IPv6 Administration Guide

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

Part No: 817–0573–10
April 2003

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Adobe PostScript

030224@5533

Contents

Preface 7

1 IPv6 (Overview) 11

IPv6 Features 11

IPv6 Header and Extensions 12

Header Format 12

Extension Headers 13

IPv6 Addressing 14

Unicast Addresses 16

Aggregate Global Unicast Addresses 16

Local-Use Addresses 17

IPv6 Addresses With Embedded IPv4 Addresses 18

Anycast Addresses 19

Multicast Addresses 19

IPv6 Routing 20

IPv6 Neighbor Discovery 21

Router Advertisement 22

Router Advertisement Prefixes 22

Router Advertisement Messages 23

Neighbor Solicitation and Unreachability 23

Comparison With IPv4 24

IPv6 Stateless Address Autoconfiguration 26

Stateless Autoconfiguration Requirements 26

Stateful Autoconfiguration Model 26

When to Use Stateless and Stateful Approaches 27

Duplicate Address Detection Algorithm 27

Autoconfiguration Process 27
IPv6 Mobility Support 29
IPv6 Quality-of-Service Capabilities 30
Flow Labels 30
Traffic Class 32
IPv6 Security Improvements 32
6to4 Tunnels Over IPv4 Networks 33
For More Information About 6to4 Routing 34

2 Administering IPv6 (Tasks) 35
Enabling IPv6 Nodes 35
Enabling IPv6 Nodes Task Map 36
▼ How to Enable IPv6 on a Node 36
▼ How to Configure a Solaris IPv6 Router 37
▼ How to Configure a 6to4 Router 38
▼ How to Add IPv6 Addresses to NIS and NIS+ 41
▼ How to Add IPv6 Addresses to DNS 42
Monitoring IPv6 43
Monitoring IPv6 Task Map 43
▼ How to Display Interface Address Assignments 44
▼ How to Display Network Status 45
▼ How to Control the Display Output of IPv6 Related Commands 49
▼ How to Monitor Only IPv6 Network Traffic 50
▼ How to Probe All Multi-homed Host Addresses 50
▼ How to Trace All Routes 51
Configuring IP in IP Tunnels 51
Configuring IP in IP Tunnels Task Map 52
▼ How to Configure IPv6 Over IPv4 Tunnels 53
▼ How to Configure IPv6 Over IPv6 Tunnels 54
▼ How to Configure IPv4 Over IPv6 Tunnels 55
▼ How to Configure IPv4 Over IPv4 Tunnels 55
▼ How to Configure Your Router to Advertise Over Tunneling Interfaces 56
▼ How to Configure a 6to4 Tunnel to a 6to4 Relay Router 57
Displaying IPv6 Name Service Information 58
Displaying IPv6 Name Service Information Task Map 59
▼ How to Display IPv6 Name Service Information 59
▼ How to Verify That DNS IPv6 PTR Records Are Updated Correctly 60

▼ How to Display IPv6 Information Through NIS 60
▼ How to Display IPv6 Information Through NIS+ 61
▼ How to Display IPv6 Information Independent of Name Service 61

3 IPv6 Files and Commands (Reference) 63
Overview of the Solaris IPv6 Implementation 63
IPv6 Network Interface Configuration File 64
IPv6 Interface Configuration File Entry 65
Utilities Added or Modified for IPv6 65
IPv6 Extensions to the `ifconfig` Utility 66
`6to4relay` Command 68
Nodes With Multiple Network Interfaces 69
IPv4 Behavior 69
IPv6 Behavior 69
IPv6 Daemons 70
`in.ndpd` Daemon 70
`in.ripngd` Daemon 72
`inetd` Internet Services Daemon 73
IPv6 Extensions to Existing Utilities 74
`netstat`(1M) 75
`snoop`(1M) 75
`route`(1M) 75
`ping`(1M) 75
`traceroute`(1M) 76
Controlling Display Output 76
Solaris Tunneling Interfaces for IPv6 76
IPv6 Extensions to Solaris Name Services 79
`/etc/inet/ipnodes` File 79
NIS Extensions for IPv6 80
NIS+ Extensions for IPv6 80
DNS Extensions for IPv6 80
Changes to the `nsswitch.conf` File 80
Changes to Name Service Commands 82
NFS and RPC IPv6 Support 82
IPv6 Over ATM Support 82

4 Making the Transition From IPv4 to IPv6 (Reference) 83

Transition Requirements 83

Standardized Transition Tools 84

Implementing Dual-Stack 84

Configuring Name Services 85

Using IPv4-Compatible Address Formats 86

Tunneling Mechanism 87

Interaction With Applications 88

IPv4 and IPv6 Interoperability 89

Site Transition Scenarios 90

6to4 as a Transition Mechanism 90

Participants in a 6to4 Tunnel 91

6to4-Derived Addressing 93

Packet Flow Through the 6to4 Tunnel 95

Considerations for Tunnels to a 6to4 Relay Router 95

Known Issues With 6to4 Router 98

Other Transition Mechanisms 98

Glossary 101

Index 107

Preface

IPv6 Administration Guide provides information about configuring and managing the IPv6 framework that is installed in your Solaris™ operating environment. This book assumes that you have already installed the SunOS™ 5.9 operating system. This book also assumes that 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 x86. 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 1 provides an overview of the new Internet Protocol that is known as IPv6.

Chapter 2 provides procedures for enabling IPv6 at your site. These procedures describe how to configure IPv6 routers, 6to4 routers, and IPv6 addresses for name services. Additional procedures show how to create various types of tunnels between routers, how to run IPv6 additions to commands for diagnostics, and how to display IPv6 name service information.

Chapter 3 describes the concepts that are associated with the Solaris implementation of IPv6.

Chapter 4 provides an overview of the approach and the standardized solutions for making the transition from IPv4 to IPv6 at your site.

The Glossary provides definitions of key IP Services terms.

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

IPv6 (Overview)

The Internet Protocol, version 6 (IPv6), is a new version of Internet Protocol (IP). IPv6 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" on page 11
- "IPv6 Header and Extensions" on page 12
- "IPv6 Addressing" on page 14
- "IPv6 Routing" on page 20
- "IPv6 Neighbor Discovery" on page 21
- "IPv6 Stateless Address Autoconfiguration" on page 26
- "IPv6 Mobility Support" on page 29
- "IPv6 Quality-of-Service Capabilities" on page 30
- "IPv6 Security Improvements" on page 32
- "6to4 Tunnels Over IPv4 Networks" on page 33

IPv6 Features

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. In addition, IPv6 provides a greater number of addressable nodes. IPv6 also 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** – This capability enables the labeling of packets that belong to particular traffic *flows* for which the sender requests special handling. For example, the sender can request non-default 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.

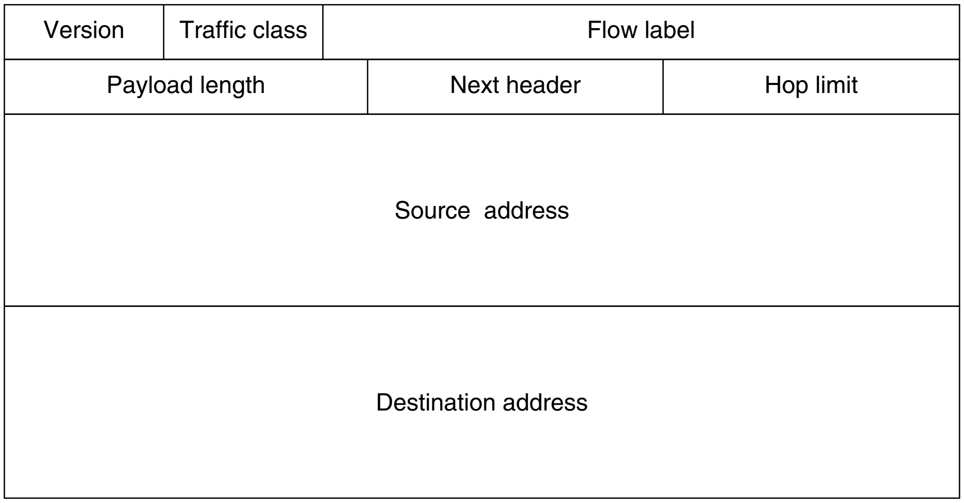

FIGURE 1–1 IPv6 Header Format

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 32.
- **Flow Label** – 20-bit field. See "IPv6 Quality-of-Service Capabilities" on page 30.
- **Payload Length** – 16-bit unsigned integer, which is the rest of the packet that follows the IPv6 header, in octets.
- **Next Header** – 8-bit selector. Identifies the type of header that immediately follows the IPv6 header. Uses the same values as the IPv4 protocol field. See "Extension Headers" on page 13.
- **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 14.
- **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.

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. IPv6 addresses are identifiers for individual interfaces and for sets of interfaces. You can assign IPv6 addresses of all types to interfaces, rather than hosts and routers as in IPv4. Because each interface belongs to a single node, any of the interface's unicast addresses 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, that is, 128 in contrast to 32. Thus, the number of potential addresses is 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 the leading bits is called the format prefix (FP). The following table shows the initial allocation of these prefixes.

TABLE 1–1 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
Link Local Use Addresses	1111 1110 10	1/1024
Site Local Use Addresses	1111 1110 11	1/1024

TABLE 1–1 Format Prefix Allocations *(Continued)*

Allocation	Prefix (Binary)	Fraction of Address Space
Multicast Addresses	1111 1111	1/256
Reserved for 6to4 Router Address	1000 0000 0000 10	

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. The remaining address space can be used for expansion of existing use. For example, the space can be used for additional aggregate global unicast addresses. The remaining space can also be used for new uses. For example, the space can be used for separate locators and separate 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 1–2 Aggregate Global Unicast Addresses Format

3 bits	13 bits	8 bits	24 bits	16 bits	64 bits

TABLE 1–2 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. In many instances, an interface identifier 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. These addresses are also used 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 address is for a single data link layer medium, such as an Ethernet or Frame Relay link. The Site-Local-Use is for use on a single site. The following table shows the Link-Local-Use address format.

TABLE 1–3 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 1–4 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 1–5 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 1–6 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.

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, the packet 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 1–7 Multicast Address Format

8 bits	4 bits	4 bits	112 bits
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. These flags must be initialized to zero.

