note – All test IP addresses in a multipathing group must use the same network prefix. That is, the test IP addresses must belong to a single IP subnet.

IPv6 Test Addresses

To configure an IPv6 test address, you use the link-local address, because link-local addresses are tied to the physical interface. Thus, you do not need a separate IP address in the IPv6 situation. For IPv6, the `-failover` option has the following syntax:

```
# ifconfig interface-name inet6 -failover
```

"How to Configure a Multipathing Interface Group With Two Interfaces" on page 482 shows the steps you perform for an IPv6 test address.

When a multipathing group has both IPv4 and IPv6 plumbed on all the group's interfaces, you might not need a separate IPv4 test address. The `in.mpathd` daemon can probe the interfaces by using an IPv6 link-local address. IPv6 link-local addresses are created when IPv6 is plumbed.

Use `failover` without the dash to turn on the failover attribute of the address.

Note – The only valid IPv6 test address is the link-local address.

Preventing Applications From Using Test Addresses

After you have configured a test address, you need to ensure that this address is not used by normal applications. If you let applications use the test address, applications fail, because test addresses do not fail over during the failover operation. To ensure that IP does not pick the test address for normal applications, you mark the test address `deprecated` by using the `ifconfig` command. This parameter has the following syntax:

```
ifconfig interface-name deprecated
```

After you mark the address as deprecated, IP does not pick this address as a source address for any communication, unless the applications explicitly bind to the address.

Only `in.mpathd` explicitly binds to such an address. See "How to Configure a Multipathing Interface Group With Two Interfaces" on page 482.

Because link-local addresses are not present in the name service (DNS, NIS, and NIS+), applications do not use link-local addresses for communication. Consequently, you do not need to mark IPv6 test addresses as deprecated.

Note – You must not mark IPv6 link-local addresses as deprecated.

Use the `-deprecated` option to turn off the deprecated attribute of the address.

Note – IPv4 test addresses should not be placed in the name service tables (DNS/NIS/NIS+). In IPv6, link-local addresses are used as test addresses and are not normally placed in the name service tables.

Autoconfigured IPv6 addresses are not preserved across system reboot. If you require that IP addresses be preserved across reboot, then applications should use static IP addresses.

Using the `hostname` File to Configure Groups and Test Addresses

You can use the `/etc/hostname.`*interface* files to configure multipathing groups and test addresses. To configure a multipathing group by using the `/etc/hostname.`*interface* file, you can add a line to the file by using the following syntax:

```
interface-address <parameters>  group group-name up \
addif logical-interface-address <parameters> up
```

For example, to create the group `test` with the following configuration:

- Physical interface `hme0` with address `19.16.85.19`
- A logical interface address of `19.16.85.21`
- With `deprecated` and `-failover` set
- Sets the netmask and broadcast address to the default value

You add the following line to the `/etc/hostname.hme0` file:

```
19.16.85.19 netmask + broadcast + group test up \
addif 19.16.85.21 deprecated -failover netmask + broadcast + up
```

"How to Configure a Multipathing Interface Group With Two Interfaces" on page 482 shows the steps you perform to configure the IPv4 `hostname` file.

For IPv6 setup, add a line to the `/etc/hostname6.`*interface* file by using the following syntax:

```
<parameter> group group-name up
```

For example, to create a test group for `hme0` with an IPv6 test address, add the following line to the `/etc/hostname6.hme0` file:

```
-failover group test up
addif 1080::56:a00:20ff:feb9:19fa up
```

"How to Configure a Multipathing Interface Group With Two Interfaces" on page 482 shows the steps you perform to configure the IPv6 `hostname6` file.

Configuring Standby Interfaces

You can configure multipathing groups with standby interfaces. As the name implies, the interface is considered as standby and is not used unless some other interface in the group fails. A standby interface has an IFF_INACTIVE flag when the interface is not hosting any failover IP address. Consequently, when an active interface fails, the standby interface is always chosen for failover. After the standby interface is chosen, the IFF_INACTIVE flag is cleared on the standby interface. From that instant, the active standby is treated the same as other active interfaces. That is, some failures might not choose a standby interface. Instead, some failures might choose an active interface that hosts fewer IP addresses than the standby interface.

The standby interface is not used to send normal data packets. Consequently, limited traffic flows on a standby interface. You must configure standby interfaces with a test address to ensure that probes are sent to determine if the interface is functional. If you do not configure standby interfaces with a test address, the interface is not chosen for failovers when another interface in the group fails. A standby interface might carry traffic under the following conditions:

- If another host on the network communicates with a host by using the standby interface address, the standby interface is subsequently used for incoming packets.
- Applications binding (either using `bind` or using `IP_ADD_MEMBERSHIP)` to the address that is hosted on the standby interface might continue to generate traffic by using the standby interface.

Thus, the system does not normally select a standby interface (except for probes), unless it is explicitly chosen by an application. If some interface in the group fails, all network access is failed over to the standby interface. To configure a standby interface, you use the `ifconfig` command's new `standby` parameter by using the following syntax:

```
# ifconfig interface-name standby group group-name
```

"How to Configure a Multipathing Group With One of the Interfaces a Standby Interface" on page 485 shows the steps you perform.

