Installing and Administering HP-UX 11i IPv6 Software

Servers and Workstations

Edition 1

Manufacturing Part Number: T1306-90001
E0901

United States

Legal Notices

The information in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty. A copy of the specific warranty terms applicable to your Hewlett- Packard product and replacement parts can be obtained from your local Sales and Service Office.

Restricted Rights Legend. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

HEWLETT-PACKARD COMPANY 3000 Hanover Street Palo Alto, California 94304 U.S.A.

Use of this manual and flexible disk(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

Contents

Contents

Contents

Contents

Printing History

The manual printing date and part number indicate its current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. the manual part number will change when extensive changes are made.

Manual updates may be issued between editions to correct errors or document product changes. To ensure that you receive the updated or new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

First Edition: September 2001 (HP-UX Release 11i)

Preface

This manual provides information for installing and administering HP-UX 11i IPv6 Software. HP-UX 11i IPv6 Software uses the next generation Internet Protocol (IPv6) to connect HP-UX Servers and Workstations with other systems running IPv4 or IPv6 over IEEE 802.3 or Ethernet Local Area Networks. A IPv6 for HP-UX network can extend over routers into a Wide Area Network.

The information in this manual is intended for network managers or operators who install and administer HP-UX 11i IPv6 software on TCP/IP networks. It is assumed the reader is experienced with HP-UX and is familiar with the basics of local and wide area networking.

Here is the manual organization:

Chapter 1	**Installing HP-UX 11i IPv6 Software** provides brief instructions for installing and minimally configuring HP-UX 11i IPv6 Software.
Chapter 2	**Configuring HP-UX 11i IPv6** describes how to automatically or manually configure HP-UX 11i IPv6.
Chapter 3	**Troubleshooting HP-UX 11i IPv6** provides flowcharts to help diagnose HP-UX 11i IPv6 software problems.
Chapter 4	**HP-UX 11i IPv6 Utilities** describes changes to useful tools for installing, configuring, and maintaining IPv6 for HP-UX software.
Chapter 5	**IPv6 Addressing and Concepts** defines networking terms and explains network interface names, network addresses, names and prefixes.
Chapter 6	**IPv6 Software and Interface Technology** defines terms used by the I/O system to identify HP-UX 11i IPv6.

Related Information

This section lists pertinent networking and system documentation and lists the protocols and standards foundations for HP-UX 11i IPv6 and Internet Services products.

Table 1 List of Networking Manuals

For Information on:	Read:
Troubleshooting IPv6	*Installing and Administering IPv6 for HP-UX Software* *man pages*
Using Sockets Interface	*HP-UX 11i IPv6 Porting Guide*
Using Internet Services for IPv6 Troubleshooting Internet Services for IPv6	*Release Notes for Internet Services* *BINDv9.1.3* *WU-FTP 2.6.1* *Sendmail 8.11.1* on www.docs.hp.com

Table 2 List of Related HP-UX System and Programmer's Manuals

For Information on:	Read:
System Administration	*Installing HP-UX* *System Administration Tasks* *Managing HP-UX Software*
HP-UX Operating System	*Using HP-UX*

Table 3 List of Networking Protocols and Standards

For Information on:	Read:
Stateless Autoconfiguration	RFC 2462

Table 3 List of Networking Protocols and Standards

For Information on:	Read:
Domain Name System (DNS) for IPv6	RFC 1886
Internet Control Message Protocol (ICMP) for IP version 6	RFC 2463
Internet Protocol version 6 (IPv6)	RFC 2460, RFC 2373
Neighbor Discovery	RFC 2461
Program Interface for IPv6	RFC 2553, RFC 2292
Transition Mechanisms for IPv6 Hosts and Routers	RFC 2893
Connection of IPv6 Domains via IPv4 Clouds (6to4)	RFC 3056
Path MTU Discovery for IPv6	RFC 1981

1 Installing HP-UX 11i IPv6

This chapter describes the manual procedures to load HP-UX 11i IPv6 onto your system.

Overview

This chapter contains the following sections:

- Step 1: Checking HP-UX 11i IPv6 Installation Prerequisites.
- Step 2: Loading HP-UX 11i IPv6.
- Step 3: Verifying Software Installation
- Step 4 Verifying IPv6 Loopback

If unfamiliar with IPv6 networking concepts, read Chapter 5 , “IPv6 Addressing and Concepts,” and Chapter 6 , “IPv6 Software and Interface Technology,” of this manual before beginning HP-UX 11i IPv6 installation.