- **T=0** – Indicates a permanently assigned, *well-known*, multicast address. This multicast address is assigned by the global Internet authority that assigns numbers.
- **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 1–8 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 that is based on policy, performance, cost, and so on
- Host mobility, route to current location
- Auto-readdressing, route to new address

You obtain the new routing 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 traffic destination neighbor. 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. One type is a pair of router solicitation and router advertisement messages. Another type is a pair of neighbor solicitation and neighbor advertisement messages. The fifth type is 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. The redirect informs the sender that the destination is a neighbor.

Router advertisements, and per-prefix flags, enable routers to inform hosts how to perform 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, that packets are 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*. 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 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 accepts packets on behalf of a target address can issue non-override neighbor advertisements. The router can accept packets for a target address that is unable to respond to neighbor solicitations. Currently, the use of proxy is not specified. However, proxy advertising can potentially be used to handle cases 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 neighbor discovery protocol of IPv6 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 message assumes that the new next-hop is on-link. In IPv4, a host ignores redirect messages 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, nodes should not 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 by using neighbor unreachability detection. Neighbor discovery 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 dead 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. The ability to identify routers 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) redirect messages. IPv4 off-link nodes can also 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 enables a host to generate its own addresses. The stateless mechanism uses local information as well as non-local information that is advertised by routers to generate the addresses. 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 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. The addresses must also 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.

Autoconfiguration Process

This section provides an overview of the typical steps that are performed by an interface during autoconfiguration. Autoconfiguration is performed only on multicast-capable links. 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. 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 need not be tested individually. The addresses must be created from the same interface identifier. In contrast, all addresses that are obtained manually should be tested individually for uniqueness. The same is true for addresses that are obtained by stateful address autoconfiguration. 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, 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, a mobile node can change its IP address each time that 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 link 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. The mobile node's current point of attachment to the Internet does not matter. 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. Active flows can also 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. New flow labels must also be chosen 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 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 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 the 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 of the 8 bits.
- Nodes that support some of the Traffic Class bits 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 beneath 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 that is 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, ESP remains flexible and is algorithm independent. Similar security protocols include SP3D and ISO NLSP.

IPv6 Authentication Header and IPv6 Encapsulating Security Payload are features of the new Internet Protocol Security (IPsec). For an overview of IPsec, see “IPsec (Overview)” in *System Administration Guide: IP Services*. For a description, of how you implement IPsec, see “Implementing IPsec Task Map” in *System Administration Guide: IP Services*

6to4 Tunnels Over IPv4 Networks

To better enable the transition from IPv4 to IPv6, the Solaris operating system now supports the *6to4 transition mechanism*. This mechanism enables the tunneling of packets from an isolated 6to4 IPv6 site across IPv4 networks to another isolated IPv6 site. 6to4 enables you to use one global IPv4 address and derive a complete 48-bit IPv6 global prefix for your site. To obtain technical information about 6to4 routing, refer to RFC 3056, "Connection of IPv6 Domains via IPv4 Clouds".

Consider implementing 6to4 if either or both of the following conditions exist at your IPv6 site:

- Your IPv6 site does not have an IPv6 connection to the Internet.
- Your isolated IPv6 site needs to communicate with another isolated IPv6 site

In the past, isolated IPv6 sites could not communicate with other IPv6 sites. 6to4 routing enables the isolated sites to transfer packets through a tunnel over an IPv4 network. The only requirement is a boundary router with a globally unique IPv4 address for the interface that connects to the IPv4 network.

You can configure more than one interface on a router for 6to4 support, provided that the interface meets the previously mentioned requirements. You do not need to manually configure hosts for 6to4 support. On receipt of a prefix advertisement from the 6to4 router, an IPv6 host automatically reconfigures an IPv6 interface with a 6to4 address.

The router encapsulates outbound IPv6 packets in an IPv4 header. An automatic tunnel is then constructed between the 6to4 router and the destination 6to4 site, over an IPv4 network. On receipt, the remote 6to4 router decapsulates the packets. The remote 6to4 router delivers the now standard IPv6 packets to the appropriate IPv6 nodes.

The 6to4 router can also tunnel packets to a native IPv6, non-6to4 site. In this instance, the 6to4 router sets up a tunnel over an IPv4 network to a 6to4 relay router. The relay router forwards the packets to an IPv6 network.

For More Information About 6to4 Routing

Configuring 6to4 routing is a straightforward task. The next table shows where to go for 6to4 configuration tasks and technical details.

Task or Detail	For Information
To configure a 6to4 router	"How to Configure a 6to4 Router" on page 38
To configure a 6to4 relay router	"How to Configure a 6to4 Tunnel to a 6to4 Relay Router" on page 57
For 6to4 modifications to `ifconfig`	"IPv6 Extensions to the `ifconfig` Utility" on page 66
For information about the `6to4relay` command	"`6to4relay` Command" on page 68
For details about 6to4 site topology	"Participants in a 6to4 Tunnel" on page 91
For details about 6to4 addressing	"6to4-Derived Addressing" on page 93
For detailed information about 6to4 relay routers	"Considerations for Tunnels to a 6to4 Relay Router" on page 95

CHAPTER 2

Administering IPv6 (Tasks)

This chapter contains procedures for configuring IPv6 on hosts and routers. The text includes the following major topics:

- "Enabling IPv6 Nodes" on page 35
- "Enabling IPv6 Nodes Task Map" on page 36
- "Monitoring IPv6" on page 43
- "Monitoring IPv6 Task Map" on page 43
- "Configuring IP in IP Tunnels" on page 51
- "Configuring IP in IP Tunnels Task Map" on page 52
- "Displaying IPv6 Name Service Information" on page 58
- "Displaying IPv6 Name Service Information Task Map" on page 59

For More Information	Go to
Overview of IPv6	Chapter 1
Detailed conceptual information that is related to the IPv6 tasks in this chapter	Chapter 3
Details about making the transition from IPv4 to IPv6	Chapter 4

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 host.

Enabling IPv6 Nodes Task Map

TABLE 2–1 Enabling IPv6 Nodes Task Map

Task	Description	For Instructions
Enable IPv6 on a node	Involves touching `hostname6.`*interface* file, displaying addresses, and entering the addresses in the `/etc/inet/ipnodes` file. See the note that follows this table.	"How to Enable IPv6 on a Node" on page 36
Configure a Solaris IPv6 router	Involves adding entries to the `indp.conf` file.	"How to Configure a Solaris IPv6 Router" on page 37
Configure a 6to4 router	Involves configuring a 6to4 interface in the `/etc/hostname6.ip.6to4tun0` file	"How to Configure a 6to4 Router" on page 38
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 41
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 42

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. **Type the following for each interface.**

```
# touch /etc/hostname6.interface
```

Interface	Interface name, such as `le0`, `le1`.

3. **Reboot the node**

Note – The reboot process sends router discovery packets. The router responds with a prefix. The response enables the node to configure the interfaces with an IPv6 address. Rebooting also restarts key network daemons in IPv6 mode.

4. **Display the IPv6 addresses of all interfaces.**

   ```
   # ifconfig -a6
   ```

   To show both IPv4 and IPv6 addresses, just use the `-a` option.

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 41.
   - For DNS, see "How to Add IPv6 Addresses to DNS" on page 42.

▼ How to Configure a Solaris IPv6 Router

1. **Become superuser on the system to become the 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 Configure a 6to4 Router

Before you configure 6to4 routing on an IPv6 network, you must have done the following:

- Configured IPv6 on all appropriate nodes at the prospective 6to4 site, as described in "How to Enable IPv6 on a Node" on page 36.
- Selected at least one router with a connection to an IPv4 network to become the 6to4 router.
- Configured a globally unique IPv4 address for the prospective 6to4 router's interface to the IPv4 network. The IPv4 address must be static. Do not use a dynamically allocated IPv4 address. For instructions for configuring an IPv4 interface, see "Network Configuration Procedures" in *System Administration Guide: IP Services*

1. **Become superuser on the prospective 6to4 router.**

2. **Configure a 6to4 pseudo-interface on the router by creating the `/etc/hostname6.ip.6to4tun0` file.**

   - If you plan to use the recommended convention of subnet ID=0 and host ID=1, use the short format for `/etc/hostname6.ip.6to4tun0`:

     `tsrc` *IPv4 address* `up`

   - If you plan to use other conventions for subnet ID and host ID, use the long format for `/etc/hostname6.ip.6to4tun0`:

     `tsrc` *IPv4 address* `2002:`*IPv4 address*`:`*subnet ID*`:`*host ID*`:/64 up`

   The next table explains the parameters of `/etc/hostname6.ip.6to4tun0` that are used in this step.

tsrc	Indicates that this interface is used as a tunnel source
IPv4 address	Specifies, in dotted decimal format, the IPv4 address that is configured on the physical interface to become the 6to4 pseudo-interface
2002	(Optional) Specifies the 6to4 prefix
IPv4 address	(Optional) Specifies, in hexadecimal, the IPv4 address of the pseudo-interface.
subnet ID	(Optional) Specifies, in hexadecimal notation, a subnet ID other than 0
host ID	(Optional) Specifies a host ID other than 1
/64	Indicates that the address has a length of 64 bits.
up	Directs `ifconfig` to configure the interface as "up"

Note – Two IPv6 tunnels cannot have the same source and same destination addresses. Packets are dropped as a result. This type of event can happen if a 6to4 router also performs tunneling through `atun`.

3. **(Optional) Create additional 6to4 pseudo-interfaces on the router.**

   Each prospective 6to4 pseudo-interface must have an already-configured, globally unique IPv4 address.

4. **Run `ifconfig` to verify the status of the interface.**

   ```
   # ifconfig ip.6to4tun0 inet6
   ```

   If the interface is correctly configured, you receive output that is similar to the following.

   ```
   ip.6to4tun0: flags=2200041<UP,RUNNING,NONUD,IPv6>mtu 1480 index 11
           inet tunnel src 129.146.87.188
           tunnel hop limit 60
           inet6 2002:8192:56bb:56bb::1/64
   ```

5. **Edit `/etc/inet/ndpd.conf` to advertise 6to4 routing.**

   For detailed information, refer to `ndpd.conf`(4).

   a. **Specify the subnet to receive the advertisement in the first line.**

      Create an `if` entry with the following format:

```
if subnet-interface AdvSendAdvertisements 1
```

For example, to advertise 6to4 routing to the subnet that is connected to interface `hme0`, replace *subnet-interface* with `hme0`.

```
if hme0 AdvSendAdvertisements 1
```

b. **Add the 6to4 prefix as the second line of the advertisement.**

Create a `prefix` entry with following format:

```
prefix 2002:IPv4-address:subnet ID::/64  subnet-interface
```

prefix 2002	Indicates a 6to4 prefix
IPv4-address	Specifies, in hexadecimal representation, the unique IPv4 address of the 6to4 pseudo-interface
subnet ID	Specifies, in hexadecimal, a subnet ID
subnet-interface	Router interface to which the subnet is connected

6. **Reboot the router.**

   Alternatively, you can issue a `sighup` to `/etc/inet/in.ndpd` to begin sending router advertisements.. The IPv6 nodes on each subnet to receive the 6to4 prefix now autoconfigure with new 6to4–derived addresses.