The `in.mpathd` daemon sends probes on the standby interface after a test address is configured on the standby interface. You should configure only test addresses on a standby interface. If any other address is added on the standby, the addition of this address fails. If standby is marked on an interface that already has addresses other than test addresses, automatic failover of these addresses occurs to a different interface in the group, leaving behind only the test address, if one exists. It is advisable not to configure non-test address on a standby interface.

You need to mark the address as a test address by using the `ifconfig` command's `deprecated` and `-failover` options before setting `standby` or setting `up`.

To configure a test address on a standby interface, use the following syntax:

```
# ifconfig interface-name plumb ip-address
   <other-parameters> deprecated -failover standby up
```

For <other-parameters>, use the parameters that are required by your configuration. See the `ifconfig`(1M) man page for descriptions.

Note – Standby interfaces are not used for failover if no test address is configured on that interface.

For example, to create a test address with the following configuration:

- Physical interface `hme2` as a standby interface
- Address of `19.16.85.22`
- With `deprecated` and `-failover` set
- Sets the netmask and broadcast address to the default value

You type the following command line:

```
# ifconfig hme2 plumb 19.16.85.22 netmask + broadcast + deprecated -failover standby up
```

Note – The interface is marked as a standby interface only after the address is marked as a NOFAILOVER address.

"How to Configure a Multipathing Group With One of the Interfaces a Standby Interface" on page 485 shows the steps you perform.

You can clear a standby interface by using the following syntax:

```
# ifconfig interface-name -standby
```

Administering Multipathing Groups With a Single Physical Interface

When you have only one network adapter in the multipathing group, you can configure the network adapter to detect failures on that NIC alone.

Although failovers cannot occur with only one NIC in the group, you still need a separate test address on each of the physical interfaces in the group. You can configure the test address as an IFF_NOFAILOVER address, which is sufficient for the daemon to send out probes on that interface. Unlike the multiple physical interface instance, you do not have to mark a single physical interface as deprecated.

Use the following syntax to configure the interface's IPv4 address as a NOFAILOVER:

```
# ifconfig interface-name -failover group group-name
```

For IPv6, use the following syntax:

```
# ifconfig interface-name inet6 -failover group group-name
```

When the daemon detects failures, the interface is marked and logged appropriately on the console.

Note – You cannot verify whether the target that is being probed has failed or the NIC has failed, because the target can be probed through only one physical interface. If only one default router is on the subnet, turn off multipathing if a single physical interface is in the group. If a separate IPv4 and IPv6 default router exists (or multiple default routers exist), more than one target needs to be probed. Hence, you can safely turn on multipathing.

Removing Network Adapters From Multipathing Groups

When you execute the `ifconfig` command's `group` parameter with a null string, the interface is removed from the existing group. See "How to Remove an Interface From a Group" on page 488. Be careful when removing interfaces from a group. If some other interface in the multipathing group failed, a failover could have happened earlier. For example, if `hme0` failed previously, all addresses are failed over to `hme1` (if `hme1` is part of the same group). The removal of `hme1` from the group causes

`in.mpathd` to return all the failover addresses to some other interface in the group. If no other interfaces are functioning in the group, failover might not restore all the network accesses.

Similarly, when an interface is part of the group and the interface needs to be unplumbed, you should remove the interface from the group first. Then ensure that the interface has all the original IP addresses configured on it. The `in.mpathd` daemon tries to restore the original configuration of an interface that is removed from the group. You need to ensure that the configuration is restored before unplumbing the interface. Refer to "Multipathing Daemon" on page 477 to see how interfaces look before and after a failover.

Detached Network Adapters

Dynamic Reconfiguration (DR) uses IP Network Multipathing to decommission a specific network device without impacting existing IP users. Before a NIC is DR-detached (off lined), all failover IP addresses that are hosted on that NIC are automatically failed over to another NIC in the same IP Network Multipathing group. The test addresses are brought down and the NIC is unplumbed.

With the IP Network Multipathing reboot-safe feature, the static IP addresses in the `/etc/hostname.*` file that are associated with the missing card are hosted automatically on an alternate interface within the same IP Network Multipathing group. However, these addresses are returned to the original interface automatically if the original interface is inserted back into the system at a later time.

Multipathing Daemon

The `in.mpathd` multipathing daemon detects failures and repairs by sending out probes on all the interfaces that are part of a group. The `in.mpathd` multipathing daemon also detects failures and repairs by monitoring the RUNNING flag on each interface in the group. When an interface is part of a group and has a test address, the daemon starts sending out probes for determining failures on that interface. If the daemon does not receive any replies to five consecutive probes, or the RUNNING flag is not set, the interface is considered to have failed. The probing rate depends on the failure detection time. By default, failure detection time is 10 seconds. Thus, the probing rate is one probe every two seconds. To avoid synchronization in the network, probing is not periodic. If five consecutive probes fail, `in.mpathd` considers the

interface as failed and performs a failover of the network access from the failed interface to another interface in the group that is functioning properly. If a standby interface is configured, it is chosen for failover of the IP addresses, and broadcasts and multicast memberships. If no standby interface exists, the interface with the least number of IP addresses is chosen. Refer to the man page `in.mpathd(1M)` for more information.