Step 1: Checking HP-UX IPv6 Installation Prerequisites

Before loading HP-UX 11i IPv6 onto the system, meet the following hardware and software prerequisites:

- Check that `/usr/bin`, `/usr/sbin`, and `/sbin` are in **PATH** using the command:

  ```
  echo $PATH
  ```

- Upgrade the operating system to HP-UX 11i software. Verify the operating system version by, executing the command:

  ```
  uname -a
  ```

```
HP-UX hpindon B.11.11 U 9000/715 2009650055 unlimited-user license
```

- Before installing HP-UX 11i IPv6, create a network map or update the existing network map.
- Have 90 MB free disk space.
- Obtain an IPv6 address, default IPv6 gateway address, Name Service addressing, and host name alias for the system.
- Have super-user capability.

Step 2: Loading HP-UX 11i IPv6

The IPv6 depot can be downloaded from **`http://www.software.hp.com`**. (Keyword “IPv6”).

Follow these steps to load the HP-UX 11i IPv6 using the HP-UX `swinstall` program.

1. Log in as root.
2. Check that `/usr/bin`, `/usr/sbin`, and `/sbin` are in your PATH.
3. Download the HP-UX 11i IPv6 into the /tmp directory or into the site software depot.
4. Run the `swinstall` program using the command:

   `swinstall`

   This opens the Software Selection Window and Specify Source Window.
5. Change the Source Host Name if necessary, enter the mount point of the drive in the Source Depot Path field, and activate the **OK** button to return to the Software Selection Window. Activate the Help button for more information.

   The Software Selection Window now contains a list of available software to install.
6. Highlight the HP-UX 11i IPv6 software.
7. Choose `Mark for Install` from the “Actions” menu to choose the product to be installed.
8. Choose `Install` from the “Actions” menu to begin product installation and open the Install Analysis Window.
9. After the Status field displays a Ready message, activate the *OK* button in the Install Analysis Window.
10. Activate the *Yes* button at the Confirmation Window to approve software installation.

    View the Install Window to read processing data while the software is being installed, until the Status field indicates Ready and the Note Window opens.

`swinstall` loads the fileset, runs the control scripts for the fileset, and builds the kernel.

11. Activate the *OK* button on the Note Window to reboot the system.

    The user interface disappears and the system reboots.

12. When the system reboots, check the `swinstall` log file in `/var/adm/sw` to make sure that the installation was complete.

For additional information on the HP-UX `swinstall` program, refer to *Installing HP-UX*.

Step 3. Verifying Software Installation

type `swlist -l bundle IPv6NCF11i`

```
swlist -l bundle IPv6NCF11i

IPv6NCF11i    B.11.11.0109.5C IPv6 11i product bundle
```

Step 4. Verifying IPv6 Loopback

To test IPv6 internal loopback capability type `ping -f inet6 ::1 -n 2`

```
# ping -f inet6 ::1 -n 2
PING ::1: 64 byte packets
64 bytes from ::1: icmp_seq=0. time=2. ms
64 bytes from ::1: icmp_seq=1. time=0. ms
----::1 PING Statistics----
2 packets transmitted, 3 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/0/2
```

Refer to later chapters for more detailed configuration information.

Step 4. Verifying IPv6 Loopback

2 Configuring IPv6 Addresses, Interfaces, and Names

This chapter summarizes the steps to configure LAN interfaces, assign IPv6 addresses, optionally enabling IPv6 tunneling through IPv4

networks, and assigning host names to IPv6 addresses.

The first interface configured on a physical LAN interface is called the **primary interface**. Additional interfaces configured on the same physical device are called **secondary interfaces**. You must configure an IPv6 primary interface to use IPv6 over that interface.

Configuring IPv6 Interfaces and Addresses

Configure IPv6 interface and routing addresses in one of two ways

- Stateless autoconfiguration, which requires an IPv6 router advertising routing information on the LAN.
- Manual configuration

The steps for each configuration method are described below. To preserve configurations across reboots, edit the `/etc/rc.config.d/netconf-ipv6` file.