7. **Add the new 6to4–derived addresses of the nodes to the name service that is used at the 6to4 site.**

   For instructions, go to “How to Add IPv6 Addresses to DNS” on page 42 or “How to Add IPv6 Addresses to NIS and NIS+” on page 41.

Configuring Multiple Routers at the 6to4 Site

For a multiple router site, the routers behind the 6to4 router might require further configuration to support 6to4. If your site uses RIP, you must configure on each non-6to4 router static routes to the 6to4 router. If you use a commercial routing protocol, you do not need to create static routes to the 6to4 router.

Examples—6to4 Router Configuration

The following is an example of the short form of `/etc/hostname6.ip.6to4tun0`.

```
# cat /etc/hostname6.ip.6to4tun0
tsrc 129.146.87.188 up
```

Here is an example of the long form of `/etc/hostname6.ip.6to4tun0`.

```
# cat /etc/hostname6.ip.6to4tun0
tsrc 129.146.87.188 2002:8192:56bb::1/64 up
```

The following example shows output of `ifconfig` for a 6to4 pseudo-interface.

```
# ifconfig ip.6to4tun0 inet6
ip.6to4tun0: flags=2200041<UP,RUNNING,NONUD,IPv6> mtu 1480 index 11
        inet tunnel src 129.146.87.188
        tunnel hop limit 60
        inet6 2002:8192:56bb::1/64
```

The next sample `/etc/inet/ndpd.conf` file advertises 6to4 routing on two subnets.

```
if qfe0 AdvSendAdvertisements 1
prefix  2002:8192:56bb:1::/64 qfe0

if qfe1 AdvSendAdvertisements 1
prefix  2002:8192:56bb:2::/64 qfe1
```

▼ 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 79 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 to administer 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 command 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, are unchanged so that they can 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 the primary or secondary DNS server for your IPv6 network.**
2. **Edit the appropriate DNS zone file by adding AAAA records for each IPv6–enabled host, as follows.**

   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*

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

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.0.6.5.2.9.0.0.0.0.0.0.0.0.0.2.0.0.0 \
    IN        PTR        vallejo.Eng.apex.COM.
8.2.5.0.2.1.e.f.f.f.9.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.0.0.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 66 and "IPv6 Extensions to Existing Utilities" on page 74.

Monitoring IPv6 Task Map

TABLE 2–2 Monitoring IPv6 Task Map

Task	Description	For Instructions, Go to ...
Display interface address assignments	Displays all address assignments, or just IPv4, IPv6, or 6to4-derived address assignments by using `ifconfig` command.	"How to Display Interface Address Assignments" on page 44
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 45
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 49
Monitor only IPv6 network traffic	Displays all IPv6 packets by using the `snoop` command.	"How to Monitor Only IPv6 Network Traffic" on page 50
Probe all multi-homed host addresses	Checks all addresses by using the `ping` command.	"How to Probe All Multi-homed Host Addresses" on page 50

TABLE 2–2 Monitoring IPv6 Task Map *(Continued)*

Task	Description	For Instructions, Go to ...
Trace all routes	Uses the `traceroute` command.	"How to Trace All Routes" on page 51

▼ 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
ip.6to4tun0: flags=2200041<UP,RUNNING,NONUD,IPv6> mtu 1480 index 11
        inet tunnel src 129.146.86.187
        tunnel hop limit 60
        inet6 2002:8192:56bb::1/64
```

Example—Displaying Addressing Information for the 6to4 Pseudo-Interface

```
% ifconfig ip.6to4tun0 inet6
6to4tun0: flags=2200041<UP,RUNNING,NONUD,IPv6. mtu 1480 index 11
        inet tunnel src 172.16.213.81
        tunnel hop limit 60
        inet6 2002:ac10:d551::1/64
```

▼ 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 or 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 or 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
                .
                .
ICMPv6 for ip.6to4tun0
        icmp6InMsgs         =    57     icmp6InErrors       =     0
        icmp6InDestUnreachs =    57     icmp6InAdminProhibs =     0
        icmp6InTimeExcds    =     0     icmp6InParmProblems =     0
        icmp6InPktTooBigs   =     0     icmp6InEchos        =     0
        icmp6InEchoReplies  =     0     icmp6InRouterSols   =     0
        icmp6InRouterAds    =     0     icmp6InNeighborSols =     0
        icmp6InNeighborAds  =     0     icmp6InRedirects    =     0
        icmp6InBadRedirects =     0     icmp6InGroupQueries =     0
        icmp6InGroupResps   =     0     icmp6InGroupReds    =     0
        icmp6InOverflows    =     0
        icmp6OutMsgs        =    57     icmp6OutErrors      =     0
        icmp6OutDestUnreachs=    57     icmp6OutAdminProhibs=     0
        icmp6OutTimeExcds   =     0     icmp6OutParmProblems=     0
        icmp6OutPktTooBigs  =     0     icmp6OutEchos       =     0
        icmp6OutEchoReplies =     0     icmp6OutRouterSols  =     0
        icmp6OutRouterAds   =     0     icmp6OutNeighborSols=     0
        icmp6OutNeighborAds =     0     icmp6OutRedirects   =     0
        icmp6OutGroupQueries=     0     icmp6OutGroupResps  =     0
        icmp6OutGroupReds   =     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
ping -afe80::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 Multi-homed 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 IP in IP Tunnels

This section describes how you manually configure IP in IP tunnels. For information on configuring an automatic 6to4 tunnel, refer to “How to Configure a 6to4 Router” on page 38.

IPv6 supports the following types of tunnel encapsulation:

- IPv4 over IPv4 tunnels

- IPv6 over IPv4 tunnels (native IPv6 and 6to4)
- IPv6 over IPv6 tunnels
- IPv4 over IPv6 tunnels
- IPv6 to IPv6 over 6to4 tunnels
- 6to4 site tunnel to 6to4 relay router to native IPv6 site

For conceptual descriptions of tunnels, see "Solaris Tunneling Interfaces for IPv6" on page 76 and "Tunneling Mechanism" on page 87.

Configuring IP in IP Tunnels Task Map

TABLE 2–3 Configuring IP in IP Tunnels Task Map

Task	Description	Instructions
Manually configuring IPv6 over IPv4 tunnels	Shows the entries required for the `hostname6.ip.tun`*n* file.	"How to Configure IPv6 Over IPv4 Tunnels" on page 53
Automatically configuring IPv6 over IPv4 tunnels (6to4 tunnels)	Shows entries required for the `hostname6.ip.6to4tun` file.	"How to Configure a 6to4 Router" on page 38
Configuring IPv6 over IPv6 tunnels	Shows the entries required for the `hostname6.ip.6.tun`*n* file.	"How to Configure IPv6 Over IPv6 Tunnels" on page 54
Configuring IPv4 over IPv6 tunnels	Shows the entries required for the `hostname.ip6.tun`*n* file.	"How to Configure IPv4 Over IPv6 Tunnels" on page 55
Configuring IPv4 over IPv4 tunnels	Shows the entries required for the `hostname.ip.tun`*n* file.	"How to Configure IPv4 Over IPv4 Tunnels" on page 55
Configuring a tunnel between a 6to4 router and a 6to4 relay router	Describes how to enable a tunnel to a 6to4 relay router by using the `6to4relay` command.	"How to Configure a 6to4 Tunnel to a 6to4 Relay Router" on page 57
Configuring your router to advertise over tunneling interfaces	Shows the entries required for the `/etc/inet/ndpd.conf` file.	"How to Configure Your Router to Advertise Over Tunneling Interfaces" on page 56

▼ How to Configure IPv6 Over IPv4 Tunnels

Note – The best way to pass IPv6 packets over an IPv4 network is to use an automatic 6to4 tunnel. For instructions for implementing 6to4 routing at your site, refer to "How to Configure a 6to4 Router" on page 38.

---

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 by 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 56.

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 IPv6 Over IPv6 Tunnels

1. **Become Superuser.**

2. **Create the file `/etc/hostname6.ip6.tun`*n*. Use the values 0, 1, 2, and so on, for *n*. Then, add entries by following these steps.**

   a. **Add the tunnel source address. Then, add the tunnel destination address.**

   ```
   tsrc IPv6-source-address tdst IPv6-destination-address
   IPv6-packet-source-address IPv6-packet-destination-address 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 56.

Example—Entry in the IPv6 Configuration File to Create an IPv6 over IPv6 Tunnel

This example shows the entry for an IPv6 over IPv6 tunnel.

```
tsrc 2000::114:a00:20ff:fe72:668c tdst 2000::103:a00:20ff:fe9b:a1c3
fe80::4 fe80::61 up
```

▼ How to Configure IPv4 Over IPv6 Tunnels

1. **Become Superuser.**
2. **Create the file `/etc/hostname.ip6.tun`*n*. Use the values 0, 1, 2, and so on, for *n*. Then, add entries by following these steps.**

   a. **Add the tunnel source address. Then, add the tunnel destination address.**

   `tsrc` *IPv6-source-address* `tdst` *IPv6-destination-address*
   tunnel-IPv4-source-address *tunnel-IPv4-destination-address* `up`

   b. **(Optional) Add a logical interface for the source and destination IPv6 addresses.**

   `addif` *IPv6-source-address* *IPv6-destination-address* `up`

   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 56.

Example—Entry in the IPv4 Configuration File to Create an IPv4 Over IPv6 Tunnel

This example shows the entry for an IPv4 over IPv6 tunnel.

```
tsrc 2000::114:a00:20ff:fe72:668c tdst 2000::103:a00:20ff:fe9b:a1c3
10.0.0.4 10.0.0.61 up
```

▼ How to Configure IPv4 Over IPv4 Tunnels

1. **Become Superuser.**
2. **Create the file `/etc/hostname.ip.tun`*n*. Use the values 0, 1, 2, and so on, for *n*. Then, add entries by following these steps.**

   a. **Add the tunnel source address. Then, add the tunnel destination address.**

   `tsrc` *IPv4-source-address* `tdst` *IPv4-destination-address*
   tunnel-IPv4-source-address *tunnel-IPv4-destination-address* `up`

   b. **(Optional) Add a logical interface for the source and destination IPv4 addresses.**

```
addif IPv4-source-address    IPv4-destination-address up
```

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 56.