The following two examples show a typical configuration and how the configuration automatically changes when an interface fails. When the `hme0` interface fails, notice that all addresses move from `hme0` to `hme1`.

EXAMPLE 27–1 Interface Configuration Before an Interface Failure

```
hme0: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
     inet 19.16.85.19 netmask ffffff00 broadcast 19.16.85.255
     groupname test
hme0:1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500
     index 2 inet 19.16.85.21 netmask ffffff00 broadcast 129.146.85.255
hme1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
     inet 19.16.85.20 netmask ffffff00 broadcast 19.16.85.255
     groupname test
hme1:1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500
     index 2 inet 19.16.85.22 netmask ffffff00 broadcast 129.146.85.255
hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
     inet6 fe80::a00:20ff:feb9:19fa/10
     groupname test
hme1: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
     inet6 fe80::a00:20ff:feb9:1bfc/10
     groupname test
```

EXAMPLE 27–2 Interface Configuration After an Interface Failure

```
hme0: flags=19000842<BROADCAST,RUNNING,MULTICAST,IPv4,NOFAILOVER,FAILED> mtu 0 index 2
        inet 0.0.0.0 netmask 0
        groupname test
hme0:1: flags=19040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER,FAILED>
        mtu 1500 index 2 inet 19.16.85.21 netmask ffffff00 broadcast 129.146.85.255
hme1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 19.16.85.20 netmask ffffff00 broadcast 19.16.85.255
        groupname test
hme1:1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500
        index 2 inet 19.16.85.22 netmask ffffff00 broadcast 129.146.85.255
hme1:2: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 6
        inet 19.16.85.19 netmask ffffff00 broadcast 19.16.18.255
hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER,FAILED> mtu 1500 index 2
        inet6 fe80::a00:20ff:feb9:19fa/10
        groupname test
hme1: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
        inet6 fe80::a00:20ff:feb9:1bfc/10
        groupname test
```

You can see that the FAILED flag is set on hme0 to indicate that hme0 has failed. You can also see that hme1:2 is now created. hme1:2 was originally hme0. The address 19.16.85.19 then becomes accessible through hme1. Multicast memberships that are associated with 19.16.85.19 can still receive packets, but now through hme1. When the failover of address 19.16.85.19 from hme0 to hme1 occurred, a dummy address 0.0.0.0 was created on hme0. The dummy address is removed when a subsequent failback takes place. The dummy address is created so that hme0 can still be accessed. hme0:1 cannot exist without hme0.

Similarly, failover of the IPv6 address from hme0 to hme1 occurred. In IPv6, multicast memberships are associated with interface indexes. They also fail over from hme0 to hme1. All the addresses that in.ndpd configures also move, this action is not shown in the examples.

The in.mpathd daemon continues to probe through the failed NIC, hme0. After the daemon receives 10 consecutive replies for a default failure detection time of 10 seconds, the daemon considers the interface repaired and invokes the failback. After failback, the original configuration is reestablished.

See in.mpathd(1M) man page for a description of all error messages there are logged on the console during failures and repairs.

Multipathing Configuration File

The in.mpathd daemon uses the settings in the /etc/default/mpathd configuration file to invoke multipathing. Changes to this file are read by in.mpathd at startup and on SIGHUP. This file contains the following default settings and information:

```
#
# Time taken by mpathd to detect a NIC failure in ms. The minimum time
# that can be specified is 100 ms.
#
FAILURE_DETECTION_TIME=10000
#

# Failback is enabled by default. To disable failback turn off this option
#
FAILBACK=yes
#

# By default only interfaces configured as part of multipathing groups
# are tracked. Turn off this option to track all network interfaces
# on the system
#
TRACK_INTERFACES_ONLY_WITH_GROUPS=yes
```

"How to Configure the Multipathing Configuration File" on page 494 shows the steps you perform to configure the `/etc/default/mpathd` configuration file.

Failure Detection Time

You can set a lower value of failure detection time. Sometimes these values might not be achieved if the load on the network is too high. Then `in.mpathd` prints a message on the console, indicating that the time cannot be met. The daemon also prints the time that it can meet currently. If the response comes back correctly, `in.mpathd` meets the failure detection time that is provided in this file.

Failback

After a failover, failbacks occurs when the failed interface is repaired. However, `in.mpathd` does not fail back the interface if `FAILBACK` is set to `no`.

As noted in "Detecting Physical Interface Failures" on page 466, automatic failback is supported for physical interfaces that are not present at system boot. See "How to Recover a Physical Interface That Was Not Present at System Boot" on page 492.

Track Interfaces Only With Groups Option

By turning off this option, `in.mpathd` tracks all interfaces in the system. When a failure is detected, an appropriate message is logged on the console. For this option to function properly, Ethernet addresses on all the interfaces must be unique.

CHAPTER 28

Administering Network Multipathing (Task)

This chapter provides procedures for creating and working with an interface group, configuring test addresses, configuring the `hostname` file, and configuring the multipathing configuration file.