Before configuring HP-UX 11i IPv6, remember:

- The `netconf-ipv6` file and the script that is executed are shell programs; therefore, shell programming rules apply
- Using `setparms` to configure IPv6 is currently not supported
- You must be superuser to edit the `netconf-ipv6` file and to activate the configuration
- Reboot the system to activate the configuration in the `netconf-ipv6` file

Stateless Autoconfiguration

With stateless autoconfiguration, IPv6 derives a primary address for an interface based on

- the interface's 48-bit MAC address, and
- the well-known prefix fe80::/10

Refer to "Address Autoconfiguration" in the "IPv6 Addressing and Concepts" chapter later in this manual or see the ifconfig(1m) man page for details.

If an IPv6 router on the network advertises network prefixes in router advertisements, IPv6 derives secondary IPv6 addresses based on the network interface identifier of the primary interface and on the network prefixes advertised. IPv6 assigns this address to a secondary interface for the network interface.

Step 1. Configure a Primary Interface (required)

To configure a primary interface, edit an or edit an `IPV6_INTERFACE[0]` statement in the `netconf-ipv6` file to specify the interface name, such as lan0. The interface name must be the name of the physical interface card, as reported by `lanscan`. Refer to "Address Autoconfiguration" in the "IPv6 Addressing and Concepts" chapter later in this manual or see the ifconfig(1m) man page for details.

Here is a sample `netconf-ipv6` file entry

```
IPV6_INTERFACE[0]="lan0"
IPV6_INTERFACE_STATE[0]="up"
```

Step 2. Configure Secondary Interfaces

If an IPv6 Router that advertises network prefixes resides on the LAN, a secondary interface is automatically configured after the primary interface comes up. IPv6 builds additional secondary interfaces for each network prefix advertised.

If you manually configure a link-local address for the primary interface, then autoconfigured secondary addresses are derived from the interface identifier part of the manually-configured address for the primary interface.

For example, if an IPv6 router on the LAN advertises two prefixes (such as fec0::/64 and 2000::/64), HP-UX 11i IPv6 configures two secondary interfaces.

Step 3. Configure Route Information

HP-UX 11i IPv6 automatically configures networks according to the prefixes it receives from an IPv6 router. HP-UX 11i IPv6 automatically adds the router to its list of default gateways if the router advertises a non-zero router-lifetime value.

Manual Configuration

Step 1. Configure a Primary Interface (Required)

To configure an IPv6 link-local address for a primary interface, edit an `IPV6_INTERFACE[0]` statement in the `/etc/rc.config.d/netconf-ipv6` file to specify the interface name, such as lan0 and the interface state, either `up` or `down`. The interface name must be the name of the physical interface card, as reported by `lanscan`. Refer to "Address Autoconfiguration" in the "IPv6 Addressing and Concepts" chapter later in this manual or see the ifconfig(1m) man page for details

Here is a sample `netconf-ipv6` file entry:

```
IPV6_INTERFACE[0]="lan0"

IPV6_INTERFACE_STATE[0]="up"
```

If you don't specify a link-local address, IPv6 derives a link-local address for an primary interface based on the interface's 48-bit MAC address. Refer to "Address Autoconfiguration" in the "IPv6 Addressing and Concepts" chapter later in this manual or see the ifconfig(1m) man page for details

You can also specify a link-local address for the primary interface. The universal/local "U" bit must be set to 0 for manually-configured primary - interface address. That is, the manually configured address for the primary interface must match the pattern FE80::xMxx:xxxx:xxxx:xxxx where x are hexadecimal digits, and M is 0, 1, 4,5,8, 9, C, or D. So M = yy0y, where y is 0 or 1 and the next-to-last bit in the first octet of the interface ID is 0. A sample `netconf-ipv6` file entry is:

```
IPV6_INTERFACE[0]="lan0"
IPV6_INTERFACE_STATE[0]="up"
```

```
IPV6_LINK_LOCAL_ADDRESS[0]= “fe80::1”
```

See the ifconfig(1m) man page for details.

Step 2 Configure Secondary Interfaces

If no IPv6 Router on the LAN advertises network prefixes, you can add secondary interface entries to the `/etc/rc.config.d/netconf-ipv6` file. Editing the `netconf-ipv6` file allows you to identify the network interface name, IPv6 address, and prefix and add entries to the network routing table.