Example—Entry in the IPv4 Configuration File to Create an IPv4 Over IPv4 Tunnel

This example shows the entry for an IPv4 over IPv4 tunnel.

```
tsrc 120.46.86.158 tdst 120.46.86.122
10.0.0.4 10.0.0.61 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.tunn AdvSendAdvertisements 1
   ```

   b. **Add prefixes as needed.**

   ```
   prefix interface-address ip.tunn
   ```

3. **Reboot.**

▼ How to Configure a 6to4 Tunnel to a 6to4 Relay Router

Caution – Because of major security issues, by default 6to4 relay router support is disabled in the Solaris operating system. See "Considerations for Tunnels to a 6to4 Relay Router" on page 95 and Internet Draft, *Security Considerations for 6to4*.

Before you enable a tunnel to a 6to4 relay router, you must have completed the following tasks:

- Configured a 6to4 router at your site, as explained in "How to Configure a 6to4 Router" on page 38.
- Reviewed the security issues that are involved in tunneling to a 6to4 relay router.

1. **Become superuser on the 6to4 router.**

2. **Enable a tunnel to the 6to4 relay router by using either of the following formats:**

   - Enable a tunnel to an anycast 6to4 relay router.

     ```
     # /usr/sbin/6to4relay -e
     ```

     The -e option sets up a tunnel between the 6to4 router and an anycast 6to4 relay router. Anycast 6to4 relay routers have the well-known IPv4 address 192.88.99.1. The anycast relay router that is physically nearest to your site becomes the endpoint for the 6to4 tunnel. This relay router then handles packet forwarding between your 6to4 site and a native IPv6 site.

     For detailed information about anycast 6to4 relay routers, refer to RFC 3068, "An Anycast Prefix for 6to4 Relay Routers"

   - Enable a tunnel to a specific 6to4 relay router.

     ```
     # /usr/sbin/6to4relay -e -a relay-router-address
     ```

     The -a option indicates that a specific router address is to follow. Replace *relay-router-address* with the IPv4 address of the specific 6to4 relay router with which you want to enable a tunnel.

   The tunnel to the 6to4 relay router remains active until you remove the 6to4 tunnel pseudo-interface.

3. **Delete the tunnel to the 6to4 relay router, when no longer needed, by typing the following:**

   ```
   # /usr/sbin/6to4relay -d
   ```

4. **(Optional) Make the tunnel to the 6to4 relay router persistent across reboots.**

Your site might have a compelling reason to have the tunnel to the 6to4 relay router reinstated each time the 6to4 router reboots. To support this scenario, you must do the following.

a. **Edit the`/etc/default/inetinit` file.**

   The line that you need to modify is at the end of the file.

b. **Change the "NO" value in the line `ACCEPT6TO4RELAY=NO` to "YES."**

c. **(Optional) Create a tunnel to a specific 6to4 relay router that persists across reboots.**

   For the parameter `RELAY6TO4ADDR`, change the address 192.88.99.1 to the IPv4 address of the 6to4 relay router that you want to use.

Examples—Getting Status Information About 6to4 Relay Router Support

You can use `/usr/bin/6to4relay` to find out whether support for 6to4 relay routers is enabled. The next example shows the output when support for 6to4 relay routers is disabled, as is the default in the Solaris operating system:

```
# /usr/sbin/6to4relay
6to4relay: 6to4 Relay Router communication support is disabled.
```

When support for 6to4 relay routers is enabled, you receive the following output:

```
# /usr/sbin/6to4relay
6to4relay: 6to4 Relay Router communication support is enabled.
IPv4 destination address of Relay Router=192.88.99.1
```

Displaying IPv6 Name Service Information

This section provides procedures to display IPv6 name service information.

Displaying IPv6 Name Service Information Task Map

TABLE 2–4 Displaying IPv6 Name Service Information Task Map

Task	Description	For Information
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 59
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 60
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 60
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 61
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 61

▼ 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.0.6.5.2.9.0.0.0.0.0.0.0.0.2.0.0.0.ip6.int

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

▼ How to Display IPv6 Information Through NIS

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

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

```
% ypmatch host-name ipnodes.byname
```

The information about *host-name* displays.

EXAMPLE 2–1 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 2–2 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 2–3 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 **3**

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” on page 63
- “IPv6 Network Interface Configuration File” on page 64
- “Nodes With Multiple Network Interfaces” on page 69
- “IPv6 Daemons” on page 70
- “IPv6 Extensions to Existing Utilities” on page 74
- “Controlling Display Output” on page 76
- “Solaris Tunneling Interfaces for IPv6” on page 76
- “IPv6 Extensions to Solaris Name Services” on page 79
- “NFS and RPC IPv6 Support” on page 82
- “IPv6 Over ATM Support” on page 82

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.

IPv6 includes the module `tun`(7M), which performs the actual packet encapsulation. This module, which is known as the tunneling module, is plumbed. You configure `tun` 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.

For automatic tunneling, IPv6 includes the `6to4tun.c` module. 6to4 also uses the general `tun.c` module during 6to4 tunnel creation.

In IPv6, the `ifconfig`(1M) utility is also modified. You use `ifconfig` to create the IPv6 stack. `ifconfig` 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. You can also add IPv6 interfaces manually, after Solaris installation.

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` or `qe`, or a logical interface, such as a tunnel. See "Solaris Tunneling Interfaces for IPv6" on page 76 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
hostname.ip.tun0
hostname.ip6.tun0
hostname6.ip6to4tun0
hostname6.ip.tun0
hostname6.ip6.tun0
```

IPv6 Interface Configuration File Entry

The autoconfiguration of interfaces in IPv6 enables a node to compute its own link-local address. This address is based on the node's link-layer address, that is, its Ethernet (MAC) 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 other 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 startup 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 37.

Utilities Added or Modified for IPv6

The following utilities have been modified or added to the IPv6 implementation:

- `ifconfig` – This command now includes several modifications for IPv6 support.
- `6to4relay` – This command sets up a tunnel between a 6to4 router and a relay router.

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 `ifconfig`. See "How to Display Interface Address Assignments" on page 44 for a description of useful diagnostic procedures that use this utility.

TABLE 3–1 New `ifconfig` Utility Options

Option	Description
`index`	Sets the interface index
`tsrc/tdst`	Sets tunnel source or destination
`addif`	Creates the next available logical interface
`removeif`	Deletes a logical interface with a specific IP address
`destination`	Sets the point-to-point destination address for an interface
`set`	Sets an address, netmask, or both for an interface
`subnet`	Sets the subnet address of an interface
`xmit/-xmit`	Enables or disables packet transmission on an interface

"Enabling IPv6 Nodes" on page 35 provides IPv6 configuration procedures.

Examples—New `ifconfig` Command 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 3–1 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 3–2 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
```

EXAMPLE 3–3 Configuring 6to4 Through `ifconfig`—Long Form

```
# ifconfig ip.6to4tun0 inet6 plumb
# ifconfig ip.6to4tun0 inet tsrc 129.146.86.187 \
2002:8192:56bb:1::8192:56bb/64 up

# ifconfig ip.6to4tun0 inet6
ip.6to4tun0: flags=2200041<UP,RUNNING,NONUD,IPv6>mtu 1480 index 11
        inet tunnel src 129.146.86.187
        tunnel hop limit 60
        inet6 2002:8192:56bb:1::8192:56bb/64
```

This example of the 6to4 pseudo-interface configuration uses the subnet ID of 1 and also specifies the host ID, in hexadecimal form.

EXAMPLE 3–4 Configuring 6to4 Through `ifconfig`—Short Form

```
# ifconfig ip.6to4tun0 inet6 plumb
# ifconfig ip.6to4tun0 inet tsrc 129.146.86.187 up

# ifconfig ip.6to4tun0 inet6
ip.6to4tun0: flags=2200041<UP,RUNNING,NONUD,IPv6>mtu 1480 index 11
        inet tunnel src 129.146.86.187
        tunnel hop limit 60
        inet6 2002:8192:56bb::1/64
```

6to4relay Command

6to4 tunneling enables communication between isolated 6to4 sites. However, to transfer packets with a native, non-6to4, IPv6 site, the 6to4 router must establish a tunnel with a 6to4 relay router. The *6to4 relay router* then forwards the 6to4 packets to IPv6 network and ultimately, the native IPv6 site. If your 6to4 site must exchange data with a native IPv6 site, you use the 6to4relay command to enable the appropriate tunnel.

Because the use of relay routers is insecure, tunneling to a relay router is disabled by default in the Solaris operating system. Carefully consider the issues that are involved in creating a tunnel to a 6to4 relay router before deploying this scenario. For detailed information on 6to4 relay routers, refer to "Considerations for Tunnels to a 6to4 Relay Router" on page 95. Should you decide to enable 6to4 relay router support, you can find the related procedures in "How to Configure a 6to4 Tunnel to a 6to4 Relay Router" on page 57.

Syntax of 6to4relay

The 6to4relay command has the following format:

```
6to4relay -e [-a IPv4-address] -d -h
```

-e Enables support for tunnels between the 6to4 router and an anycast 6to4 relay router. The tunnel endpoint address is then set to 192.88.99.1, the default address for the anycast group of 6to4 relay routers.

-a *IPv4–address* Enables support for tunnels between the 6to4 router and a 6to4 relay router with the specified *IPv4–address*.

-d Disables support for tunneling to the 6to4 relay router, the default for the Solaris operating system.

–h Displays usage help for 6to4relay.

For more information, refer to the 6to4relay(1M) man page.

Examples—Displaying the Status of 6to4 Relay Router Support

6to4relay, without arguments, shows the current status of 6to4 relay router support. The first example shows the default.

```
# /usr/sbin/6to4relay
6to4relay:6to4 Relay Router communication support is disabled
```

If relay router support is enabled, 6to4relay displays the following output.

```
# /usr/sbin/6to4relay
6to4relay:6to4 Relay Router communication support is enabled
IPv4 destination address of Relay Router=192.88.99.1
```

If you specified the `-a` option and IPv4 address to `6to4relay`, the IPv4 address that you gave with `-a` is displayed instead of 192.88.99.1.