This chapter contains the following information:

Configuring Multipathing Interface Groups

This section provides procedures for configuring multipathing interface groups. It also describes how to make an interface a hot standby interface.

"Grouping Physical Interfaces" on page 470 provides additional information.

Configuring Multipathing Interface Groups—Task Map

TABLE 28–1 Configuring Multipathing Interface Groups—Task Map

Task	Description	For Instructions, Go to ..
Configuring a multipathing interface group with two interfaces	Use the `ifconfig` command, the `group` parameter, `-failover` option, the `deprecated` option, and the `/etc/hostname.`*interface* file	"How to Configure a Multipathing Interface Group With Two Interfaces" on page 482
Configuring a multipathing group where one of the interfaces is a standby interface	Use the `ifconfig` command, the `group` parameter, standby parameter, `-failover` option, and the `/etc/hostname.`*interface* file	"How to Configure a Multipathing Group With One of the Interfaces a Standby Interface" on page 485
Displaying the group to which a physical interface belongs	Use the `ifconfig` command and the interface name	"How to Display the Group to Which a Physical Interface Belongs" on page 487
Adding an interface to a group	Use the `ifconfig` command and the interface name	"How to Add an Interface To a Group" on page 488
Removing an interface from a group	Use the `ifconfig` command and a null string to disable IP network multipathing	"How to Remove an Interface From a Group" on page 488
Moving an interface from an existing group to a different group	Use the `ifconfig` command and the `group` parameter	"How to Move an Interface From an Existing Group to a Different Group" on page 489

▼ How to Configure a Multipathing Interface Group With Two Interfaces

1. **Become superuser.**
2. **Place each physical interface into a multipathing group by typing the following command.**

   `# ifconfig` *interface-name* `group` *group-name*

   For example, to place `hme0` and `hme1` under group `test`, you type the following commands:

   ```
   # ifconfig hme0 group test
   # ifconfig hme1 group test
   ```

3. **Configure a test address for all the physical interfaces.**

- For an IPv4 test address, type the following command.

Note – This step assumes that you have already configured your physical interfaces' addresses.

```
# ifconfig interface-name addif ip-address <parameters> -failover deprecated up
```

For example, to configure a test address on `hme0` with the following configuration:

- Address set to `19.16.85.21`
- Netmask and broadcast address set to the default value
- `-failover` and `deprecated` options set

You type the following command:

```
# ifconfig hme0 addif 19.16.85.21 netmask + broadcast + -failover deprecated up
```

You can check the configuration by typing the following:

```
# ifconfig hme0:1
    hme0:1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
    mtu 1500 index 2 inet 19.16.85.21 netmask ffffff00 broadcast 19.16.85.255
```

Note – You must mark an IPv4 test address as deprecated to prevent applications from using the test address.

To configure a test address on `hme1` with the following configuration:

- Address set to `19.16.85.22`
- Netmask and broadcast address set to the default value
- `-failover` and `deprecated` options set

Type the following command:

```
# ifconfig hme1 addif 19.16.85.22 netmask + broadcast + -failover deprecated up
```

- For an IPv6 test address, type the following command.

  ```
  # ifconfig interface-name inet6 -failover
  ```

Note – Because you have already placed the physical interfaces with IPv4 addresses into a multipathing group, physical interfaces with IPv6 addresses are also implicitly placed in the same multipathing group. You might have placed physical interfaces with IPv6 addresses into a multipathing group first. Then physical interfaces with IPv4 addresses would have been also implicitly placed in the same multipathing group.

For example, to configure `hme0` with an IPv6 test address, you type the following command:

```
# ifconfig hme0 inet6 -failover
```

You can check the configuration by typing the following:

```
# ifconfig hme0 inet6
    hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500
            index 2 inet6 fe80::a00:20ff:feb9:17fa/10
            groupname test
```

Note – You do not need to mark an IPv6 test address as deprecated to prevent applications from using the test address.

For the second interface, `hme1`, type the following command:

```
# ifconfig hme1 inet6 -failover
```

4. **(Do this step only if you want to preserve the configuration across reboots.) To preserve the configuration across reboots, do the following substeps.**

   - For IPv4, edit the `/etc/hostname.`*interface* file and add the following line.

     interface-address `<parameters> group` *group-name* `up \`
     `addif` *logical-interface* `-failover deprecated <parameters> up`

Note – This test IP address is configured only on the next reboot. If you want the configuration to be invoked in the current session, do steps 1, 2, and 3.

For example, to create a group `test` with the following configuration for `hme0`:

- Physical interface `hme0` with address `19.16.85.19`
- A logical interface address of `19.16.85.21`
- With `deprecated` and `-failover` set
- Sets the netmask and broadcast address to the default value

You add the following line to the `/etc/hostname.hme0` file:

```
19.16.85.19 netmask + broadcast + group test up \
    addif 19.16.85.21 deprecated -failover netmask + broadcast + up
```

Similarly, to place `hme1` under the same group `test` and configure a test address, type the following command:

```
19.16.85.20 netmask + broadcast + group test up \
    addif 19.16.85.22 deprecated -failover netmask + broadcast + up
```

- For IPv6, edit the `/etc/hostname6.`*interface* file and add the following line.

```
-failover group group-name up
```

Note – This test IP address is configured only on the next reboot. If you want the configuration to be invoked in the current session, do steps 1, 2, and 3.