The steps to add this information to the `netconf-ipv6` file are listed below.

```
IPV6_SECONDARY_INTERFACE_NAME[0]="lan0:8"

IPV6_ADDRESS[0]="2345::5432"

IPV6_PREFIXLEN[0]="64"

IPV6_SECONDARY_INTERFACE_STATE[0]="up"

DHCPV6_ENABLE[0]=0
```

Always set `DHCPV6_ENABLE` to 0.

For more information about specifying interface names for multiple interfaces, see chapter 5, “IPv6 Addressing and Concepts.”

Step 3 Configure a Default IPv6 Route

If a local IPv6 router advertises zero-lifetime prefixes only, you can enable the IPv6 system to communicate with the local IPv6 router by adding the router-addressing and hop-count parameter information. The routing configuration parameters have an index value, [x], that groups the routing parameters together. Here is a sample `netconf-ipv6` entry:

```
IPV6_DESTINATION[0]="default"

IPV6_GATEWAY[0]="2008:7:6:5:4:3:2:1"

IPV6_ROUTE_COUNT[0]="1"

IPV6_ROUTE_ARGS[0]=""
```

Step 4 Enable Tunneling (optional)

To enable tunneling set the IPV6_TUNNEL variable to “1”. Refer to the

"Tunneling" subsection of Chapter 6 for details.

Step 5 Create a Configured Tunnel (optional)

If you regularly expect to exchange data between isolated IPv6 networks over an IPv4 network, creating a configured tunnel.

1. Configure the tunnel. Example route command and netconf-ipv6 configurations are

   where `c001::` is a destination network and `192.1.1.1` is a gateway address.

   Example netconf-ipv6 configuration is

   ```
   IPV6_DESTINATION[0]="C001::/64"
   IPV6_GATEWAY[0]="::192.1.1.1"
   IPV6_ROUTE_COUNT[0]="1"
   IPV6_ROUTE_ARGS[0]="-t"
   ```

Step 6 Activate the netconf-ipv6 configuration

You can activate the `netconf-ipv6` configuration in one of two ways:

1. Reboot. HP recommends rebooting your system to activate any changes you made to your `netconf-ipv6` file. A reboot is the cleanest way to reconfigure an interface because the reboot handles any network initialization dependencies.
2. Execute the `ifconfig` and `route` commands at the HP-UX prompt. HP recognizes that system reboots are disruptive to end users. To delay or schedule the reboot, but still make your configuration changes active, you may execute the `ifconfig` and `route` commands with the appropriate values for your network. When you do reboot, the values in your `netconf-ipv6` file will be used. Refer to the example below or see the `ifconfig(1M)`, and `route(1M)` man pages for information on command usage.

Example ifconfig and route Commands

HP recommends editing the `/etc/rc.config.d/netconf-ipv6` file to preserve IPv6 interface and address configurations after the system reboots. For reference, here are the commands equivalent to the netconf-ipv6 edits described earlier. Remember that configuration from the command line is not kept after a system reboot. Refer to the ifconfig(1M)

and route(1M) man pages for details.

To configure a primary interface:

```
 ifconfig lan0 inet6 up
```

To configure a secondary interface:

```
ifconfig lan0:8 inet6 2345::5432 up
```

To add a default IPv6 route:

```
route inet6 add net default 2008:7:6:5:4:3:2:1
```

To enable tunneling

```
ifconfig tu0 inet6 up
```

To create a configured tunnel:

```
route inet6 -t add c001::/64 ::192.1.1.1l
```

Host Names and IPv6 Addresses

Creating the /etc/hosts File (Optional)

The IETF [RFC 2893] recommends adding IPv6 addresses (know as A6/AAAA records) to a DNS Name Server only when the following conditions are true:

- The IPv6 address is assigned to the interface on the node
- The address is configured on the interface
- The interface is on a link which connects to the IPv6 infrastructure

HP recommends beginning with IPv6 addresses and host names in the `/etc/hosts` file on a development network; then adding IPv6 addresses and hosts to a Domain Name Service when moving IPv6 to a production backbone network.

This subsection describes how to edit the `/etc/hosts` file to add an IPv6 address and hostname for the network interface you are configuring.

NOTE If using the name service DNS over IPv6, add the IP address and host name to the appropriate databases on the name server system. Refer to the Release Notes for BIND v9.1.3 on www.docs.hp.com for more information on DNS over IPv6.

The `/etc/hosts` file associates IP host addresses with mnemonic host names and alias names. It contains the names of other nodes in the network with which your system can communicate.