`6to4relay` does not report successful execution of the `-d`, `-e`, and `-a` *IPv4 address* options. However, `6to4relay` does display any errors that might have occurred when you ran these options.

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 "Three Interconnected Networks" in *System Administration Guide: IP Services*. 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. The daemon also implements auto-address configuration for IPv6. The following table shows the supported options.

TABLE 3–2 `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. Advertised packets by the router might be dropped. Also, advertised packets might not reach the host. The `/var/inet/ndpd_state.`*interface* file is a state file. This file is updated periodically by each node. When the node fails and is restarted, the node can configure its interfaces in the absence of routers. This file contains the interface address, the 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 3–3 `/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*]
`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` and `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 hme1 AdvSendAdvertisements 1
prefix  2002:8192:56bb:1::/64 qfe0

if hme1 AdvSendAdvertisements 1
prefix  2002:8192:56bb:2::/64 hme1
```

`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 3–4 `in.ripngd` Daemon Options

Option	Description
`-p` *n*	*n* specifies the alternate port number that is used to send or receive RIPNG packets.
`-q`	Suppresses routing information.

TABLE 3–4 in.ripngd Daemon Options *(Continued)*

-s	Forces routing information even if the daemon is acting as a router.
-P	Suppresses use of poison reverse.
-S	If in.ripngd does not act as a router, the daemon enters only a default route for each router.

inetd 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 3–5 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 tcp6, udp6, tcp, or 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 3–6 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 66 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 45 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 50 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 multi-homed target host by using the -a command-line option. See the ping(1M) man page for details. See "How to Probe All Multi-homed Host Addresses" on page 50 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 multi-homed 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 49 for a description of procedures.

Solaris Tunneling Interfaces for IPv6

Note – The best way to encapsulate IPv6 packets over an IPv4 network is to use 6to4 routing. 6to4 automatically creates a tunnel by using tun. For details about 6to4 routing, see “6to4 as a Transition Mechanism” on page 90. For tasks for configuring 6to4 tunnels, see “How to Configure a 6to4 Router” on page 38.

Tunneling interfaces have the following format:

```
ip.tun ppa
```

ppa is the physical point of attachment.

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 `hostname6.*` 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 3–7 `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. These addresses are the source and destination for the `ip.tun0` interface. Two interfaces are configured. The `ip.tun0` interface is configured. A logical interface, `ip.tun0:1`, is also configured. The logical interface 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 – When 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
```

To create a tunnel to encapsulate IPv6 packets over an IPv6 network, IPv6 over IPv6, you create the following file name:

```
/etc/hostname6.ip6.tun0
```

The following listing is an example of entries in the `hostname6.ip6.tun0` file.

EXAMPLE 3–8 `hostname6.`*interface* Entries

```
tsrc 2000::114:a00:20ff:fe72:668c tdst 2000::103:a00:20ff:fe9b:a1c3
fe80::4 fe80::61 up
```

To create a tunnel to encapsulate IPv4 packets over an IPv6 network, IPv4 over IPv6, you create the following file name:

```
/etc/hostname.ip6.tun0
```

The following listing is an example of entries in the `hostname.ip6.tun0` file.

EXAMPLE 3–9 `hostname.`*interface* Entries

```
tsrc 2000::114:a00:20ff:fe72:668c tdst 2000::103:a00:20ff:fe9b:a1c3
10.0.0.4 10.0.0.61 up
```

To create a tunnel to encapsulate IPv4 packets over an IPv4 network, IPv4 over IPv4, you create the following file name:

```
/etc/hostname.ip.tun0
```

The following listing is an example of entries in the`hostname.ip.tun0` file.

EXAMPLE 3–10 hostname.*interface* Entries

```
tsrc 120.46.86.158 tdst 120.46.86.122
10.0.0.4 10.0.0.61 up
```

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 87. For a description of procedures for configuring tunnels, see “How to Configure IPv6 Over IPv4 Tunnels” on page 53.

IPv6 Extensions to Solaris Name Services

This section describes naming changes that were introduced by the implementation of IPv6. You can store IPv6 addresses in any of the 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 Configuration Procedures” in *System Administration Guide: IP Services* 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.

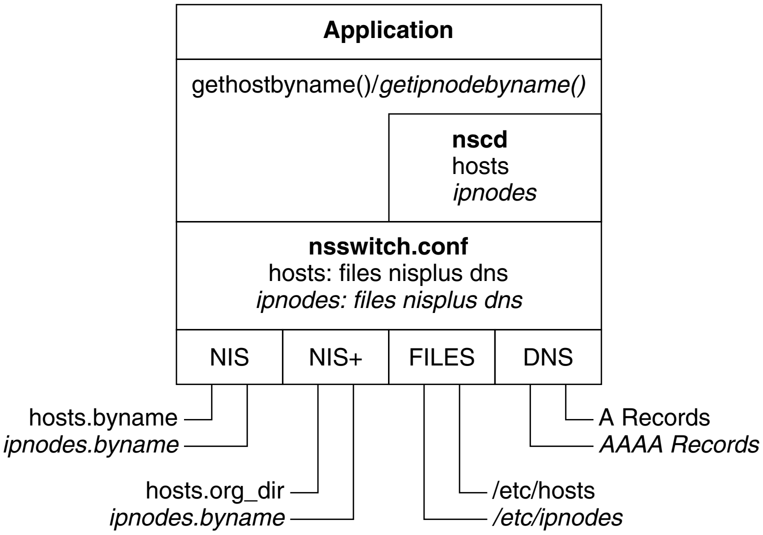

FIGURE 3–1 Relationship Between `nsswitch.conf` and Name Services

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 80, "NIS+ Extensions for IPv6" on page 80, and "DNS Extensions for IPv6" on page 80.

For a description of procedures that use these commands, see "Displaying IPv6 Name Service Information" on page 58.

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, permanent virtual circuits (PVC), and static switched virtual circuits (SVC).

CHAPTER 4

Making the Transition From IPv4 to IPv6 (Reference)

When hosts and routers are upgraded to IPv6, these nodes now have to interoperate with the IPv4 hosts and IPv4 routers. This chapter provides an overview of the standardized solutions to make the transition from IPv4 to IPv6. RFC 1933 also provides detailed solutions to the transition problem.

This chapter contains the following information:

- "Transition Requirements" on page 83
- "Standardized Transition Tools" on page 84
- "IPv4 and IPv6 Interoperability" on page 89
- "Site Transition Scenarios" on page 90
- "6to4 as a Transition Mechanism" on page 90
- "Other Transition Mechanisms" on page 98

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 during the transition process. 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. Compatibility is also required for 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.
- **6to4 router** – Any boundary router that is configured with a 6to4 pseudo-interface on its IPv4 network connection. A 6to4 router serves as an endpoint of a 6to4 tunnel, over which the router forwards packets to another IPv6 site.
- **6to4 host** – Any IPv6 host with an interface that is configured with a 6to4–derived address.
- **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.

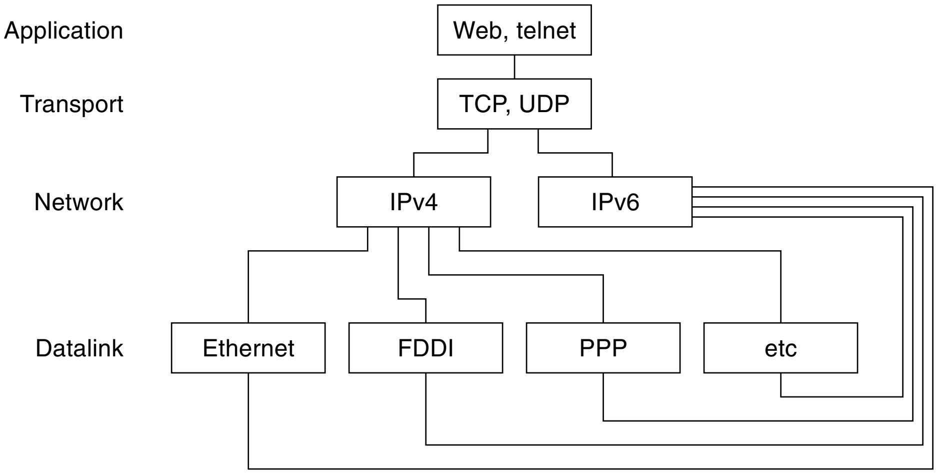

FIGURE 4–1 Dual-Stack Protocols

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, and dual-stacks, enables a dual node to use IPv4 when the dual node communicates with IPv4-only nodes. Also, this name service support enables a dual node to use IPv6 when the dual node communicates 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**

000 ... 000	IPv4 Address

- **IPv4–mapped address**

000 ... 000	0xffff	IPv4 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 IPv4 routers (R).

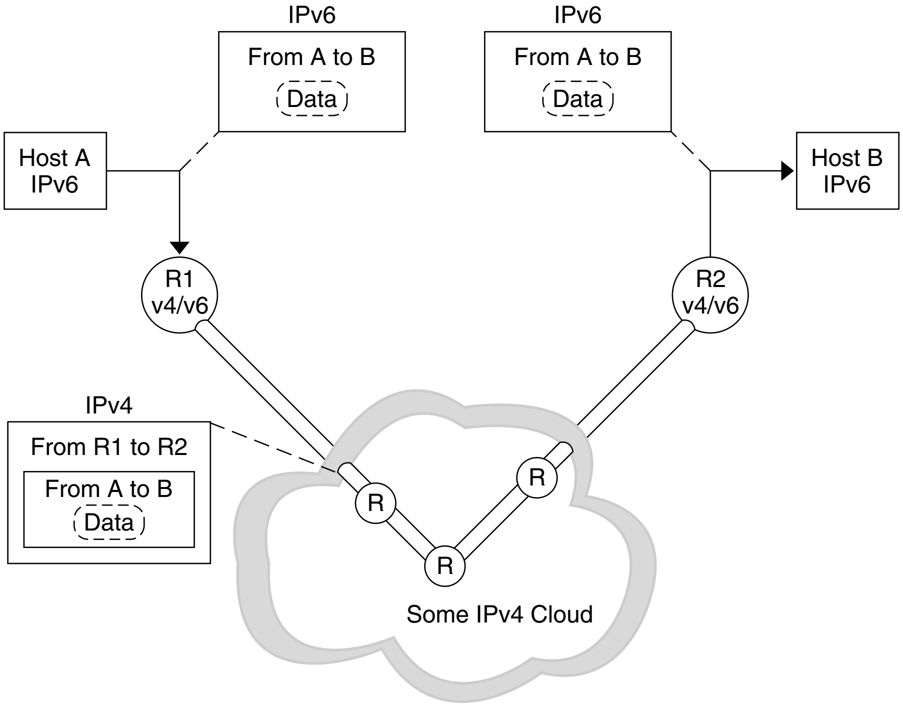

FIGURE 4–2 Tunneling Mechanism

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

Note – The preferred method for creating automatic tunnels is through 6to4 tunneling. For detailed information about the 6to4 routing and tunneling mechanism, refer to "6to4 as a Transition Mechanism" on page 90.