For example, to create a test group for `hme0` with an IPv6 address, add the following line to the `/etc/hostname6.hme0` file:

```
-failover group test up
```

Similarly, to place `hme1` under the same group `test` and configure a test address, add the following line to the `/etc/hostname6.hme1` file:

```
-failover group test up
```

Note – To add more interfaces to the multipathing group, repeat steps 1 through 3. New interfaces can be added to an existing group on a live system. However, changes are lost across reboots.

▼ How to Configure a Multipathing Group With One of the Interfaces a Standby Interface

The examples that are used in this procedure assume that `hme1` will be configured as the standby interface.

Note – A standby interface has only a test address.

---

1. **Do steps 1 and 2 in "How to Configure a Multipathing Interface Group With Two Interfaces" on page 482.**
2. **Configure the test address on all physical interfaces by using the following substeps.**

a. For the non-standby interface, such as `hme0`, do step 3 in "How to Configure a Multipathing Interface Group With Two Interfaces" on page 482.

b. For a standby interface, configure a test address by typing the following command.

Note – A standby interface can have only a test address. A standby interface cannot have any other IP address.

```
# ifconfig interface-name plumb ip-address <other-parameters> deprecated -failover
standby up
```

Note – You must set the `-failover` option before the `standby` option and the `standby` option before `up`.

For `<other-parameters>`, use the parameters that are required by your configuration. See the `ifconfig`(1M) man page for descriptions.

For example, to create a test address with the following configuration:

- Physical interface `hme1` as a standby interface
- Address of `19.16.85.22`
- With `deprecated` and `-failover` set
- Sets the netmask and broadcast address to the default value

You type the following command:

```
# ifconfig hme1 plumb 19.16.85.22 netmask + broadcast + deprecated -failover standby up
```

You can check the results by typing the following:

```
# ifconfig hme1
flags=69040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER,STANDBY,INACTIVE>
        mtu 1500 index 4 inet 19.16.85.22 netmask ffffff00 broadcast 19.16.85.255
        groupname test
```

For IPv6, to create a test address, type the following command:

```
ifconfig hme1 plumb -failover standby up
```

The `INACTIVE` flag indicates that this interface is not used for any outbound packets. When a failover occurs on this standby interface, the `INACTIVE` flag is cleared.

3. (Do this step only if you want to preserve the configuration across reboots.) To preserve the configuration across reboots, do the following substeps.

- For IPv4, edit the `/etc/hostname.`*interface* file and add the following line.

```
interface-address <parameters> group group-name up \
     addif logical-interface-failover deprecated <parameters> up
```

Note – This test IP address is configured only on the next reboot. If you want the configuration to be invoked in the current session, do steps 1 and 2.

For example, to create a group `test` with the following configuration for `hme0`:

- Physical interface `hme0` with address `19.16.85.19`
- A logical interface address of `19.16.85.21`
- With `deprecated` and `-failover` set
- Sets the netmask and broadcast address to the default value

You add the following line to the `/etc/hostname.hme0` file:

```
19.16.85.19 netmask + broadcast + group test up \
    addif 19.16.85.21 deprecated -failover netmask + broadcast + up
```

Similarly, to place the standby interface `hme1` under the same group `test` and configure a test address, type the following command:

```
19.16.85.22 netmask + broadcast + deprecated group test -failover standby up
```

- For IPv6, edit the `/etc/hostname6.`*interface* file and add the following line.

  `-failover group` *group-name* `up`

Note – This test IP address is configured only on the next reboot. If you want the configuration to be invoked in the current session, do steps 1 and 2.

For example, to create a test group for `hme0` with an IPv6 address, add the following line to the `/etc/hostname6.hme0` file:

```
-failover group test up
```

Similarly, to place the standby interface `hme1` under the same group `test` and configure a test address, add the following line to the `/etc/hostname6.hme1` file:

```
-failover group test standby up
```

▼ How to Display the Group to Which a Physical Interface Belongs

1. **Become superuser.**
2. **On a command line, type the following command.**

   `# ifconfig` *interface-name*

For example, to display the group name for `hme0`, you type the following command:

```
# ifconfig hme0
    hme0: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
      index 2 inet 19.16.85.19 netmask ffffff00 broadcast 19.16.85.255
      groupname test
```

To display the group name for only the IPv6 instance, you type the following command:

```
# ifconfig hme0 inet6
    hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
            inet6 fe80::a00:20ff:feb9:19fa/10
            groupname test
```

▼ How to Add an Interface To a Group

1. **Become superuser.**
2. **On a command line, type the following command.**

   `# ifconfig` *interface-name* `group` *group-name*

   For example, to add `hme0` to the group `test`, you type the following command:

   ```
   # ifconfig hme0 group test
   ```

▼ How to Remove an Interface From a Group

1. **Become superuser.**
2. **On a command line, type the following command.**

   `# ifconfig` *interface-name* `group ""`

   The quotation marks indicate a null string.