An example `/etc/hosts` file ships with HP-UX 11i.

Example Host Name Entry

The example below shows how a system with the name, `host3`, might be referenced in the `/etc/hosts` file:

System name in `swinstall` screen:

```
host3
```

/etc/hosts file:

```
8:7:6:5:4:3:2:1 host3 host3.site2.region4
1.2.3.4 host3 host3.site2.region4
```

NOTE HP-UX 11i IPv6 is a dual stack implementation. A single host name can have entries for both an IPv6 address and an IPv4 address in /etc/hosts.

Name and Address lookup for IPv6

/etc/nsswitch.conf (nsswitch.conf(4)) is a configuration file for the name service switch. A new entity, ipnodes, specifies which name services resolve addresses and host names for the IPv6 Dual Stack.

For HP-UX 11i IPv6, the new keyword "ipnodes" specifies the resolver policy for the library functions getnameinfo(3N) and getaddrinfo(3N) for both IPv4 and IPv6 addresses. The existing keyword "hosts" specifies the resolver policy for the library functions gethostbyname() and gethostbyaddr() for IPv4 addresses.

By default, the /etc/nsswitch.conf is not on a system. The default ipnodes policy is

```
dns [NOTFOUND=return] files
```

The policy "dns [NOTFOUND=return] files" implies that dns is the authoritative resolver. If dns is available but returns NOTFOUND, the search stops. The search continues with files only if dns is down.

WARNING **If the "ipnodes" entry is not added to the /etc/nsswitch.conf file after HP-UX 11i IPv6 product is installed, then address resolution for IPv6 commands or services will always fail.**

This policy matches the default "hosts" resolver policy. However, IPv6 Dual Stack commands or applications cannot resolve host names and IPv6 addresses using /etc/hosts with this policy. So Hewlett-Packard recommends the policy

```
ipnodes: dns [NOTFOUND=continue] files
```

After HP-UX 11i IPv6 is installed, `telnet` and the `r*` commands will use `ipnodes` search criteria for all host name and address lookups.

Manually editing nsswitch.conf

The configuration file for the name service switch is /etc/nsswitch.conf. If the current system has no `nsswitch.conf` file, use a text editor to create an /etc/nsswitch.conf file containing the following line.

```
ipnodes: dns [NOTFOUND=continue] files {NOTFOUND=return}
```

Or if /etc/hosts is the primary Name Service

```
ipnodes: files [NOTFOUND=continue] dns {NOTFOUND=return}
```

Refer to the man page `nsswitch.conf(4)` for hosts syntax.

3 Troubleshooting HP-UX 11i IPv6

This chapter provides guidelines for troubleshooting HP-UX 11i IPv6. It contains a troubleshooting overview and diagnostic flowcharts:

Troubleshooting Overview

Troubleshooting HP-UX 11i IPv6 problems can involve a variety of hardware and software components. The problem impacting your system might originate in another part of the network.

Because HP-UX 11i IPv6 runs over an IPv6/IPv4 Dual Stack, test IPv4 connectivity before testing IPv6 connectivity. Refer to the *Installing LAN/9000 Software* manual on www.docs.hp.com for IPv4 troubleshooting advice.

Then if you are still unable to identify your problem, proceed to the troubleshooting flowcharts. The troubleshooting flowcharts provide logical steps to follow when troubleshooting HP-UX 11i IPv6. Use the diagnostic flowcharts provided in this chapter to identify whether the problem is with HP-UX 11i IPv6, or router configuration. Verify your assumptions.

Diagnostic Flowcharts

Below is a summary of the types of network tests in the diagnostic flowcharts. To diagnose your problem, first check the connections and configuration on your system (Flowcharts 1 through 5). If this does not solve your problem, use flowcharts 6 through 8 to test/verify connectivity with a remote system.

1	Transport Level Test using Internet Services
2	Network Connectivity Test
3	Name Services Test
4	Interface Test
5	Interface Test (continued)
6	Router Remote Loopback Test

Transport Level Loopback Test using Internet Service: Checks roundtrip communication between Transport Layers on the source and target host using `telnet`.

Network Connectivity Test: Checks roundtrip communication between Network Layers on the source and target host using the `ping(1M)` diagnostic.