Automatic tunnels require IPv4–compatible addresses. Automatic tunnels 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, which 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. This application 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. A dual-stack 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. You 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 4–1 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 new NIS+ table can be created and administered on any Solaris system. See "IPv6 Extensions to Solaris Name Services" on page 79 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.

6to4 as a Transition Mechanism

The Solaris operating system includes 6to4 as a preferred interim method for making the transition from IPv4 to IPv6 addressing. 6to4 enables isolated IPv6 sites to communicate across an automatic tunnel over an IPv4 network that does not support

IPv6. To use 6to4 tunnels, you must configure a boundary router on your IPv6 network as one endpoint of the 6to4 automatic tunnel. Thereafter, the 6to4 router can participate in a tunnel to another 6to4 site, or, if required, to a native IPv6, non-6to4 site.

This section provides reference materials on the following 6to4 subjects:

- Topology of the 6to4 tunnel
- 6to4 addressing, including the format of the advertisement
- Description of packet flow across a 6to4 tunnel
- Topology of a tunnel between a 6to4 router and 6to4 relay router
- Points to consider before you configure 6to4 relay router support.

More information about 6to4 routing is available from the following sources.

Task or Detail	For Information
Overview of 6to4 routing	"6to4 Tunnels Over IPv4 Networks" on page 33
Tasks for configuring a 6to4 site	"How to Configure a 6to4 Router" on page 38
6to4 related RFC, "Connection of IPv6 Domains via IPv4 Clouds"	RFC 3056, "Connection of IPv6 Domains via IPv4 Clouds"
Detailed information about the `6to4relay` command, which enables support for tunnels to a 6to4 relay router	`6to4relay`(1M) man page
6to4 security issues Internet Draft, "Security Considerations for 6to4"	"Security Considerations for 6to4

Participants in a 6to4 Tunnel

The following figure shows a 6to4 tunnel between two 6to4 sites.

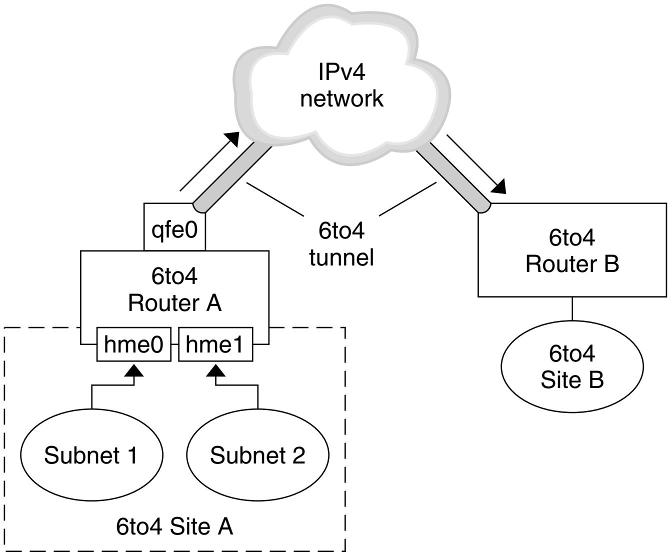

FIGURE 4–3 Tunnel Between Two 6to4 Sites

The figure depicts two isolated 6to4 networks, Site A and Site B. Each site has configured a router with an external connection to an IPv4 network. In the figure, a 6to4 tunnel across the IPv4 network connects the 6to4 sites.

Before an IPv6 site can become a 6to4 site, you must configure at least one router interface for 6to4 support. This interface must provide the external connection to the IPv4 network. The address that you configure on `qfe0` must be globally unique. In the previous figure, boundary Router A's interface `qfe0` connects Site A to the IPv4 network. Interface `qfe0` must already be configured with an IPv4 address before you can configure `qfe0` as a *6to4 pseudo-interface*.

In the figure, 6to4 Site A is composed of two subnets, which are connected to interfaces `hme0` and `hme1` on Router A. All IPv6 hosts on either subnet of Site A automatically reconfigure with 6to4–derived addresses on receipt of the advertisement from Router A.

Site B is the opposite endpoint of the tunnel from Site A. To correctly receive traffic from Site A, a boundary router on Site B must be configured for 6to4 support. Otherwise, packets that the router receives from Site A are not recognized and dropped.

6to4-Derived Addressing

As with native IPv6 routers, you must advertise the subnet prefixes derived from the site 6to4 prefix in `/etc/inet/ndpd.conf`. The next figure shows the parts of a prefix for a 6to4 site, as described in "6to4 Prefix Format" on page 93 and "6to4 Advertisement Example" on page 94.

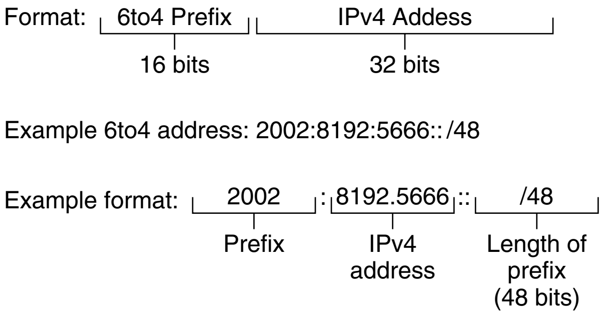

FIGURE 4–4 Parts of a Site Prefix

The next figure shows the parts of a subnet prefix for a 6to4 site, such as you would include in the `ndpd.conf` file.

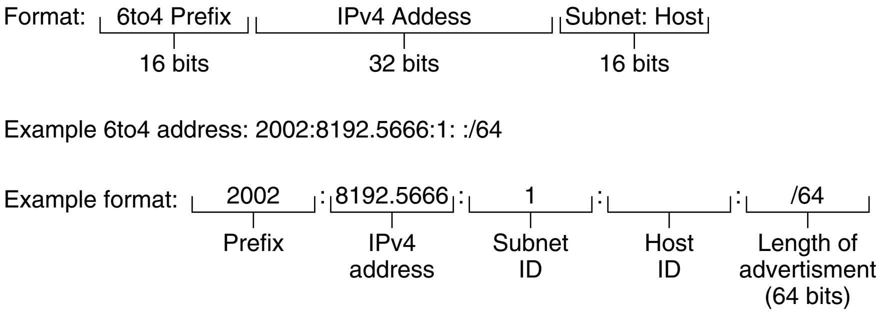

FIGURE 4–5 Parts of a Subnet Prefix

6to4 Prefix Format

The format line in the previous figure contains the following parts.

Part	Length	Definition
Prefix	16 bits	6to4 prefix 2002 (0x2002).

Part	Length	Definition
IPv4 address	32 bits	Unique IPv4 address that is already configured on the 6to4 interface. For the advertisement, you specify the hexadecimal representation of the IPv4 address, rather than the IPv4 dotted–decimal representation.
Subnet ID	16 bits	Subnet ID, which must be a value that is unique for the link at your 6to4 site.

6to4 Advertisement Example

The example in the previous figure has the following values.

Advertisement Part	Corresponding Value
6to4 prefix	2002
IPv4 address	8192:56bb, which corresponds to IPv4 address 129.146.87.188
Subnet ID	1
/64	Length of prefix

6to4-Derived Addressing on a Host

When an IPv6 host receives the 6to4–derived prefix by way of a router advertisement, the host automatically reconfigures a 6to4–derived address on an interface. The address has the following form.

prefix:IPv4 address:subnet ID:host ID`/64`

The results of `ifconfig -a` on a host with a 6to4 interface might resemble the following:

```
qfe1:3: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6>
 mtu 1500 index 7
        inet6 2002:8192:56bb:9258:a00:20ff:fea9:4521/64
```

The 6to4–derived address follows `inet6` in the output from `ifconfig`.

Address Part	Corresponding Value
Prefix	2002, which is the 6to4 prefix

Address Part	Corresponding Value
IPv4 value	8192:56bb, which is the IPv4 address, in hexadecimal notation, for the 6to4 pseudo-interface that is configured on the 6to4 router
subnet ID	9258, which is the address of the subnet of which this host is a member
MAC address	a00:20ff:fea9:4521, which is the link layer address of the host interface that is now configured for 6to4

Packet Flow Through the 6to4 Tunnel

This section describes the path of packets from a host at one 6to4 site to a host in a remote 6to4 site. The next scenario uses the topology that is shown in Figure 4–3 as its example. Moreover, the scenario assumes that the 6to4 routers and 6to4 hosts are already configured.

1. A host on Subnet 1 of 6to4 Site A sends a transmission, with a host at 6to4 Site B as the destination. Each packet header in the flow has a source 6to4–derived address and destination 6to4– derived address.
2. 6to4 Router A receives the outgoing packets and creates a tunnel over an IPv4 network to 6to4 Site B.
3. Site A's router encapsulates each 6to4 packet into an IPv4 header. Then the router uses standard IPv4 routing procedures to forward the packet over the IPv4 network.
4. Any IPv4 routers that the packets encounter use the packets' destination IPv4 address for forwarding. This address is the globally unique IPv4 address of the interface on Router B, which also serves as the 6to4 pseudo-interface.
5. Packets from Site A arrive at Router B, which decapsulates the IPv6 packets from the IPv4 header.
6. Router B then uses the destination address in the IPv6 packet to forward the packets to the recipient host at Site B.

Considerations for Tunnels to a 6to4 Relay Router

6to4 relay routers function as endpoints for tunnels from 6to4 routers that need to communicate with native IPv6, non-6to4 networks. Relay routers are essentially bridges between the 6to4 site and native IPv6 sites. Because this solution is very insecure, by default the Solaris operating system does not enable 6to4 relay router support. However, if your site requires such a tunnel, you use the `6to4relay` command to enable the following tunneling scenario.