   For example, to remove `hme0` from the group `test`, you type the following command:

   ```
   # ifconfig hme0 group ""
       # ifconfig hme0
       hme0: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
               index 2 inet 19.16.85.19 netmask ffffff00 broadcast 19.16.85.255
       # ifconfig hme0 inet6
       hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
               inet6 fe80::a00:20ff:feb9:19fa/10
   ```

   "Removing Network Adapters From Multipathing Groups" on page 476 provides additional information.

▼ How to Move an Interface From an Existing Group to a Different Group

1. **Become superuser.**
2. **On a command line, type the following command.**

   `# ifconfig` *interface-name* `group` *group-name*

Note – Placing the interface in a new group automatically removes it from any existing group.

For example, to remove `hme0` from group `test` and place it in group `cs-link`, you type the following:

```
# ifconfig hme0 group cs-link
```

This command removes the interface from any existing group and then puts the interface in the group cs-link.

Replacing a Physical Interface That Has Failed or DR-detaching/DR-attaching a Physical Interface

The steps in this section pertain to only IP layers that are configured by using `ifconfig`(1M). Layers above or below the IP layer, such as ATM or other services, require specific manual steps if the layers are not automated. The specific steps are used to unconfigure during pre-detach and configure after post-attach. See the layers and applications documentation for instructions on how to handle the failure and DR scenarios.

You must do the following manual steps before replacing a physical interface that has failed. The following procedures use physical interfaces `hme0` and `hme1` as example interfaces. The procedures assume that both interfaces are in a multipathing group and that `hme0` has failed. The procedures also assume that the logical interface `hme0:1` has the test address.

Note – These procedures assume that you are replacing the failed interface with the same physical interface name (for example, `hme0` with `hme0`).

▼ How to Remove a Physical Interface That Has Failed

Note – You can skip step 1 if the test address is plumbed by using the `/etc/hostname.hme0` file.

1. **Retrieve the test address configuration by typing the following command.**

```
# ifconfig hme0:1

hme0:1:
flags=9040842<BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 3
inet 129.146.233.250 netmask ffffff00 broadcast 129.146.233.255
```

   You need this information to replumb the test address when replacing the physical interface.

   See "Using the `hostname` File to Configure Groups and Test Addresses" on page 473 for details on how to configure test addresses by using the `hostname` file.

2. **Refer to the `cfgadm`(1M) man page,** *Sun Enterprise 6x00, 5x00, 4x00, and 3x00 Systems Dynamic Reconfiguration User's Guide*, **or** *Sun Enterprise 10000 DR Configuration Guide* **for a description of how to remove the physical interface.**

▼ How to Replace a Physical Interface That Has Failed

1. **Refer to the `cfgadm`(1M) man page,** *Sun Enterprise 6x00, 5x00, 4x00, and 3x00 Systems Dynamic Reconfiguration User's Guide*, **or** *Sun Enterprise 10000 DR Configuration Guide*, **or** *Sun Fire 880 Dynamic Reconfiguration User's Guide* **for a description of how to replace the physical interface.**

2. **Plumb in and bring up the test address by typing the following command.**

```
# ifconfig hme0 <test address configuration>
```

Note – The test address configuration is the same test address that was configured in the `/etc/hostname.hme0` file. Using the previous procedure, the test configuration is the same configuration that is displayed in step 1.

This configuration triggers the `in.mpathd` daemon to resume probing. As a result of this probing, `in.mpathd` detects the repair. Consequently, `in.mpathd` causes the original IP address to fail back from `hme1`.

See "Configuring Test Addresses" on page 471 for more details about how to configure test addresses.

Note – The failback of IP addresses during the recovery of a failed physical interface requires as much as three minutes. This time might vary. The time depends on network traffic. The time also depends on the determination of the stability of the incoming interface to failback failed over interfaces by the `in.mpathd` daemon.

Recovering a Physical Interface That Was Not Present at System Boot

The steps in this section pertain to only IP layers that are configured by using `ifconfig`(1M). Layers above or below the IP layer, such as ATM or other services, require specific manual steps if the layers are not automated. The specific steps are used to unconfigure during pre-detach and configure after post-attach. See the layers and applications documentation for instructions on how to handle the failure and DR scenarios.

Recovery after a DR operation for a NIC that is part of the IO board on a Sun Fire platform or if the NIC is a cPCI device is automatic. Consequently, the following steps are not required for a NIC that is coming back as part of a DR operation. For more information on the Sun Fire x800 and Sun Fire 15000, see the `cfgadm_sbd`(1M) man page. The physical interface is recovered to the configuration that is specified in the `/etc/hostname.`*interface* file. See "Configuring Multipathing Interface Groups" on page 481 for details on how to configure interfaces to preserve the configuration across reboots.

Note – On Sun Fire legacy (Exx00) systems, DR detaches are still subject to manual procedures. However, DR attaches are automated.

You must do the following manual steps before recovering a physical interface that was not present at system boot. The following procedure uses physical interfaces `hme0` and `hme1` as example interfaces. The procedure assumes that both interfaces are in a multipathing group and that `hme0` was not present at system boot.

Note – The failback of IP addresses during the recovery of a failed physical interface last three minutes. This time might vary. The time depends on network traffic. The time also depends on the determination of the stability of the incoming interface to fail back failed-over interfaces by the `in.mpathd` daemon.

▼ How to Recover a Physical Interface That Was Not Present at System Boot

1. **Retrieve the failed network information from the console log failure error message.**

   See the `syslog`(3C)man page. The error message might be similar to the following message:

   ```
   moving addresses from failed IPv4 interfaces:
   hme1 (moved to hme0)
   ```

   The error message might also be similar to the following message:

   ```
   moving addresses from failed IPv4 interfaces:
   hme1 (couldn't move, no alternative interface)
   ```

2. **Attach the physical interface to the system.**

   Refer to the `cfgadm`(1M) man page, *Sun Enterprise 10000 DR Configuration Guide*, or *Sun Enterprise 6x00, 5x00, 4x00, and 3x00 Systems Dynamic Reconfiguration User's Guide* for a description of how to replace the physical interface.

3. **Refer to the message content from step 1. If the addresses could not be moved, go to step 5. If the addresses were moved, do step 4.**

4. **Unplumb the logical interfaces that are configured as part of the failover process by doing the following substeps.**

   a. **Look at the contents of the file /etc/hostname.<moved_from_interface> to see what logical interfaces were configured as part of the failover process.**

   b. **Unplumb each failover IP address by typing the following command:**

      ```
      # ifconfig moved_to_interface removeif moved_ip_address
      ```

Note – Failover addresses are those addresses that are marked with the `failover` parameter, or those addresses that are not marked with the `-failover` parameter. You do not need to unplumb IP addresses that are marked `-failover`.

For example, assume that the contents of the `/etc/hostname.hme0` file contained the following lines:

```
inet 1.2.3.4 -failover up group one
addif 1.2.3.5 failover up
addif 1.2.3.6 failover up
```

Then, to unplumb each failover IP address, you would type the following commands:

```
# ifconfig hme0 removeif 1.2.3.5
# ifconfig hme0 removeif 1.2.3.6
```

5. **Reconfigure the IPv4 information for the replaced physical interface by typing the following command for each interface that was removed.**

   `# ifconfig` *removed_from_NIC* `<parameters>`

   By using the example in step 4, you would type the following commands:

```
# ifconfig hme1 inet plumb
# ifconfig hme1 inet 1.2.3.4 -failover up group one
# ifconfig hme1 addif 1.2.3.5 failover up
# ifconfig hme1 addif 1.2.3.6 failover up
```

Configuring the Multipathing Configuration File

The multipathing `/etc/default/mpathd` configuration file contains three parameters that you can adjust for your configuration requirements:

- `FAILURE_DETECTION_TIME`
- `FAILBACK`
- `TRACK_INTERFACES_ONLY_WITH_GROUPS`

See “Multipathing Configuration File” on page 479 for a description of these parameters.

▼ How to Configure the Multipathing Configuration File

1. **Become superuser.**
2. **Edit the** `/etc/default/mpathd` **and change the default value of one or more of the three parameters by using one or more of the following substeps.**
   a. **Type the new value for the** `FAILURE_DETECTION_TIME` **parameter.**

      ```
      FAILURE_DETECTION_TIME=n
      ```

   b. **Type the new value for the** `FAILBACK` **parameter.**

      ```
      FAILBACK=[yes | no]
      ```

   c. **Type the new value for the** `TRACK_INTERFACES_ONLY_WITH_GROUPS` **parameter.**

      ```
      TRACK_INTERFACES_ONLY_WITH_GROUPS=[yes | no]
      ```

3. **On a command line, type the following command.**

   ```
   # pkill -HUP in.mpathd
   ```

Glossary

This glossary contains only definitions of new terms in this book that are not in the *Sun Global Glossary*. For definitions of other terms, see the *Sun Global Glossary* at http://docs.sun.com:80/ab2/coll.417.1/GLOBALGLOSS/@Ab2TocView.

address pool A set of addresses that are designated by the home network administrator for use by mobile nodes that need a home address.

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.

agent advertisement A message that is periodically sent by home agents and foreign agents to advertise their presence on any attached link.

agent discovery The process by which a mobile node determines if it has moved, its current location, and its care-of address on a foreign network.

anycast address An IP address that is assigned to more than one interface (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.

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. Diffie–Hellman 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 of a host automatically configuring its interfaces in IPv6.

bidirectional tunnel A tunnel that can transmit datagrams in both directions.