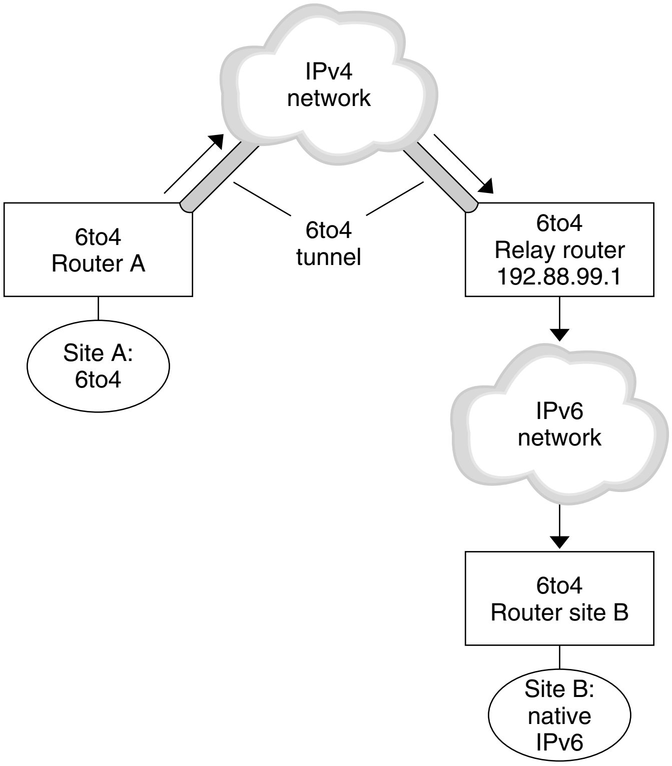

FIGURE 4–6 Tunnel From a 6to4 Site to a 6to4 Relay Router

In Figure 4–6 , 6to4 Site A needs to communicate with a node at native IPv6 Site B. The figure shows the path of traffic from Site A onto a 6to4 tunnel over an IPv4 network. The tunnel has 6to4 Router A and a 6to4 relay router as its endpoints. Beyond the 6to4 relay router is the IPv6 network, to which IPv6 Site B is connected.

Packet Flow Between a 6to4 Site and Native IPv6 Site

This section describes the flow of packets from a 6to4 site to a native IPv6 site. The text uses the scenario that is shown in Figure 4–6 as an example.

1. A host on 6to4 Site A sends a transmission that specifies as the destination a host at native IPv6 Site B. Each packet header in the flow has a 6to4–derived address as its source address. The destination address is a standard IPv6 address.
2. 6to4 Router A receives the outgoing packets and creates a tunnel over an IPv4 network to a 6to4 relay router.

6to4 relay routers that are part of the 6to4 relay router anycast group have the address 192.88.99.1. This anycast address is the default address for 6to4 relay routers. If you need to use a specific 6to4 relay router, you can override the default and specify that router's IPv4 address.

3. Site A's 6to4 router encapsulates each packet into a IPv4 header, which has the IPv4 address of the 6to4 relay router as its destination. The 6to4 router uses standard IPv4 routing procedures to forward the packet over the IPv4 network. Any IPv4 routers that the packets encounter forward the packets to the 6to4 relay router.
4. The physically closest anycast 6to4 relay router to Site A retrieves the packets that are destined for the 192.88.99.1 anycast group.
5. The relay router decapsulates the IPv4 header from the 6to4 packets, revealing the native IPv6 destination address.
6. The relay router then sends the now IPv6–only packets onto the IPv6 network, where the packets are ultimately retrieved by a router at Site B. The router then forwards the packets to the destination IPv6 node.

Security Issues for 6to4 Relay Router Support

By nature, a tunnel between a 6to4 router and 6to4 relay router is insecure. Security problems, such as the following, are inherent in such a tunnel.

- Though 6to4 relay routers do encapsulate and decapsulate packets, these routers do not check the data that is contained within the packets.
- Address spoofing is a major issue on tunnels to a 6to4 relay router. For incoming traffic, the 6to4 router is unable to match the IPv4 address of the relay router with the IPv6 address of the source. Therefore, the address of the IPv6 host can easily be spoofed. The address of the 6to4 relay router can also be spoofed.
- By default, no trust mechanism exists between 6to4 routers and 6to4 relay routers. Thus, a 6to4 router cannot identify whether the 6to4 relay router is to be trusted, or even a legitimate 6to4 relay router. A trust relationship between the 6to4 site and the IPv6 destination must exist, or the both sites leave themselves open to possible attacks.

These problems and other security issues that are inherent with 6to4 relay routers are explained in Internet Draft *Security Considerations for 6to4.* Generally, you should consider enabling support for 6to4 relay routers only for the following reasons:

- Your 6to4 site intends to communicate with a private, trusted IPv6 network. For example, you might enable 6to4 relay router support on a campus network that consists of isolated 6to4 sites and native IPv6 sites.
- Your 6to4 site has a compelling business reason to communicate with certain native IPv6 hosts.
- You have implemented the checks and trust models that are suggested in Internet Draft, *Security Considerations for 6to4*.

Known Issues With 6to4 Router

The following known bugs affect 6to4 configuration:

- 4709338 – Need a RIPng implementation which recognizes static routes
- 4152864 – Configuring two tunnels with the same tsrc/tdst pair works

Implementing Static Routes at the 6to4 Site (BugID 4709338)

The following issue occurs on 6to4 sites with routers that are internal to the 6to4 boundary router. When you configure the 6to4 pseudo-interface, the static route 2002::/16 is automatically added to the routing table on the 6to4 router. Bug 4709338 describes a limitation in the Solaris `RIPng` routing protocol that prevents this static route from being advertised to the 6to4 site.

Either of the following work arounds are available for Bug 4709338.

- Add the 2002::/16 static route to the routing tables of all intra-site routers within the 6to4 site.
- Use a routing protocol other than `RIPng` on the 6to4 site's internal router.

Configuring Tunnels with the Same Source Address (BugID 4152864)

Bug ID 4152864 describes problems that occur when two tunnels are configured with the same tunnel source address, which is a serious issue for 6to4 tunnels.

Caution – Do not configure a 6to4 tunnel and an automatic tunnel with the same tunnel source address.

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. The NAT boxes 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, the addresses must be 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.

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*

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.

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.

Certificate Authority (CA) A trusted third-party organization or company that issues digital certificates used to create digital signatures and public-private key pairs. The CA guarantees the identity of the individual who is granted the unique certificate.

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.

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.

firewall Any device or software that protects an organization's private network or intranet from intrusion by external networks such as the Internet.

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.

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.

IP in IP encapsulation The mechanism for tunneling IP packets within IP packets.

IP link A communication facility or medium over which nodes can communicate at the link layer. The link layer is the layer immediately below IPv4/IPv6. Examples include Ethernets (simple or bridged) or ATM networks. One or more IPv4 subnet numbers or prefixes are assigned to an IP link. A subnet number or prefix cannot be assigned to more than one IP link. In ATM LANE, an IP link is a single emulated LAN. When you use ARP, the scope of the ARP protocol is a single IP link.

IPsec The security architecture (IPsec) that provides protection for IP datagrams.

IPv4 Internet Protocol, version 4. IPv4 is sometimes referred to as IP. This version supports a 32–bit address space.

IPv6 Internet Protocol, version 6. 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.

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 different keys. The public key is known to everyone. The private key is known only to the recipient of the message. IKE provides public keys for IPsec.

redirect In a router, to inform a host of a better first-hop node to reach a particular destination.

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. The keys and algorithms are used in the secure transmission of data.

security associations Associations that specify security properties from one host to a second host.

Security Parameter Index (SPI) An integer that specifies the row in the security associations database (SADB) that a receiver should use to decrypt a received packet.

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. This common key is used to encrypt and decrypt the message. Symmetric keys are used to encrypt the bulk of data transmission in IPsec. DES is one example of a symmetric key system.

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.

unicast address An IP address that identifies a single interface.

Index

Numbers and Symbols

6to4 address
  format, 93
  host address, 94
6to4 advertisement, 39
6to4 prefix
  explanation of parts, 93
  in `/etc/inet/ndpd.conf`
    advertisement, 40
6to4 problems, 98
6to4 pseudo-interface configuration, 38
6to4 relay router
  in a 6to4 tunnel, 68
  security issues, 97
  tunnel configuration tasks, 57, 58
  tunnel topology, 96
6to4 router
  configuration examples, 40
  configuration tasks, 38, 40
6to4 tunnel
  known problems, 98
  overview, 33
  packet flow, 95
  packet flow, with relay router, 96
  sample topology, 91
  to 6to4 relay router, 57
`6to4relay` command
  examples, 68
  syntax, 68
  tunnel configuration tasks, 57

A

AAAA records, 42, 59, 80, 90
address autoconfiguration
  IPv6, 21, 27, 70
address resolution, IPv6, 21
address space, IPv6, 15
addresses
  link-local-use, 17
  local-use, 16
addresses, IPv6
  6to4 format, 93
  aggregate global unicast, 16
  anycast, 14
  IPv4-capable host, 16
  IPv6, 27
  IPX, 16
  link-local-use, 16
  local-use, 17
  multicast, 14, 16
  neutral-interconnect, 16
  NSAP, 16
  site-local-use, 16, 17
  types of addresses, 14
  unicast, 14, 16
    aggregate global, 16
advertisement, 6to4 router, 94
aggregate global unicast addresses, 16
anycast addresses, 57
  IPv6, 19, 24
anycast groups, 6to4 relay router, 57
application layer, gateways, 99
ATM support, IPv6 over, 82
authentication field, IPv6 extension header, 14

authentication header
IPv6, 21, 33
automatic tunnels, transition to IPv6, 88
autonomous address-configuration flag, router advertisement prefix field, 29

B
boundary router, in 6to4 site, 92

D
daemons
in.ndpd, 70
in.ripngd, 72
inetd Internet services, 73
IPv6, 70
DEFAULT_IP variable, 49
destination address field, IPv6 header, 13
destination options field, IPv6 extension header, 14
DNS
AAAA records, 42, 80, 90
adding IPv6 addresses, 42
IPv6 extensions to, 80
PTR records, 60
reverse zone file, 42
zone file, 42
dual-stack
IPv6, 84, 85, 89
duplicate address detection
algorithm, 27
IPv6, 22

E
encapsulating IPv6 packets
6to4 scenario, 34
overview, 64
encapsulation field
IPv6 extension header, 14, 33
/etc/default/inet_type file, 49
DEFAULT_IP value, 75, 76
/etc/hostname6.interface file, 36, 53
IPv6 tunneling, 77
/etc/hostname6.interface file (Continued)
multiple network interfaces, 64, 65
/etc/hostname6.ip.6to4tun0 file, 38
/etc/hosts file, 79
/etc/inet/inetd.conf file, 73
/etc/inet/ipnodes file, 41, 79, 80
/etc/inet/ndpd.conf file, 37, 56, 65, 70
6to4 advertisement, 93
6to4 router advertisement, 39
keywords, 71
/etc/nsswitch.conf file, 80
extension headers, IPv6, 14

F
flow, packets, 31
flow label field
IPv6 header, 13
IPv6 quality-of-service, 30
format prefix, IPv6, 15
fragmentation field, IPv6 extension header, 14

G
getent command, ipnodes option, 61
gethostbyname command, 81
getipnodebyname command, 81
group ID, multicast addresses, 20

H
header fields, IPv6, 13
hop-by-hop option field
IPv6 extension header, 14, 31
hop limit field, IPv6 header, 13
host configuration, 6to4 host address, 94
hostname.interface file
multiple network interfaces, 64, 65
hosts.byaddr map, 41, 80
hosts.byname map, 41, 80
hosts.org_dir table, 41, 80

I

ifconfig command, 44, 64, 76, 77
  6to4 extensions, 39
  -a option, 37
  adding addresses, 65
  examples for IPv6, 67
  IPv6 extensions to, 66
in.ndpd daemon, 64, 65
  options, 70
in.ripngd daemon, IPv6 options, 72
inbound load balancing, 24
inet6 option, route command, 75
inetd daemon, 73
interface address
  6to4, 38
  IPv6, 14
interface ID
  IPv6 link-local-use addresses, 18
  IPv6 site-local-use addresses, 18
Internet Protocol Security
  See IPsec
IP addresses, IPv6, 14
ipnodes.byaddr map, 41
ipnodes.byname map, 41
ipnodes option, getent command, 61
ipnodes.org_dir table, 41, 80
IPsec, 33
  IPv6 authentication header, 33
  IPv6 encapsulating security header, 33
IPv4, interoperability with IPv6, 89
IPv4-capable host address, 16
IPv4-compatible IPv6 address, 18
IPv4-mapped IPv6 address, 19
IPv6
  adding addresses to DNS, 42
  adding addresses to NIS, 41
  adding addresses to NIS+, 41
  address autoconfiguration, 21, 27, 70
  address resolution, 21
  address space, 15
  addresses, 27
  addressing, 14
    prefix format allocations, 15
  anycast addresses, 14, 19, 24
  ATM support, 82
  authentication header, 21, 33
  automatic tunnels, 88
  behavior, 69

IPv6 (Continued)
  comparison with IPv4, 24
  configuring a router, 37
  configuring name services, 85
  configuring routers, 56
  configuring tunnels, 52
  controlling display output, 49
  displaying address assignments, 44
  displaying information through NIS, 60
  displaying information through NIS+, 61
  displaying name service information, 58, 59
  displaying network status, 45
  DNS AAAA records, 59, 90
  DNS extensions, 80
  dual-stack, 84, 89
  duplicate address detection, 22
  enabling nodes, 35
  encapsulating packets, 64
  /etc/hostname6.interface file, 53
  /etc/inet/inetd.conf file, 73
  /etc/inet/ipnodes file, 79, 80
  /etc/inet/ndpd.conf file, 56
  extension header fields, 14
    authentication, 14
    destination options, 14
    encapsulation, 14, 33
    fragmentation, 14
    hop-by-hop option, 14, 31
    routing, 14
  extension headers, 14
  extensions to existing utilities, 74
  extensions to ifconfig command, 66
  features, 11
  getent command, 61
  header
    traffic class field, 13, 32
  header and extensions, 12
  header fields
    destination address, 13
    flow label, 13
    hop limit, 13
    next header, 13
    payload length, 13
    source address, 13
    traffic class, 30, 32
  header format, 12
  header options, 14
  ifconfig command, 44

IPv6 (Continued)
- in.ndpd daemon, 70
- in.ripngd daemon, 72
- interaction with applications, 88
- interoperability with IPv4, 89
- IPv4-capable host address, 16
- link-local addresses, 25, 26, 28, 29
- link-local-use addresses, 16, 17
- local-use addresses, 16, 17
- mobility support
  - home address, 30
- monitoring, 43
- monitoring network traffic, 50
- multicast addresses, 14, 16, 19, 25
- neighbor discovery, 21, 25, 65
- neighbor solicitation, 22
- neighbor solicitation and unreachability, 23
- neighbor unreachability detection, 21, 25
- netstat command, 45, 75
- next-hop determination, 21
- NFS and RPC support, 82
- NIS+ extensions, 80
- NIS+ table, 90
- NIS extensions, 80
- NIS maps, 90
- nslookup command, 59, 60
- parameter discovery, 21
- ping command, 50, 75
- prefix discovery, 21
- probing multihomed host addresses, 50
- protocol overview, 27
- quality-of-service capabilities, 30
  - flow labels, 30
- redirect, 22, 25
- route command, 75
- router advertisement, 22, 23, 25, 28, 29
- router discovery, 21, 24, 70
- router solicitation, 22, 28
- routing, 20
- security improvements, 32
- site-local addresses, 26
- site-local-use addresses, 16, 17
- snoop command, 50, 75
- stateful address autoconfiguration, 26, 29
- stateless address autoconfiguration, 26, 29, 90
- traceroute command, 51, 76
- tracing routes, 51

IPv6 (Continued)
- transition
  - IPv4 compatible address, 86
- transition requirements, 83
- transition scenarios, 90
- transition to, 83
- transition tools, 83, 84
- tunneling, 76, 84
- tunneling mechanism, 87
- unicast addresses, 14, 16

IPv6 addresses
- uniqueness, 29
- with embedded IPv4 addresses, 18

IPX addresses, 16

K

known issues with IPv6, 39, 98

L

link-layer address change, 24

link-local addresses
- IPv6, 25, 26, 28, 29, 77

link-local-use addresses, 16, 17
- interface ID, 18

load balancing, inbound, 24

local-use, 16

local-use addresses, 17

M

managed address configuration flag, router advertisement, 28

messages, router advertisement, 23

mobility support
- home address, 30
- IPv6, 29

MTU, 25

multicast addresses, 16
- group ID, 20
- IPv6, 19, 25
- scope value, 20

multiple network interfaces
- /etc/hostname6.interface file, 64, 65

N

name services
  displaying IPv6 information, 58, 59
  IPv6 extensions to, 79
neighbor discovery
  IPv6, 21, 25
neighbor discovery daemon, 65
neighbor solicitation, IPv6, 22
neighbor solicitation and unreachability, 23
neighbor unreachability detection
  IPv6, 21, 25
`netstat` command, 45, 76
  `-a` option, 45
  `-f` option, 45, 75
  `inet` option, 45
  `inet6` option, 45
  IPv6, 75
  `-p` option, 75
network interfaces
  multiple network interfaces
    `/etc/hostname6.interface` file, 64, 65
network mask, 25
neutral-interconnect addresses, 16
next header field, IPv6 header, 13
next-hop, 25
next-hop determination, IPv6, 21
NFS support, IPv6, 82
NIS
  adding IPv6 address, 41
  IPv6 extensions to, 80
NIS+
  adding IPv6 address, 41
  IPv6 extensions to, 80
NIS+ table, IPv6, 90
NIS maps, IPv6, 90
`nisaddent` command, 41
`nisserver` command, 41
`nissetup` command, 41
`nistbladm` command, 41
NSAP addresses, 16
`nslookup` command, 82
  IPv6, 59, 60

O

other stateful configuration flag, router advertisement, 28

P

packet flow, IPv6
  6to4 and native IPv6, 96
  belonging to the same flow, 31
  definition, 31
  through 6to4 tunnel, 95
parameter discovery, IPv6, 21
payload length field, IPv6 header, 13
`ping` command
  `-A` option, 75
  `-a` option, 50, 75
  IPv6, 50, 75
prefix discovery, IPv6, 21
prefix format allocations, IPv6 addresses, 15
prefixes
  router advertisement, 23, 25, 29
proxy advertisements, 24
PTR records, DNS, 60

Q

quality-of-service
  IPv6, 30
  IPv6 flow label field, 30

R

redirect
  IPv6, 22, 25
relay router, 6to4 tunnel configuration, 57, 58
reverse zone file, 42
`route` command
  `inet6` option, 75
  IPv6, 75
router
  configuration, 6to4, 38
  role in 6to4 topology, 91
router advertisement
  IPv6, 22, 23, 25, 28, 29
  prefix, 29

router configuration, IPv6, 37
router discovery
  IPv6, 21, 24, 70
router solicitation
  IPv6, 22, 28
routers, flow of packets, 31
routing, IPv6, 20
routing field, IPv6 extension header, 14
RPC support, IPv6, 82

S

scope value, multicast addresses, 20
security, IPv6
  6to4 relay router issues, 97
  improvements in IPv6, 32
site-local addresses
  IPv6, 26
site-local-use addresses, 16, 17
  interface ID, 18
  subnet ID, 18
`snoop` command
  `ip6` option, 50
  `ip6` protocol keyword, 75
  IPv6, 75
source address field, IPv6 header, 13
stateful address autoconfiguration, 26, 27, 29
stateless address autoconfiguration, 26, 27, 29
  IPv6, 90
subnet ID, IPv6 site-local-use addresses, 18
subnets in IPv6, in 6to4 topology, 92

T

TCP/IP configuration files
  `/etc/hostname6.interface`, 64, 65
TCP/IP networks
  configuration files
    `/etc/hostname6.interface`, 64, 65
`traceroute` command, 76
  `-a` option, 51, 76
  IPv6, 76
tracing routes, IPv6, 51
traffic class field
  IPv6 header, 13, 32
traffic field, IPv6 header, 30
transition scenarios, IPv6, 90
transition scenarios, IPv6, 6to4 mechanism, 33
transition to IPv6, 6to4 mechanism, 90
`tun` module, 64, 77
tunnel
  6to4 tunnel packet flow, 95
  6to4 tunnel topology, 91
  topology, to 6to4 relay router, 96
tunneling, 84
  configuring routers, 56
  IPv6, 76, 87
tunnels, configuring IPv6, 52

U

unicast addresses, 16
  aggregate global, 16
  format prefix, 17

V

`/var/inet/ndpd_state.interface`
  file, 70

Z

zone file, 42