binding table	A home agent table that associates a home address with a care-of address, including remaining lifetime and time granted.
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.
care-of address	A mobile node's temporary address that is used as a tunnel exit point when the mobile node is connected to a foreign network.
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 that the individual granted the unique certificate is who she or he claims to be.
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.
digital signature	A digital code that is attached to an electronically transmitted message that uniquely identifies the sender.
DSA	Digital Signature Algorithm. A public key algorithm with a variable key size from 512 to 1024 bits. It relies on SHA-1 for input.
Diffie-Hellman protocol	Also known as public key cryptography. An asymmetric cryptographic key agreement protocol that was developed by Diffie and Hellman in 1976. The protocol enables two users to exchange a secret key over an insecure medium without any prior secrets. Diffie-Hellman is used by the IKE protocol.
dual stack	In the context of IPv6 transition, a protocol stack that contains both IPv4 and IPv6, with the rest of the stack being identical.
encapsulating security header	An extension header that provides integrity and confidentiality to datagrams.
encapsulation	The process of a header and payload being placed in the first packet, which is subsequently placed in the second packet's payload.
failback	The process of switching back network access to an interface that has its repair detected.
failover	The process of switching network access from a failed interface to a good physical interface. Network access includes IPv4 unicast, multicast, and broadcast traffic, as well as IPv6 unicast and multicast traffic.
failure 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.
firewall	Any device or software that protects an organization's private network or intranet from intrusion by external networks such as the Internet.

foreign agent	A router or server on the foreign network that the mobile node visits.
foreign network	Any network other than the mobile node's home network.
forward tunnel	A tunnel that starts at the home agent and terminates at the mobile node's care-of address.
Generic Routing Encapsulation (GRE)	An optional form of tunneling that can be supported by home agents, foreign agents, and mobile nodes. GRE enables a packet of any network-layer protocol to be encapsulated within a delivery packet of any other (or the same) network-layer protocol.
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.
HMAC	Keyed hashing method for message authentication. 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.
home address	An IP address that is assigned for an extended period to a mobile node. The address remains unchanged when the node is attached elsewhere on the Internet or an organization's network.
home agent	A router or server on the home network of a mobile node.
home network	A network that has a network prefix that matches the network prefix of a mobile node's home address.
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.
IKE	Internet Key Exchange. IKE automates the provision of authenticated keying material for IPsec security associations.
IP-in-IP encapsulation	The Internet-standard protocol for tunneling IPv4 packets within IPv4 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.
IPsec	The security architecture (IPsec) that provides protection for IP datagrams.
IPv4	Internet Protocol, version 4. Sometimes referred to as IP. This version supports a 32–bit address space.

IPv6	Internet Protocol, version 6. This version supports a 128–bit address space.
key management	The way in which you manage security associations.
link-local-use address	A designation that is used for addressing on a single link for purposes such as automatic address configuration.
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.
MD5	An iterative cryptographic hash function that is used for message authentication, including digital signatures. The function was developed in 1991 by Rivest.
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.
mobile node	A host or router that can change its point of attachment from one network to another network while maintaining all existing communications by using its IP home address.
mobility agent	Either a home agent or a foreign agent.
mobility binding	The association of a home address with a care-of address, along with the remaining lifetime of that association.
mobility security association	A collection of security measures, such as an authentication algorithm, between a pair of nodes, which are applied to Mobile IP protocol messages that are exchanged between the two nodes.
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 IP 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.
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 Access Identifier (NAI)	A designation that uniquely identifies the mobile node in the format of user@domain.
network interface card (NIC)	Network adapter that is either internal or a separate card that serves as an interface to a link.
node	A host or a router.
packet	A group of information that is transmitted as a unit over communications lines. Contains a header plus payload.
physical interface	A node's attachment to a link. This attachment is often implemented as a device driver plus a network adapter. Some network adapters can have multiple points of attachment, for example, qfe. The usage of *network adapter* in this document refers to a "single point of attachment."
physical interface group	The set of physical interfaces on a system that are connected to the same link. These interfaces are identified by assigning the same (non-null) character string name to all the physical interfaces in the group.
physical interface group name	A name that is assigned to a physical interface that identifies the group. The name is local to a system. Multiple physical interfaces, sharing the same group name, form a physical interface group.
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.
public key cryptography	A cryptographic system that uses two keys - a public key known to everyone and a private key 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.
registration	The process by which a mobile node registers its care-of address with its home agent and foreign agent when it is away from home.
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.
reverse tunnel	A tunnel that starts at the mobile node's care-of address and terminates at the home agent.
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.
SADB	Security Associations Database. A table that specifies cryptographic keys and algorithms that are used in the transmission of data.
security associations	Associations that specify security properties from one host to another.
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.
SHA-1 algorithm	Secure Hashing Algorithm. The algorithm operates on any input length less than 2^{64} to produce a message digest. It is input to DSA.
site-local-use address	A designation that is used for addressing on a single site.
SPI	Security Parameters Index. An integer that specifies the row in the SADB that a receiver should use to decrypt a received packet.
standby	A physical interface that is not used to carry data traffic unless some other physical interface has failed.
stateful autoconfiguration	The process of a host obtaining interface addresses, configuration information, and parameters from a server.
stateless autoconfiguration	The process of a host generating its own addresses by using a combination of locally available information and information that is advertised by routers.
symmetric key cryptography	An encryption system in which the sender and receiver of a message share a single, common key that 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.
Triple-DES	Triple-Data Encryption Standard. A symmetric-key encryption method which provides a key length of 168 bits.
tunnel	The path that is followed by a datagram while it is encapsulated.
tunneling	The mechanism by which IPv6 packets are placed inside IPv4 packets and routed through the IPv4 routers. The term is specific to IPv6 only.
unicast address	An IP address that identifies a single interface.
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

B

C

D

E

F

G

H

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z