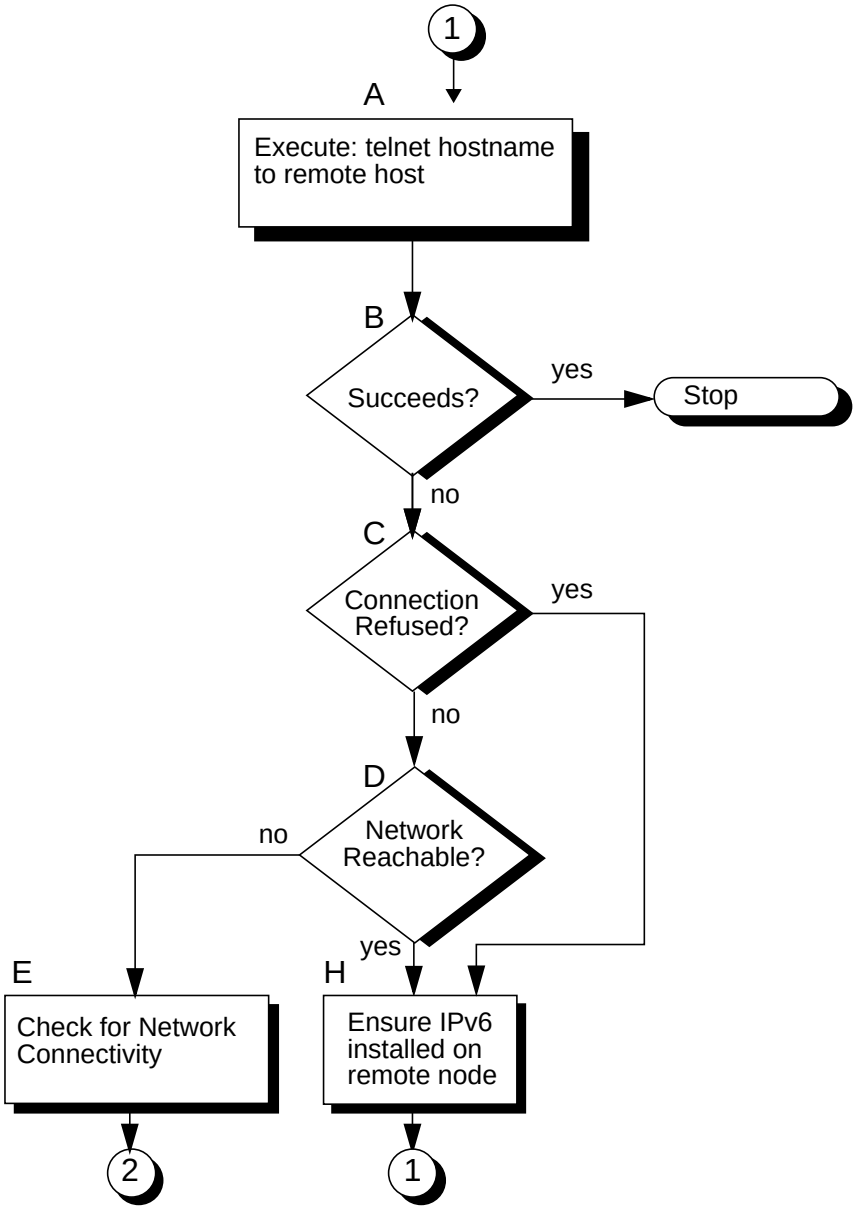
Name Services Test: Checks host name and IPv6 address resolution.

Interface Test: Verifies the configuration of the network interface on a host using the `lanscan(1M)`, and `ifconfig(1M)` commands.

Router Remote Loopback Test: Verifies the connection between local and remote nodes through IPv6 routers using the `ping(1M)` and `netstat(1)` commands.

Flowchart 1: Transport Level Testing using Internet Services

Figure 3-1 **Flowchart 1**



Flowchart 1 Procedures

A. *Execute: telnet <hostname> to remote host.* Try to connect using `telnet` to a remote host.

B. *Succeeds?* If `telnet` succeeds, stop. The system connects using TCP6 over IPv6 through the Transport Layer (OSI Layer 4).

C. *Connection Refused?* Trying to connect to a remote system where HP-UX 11i IPv6 is not installed can cause this message.

D. *Network Reachable?* Trying to connect to a remote system where HP-UX 11i IPv6 is not installed can cause this message.

E. *Check for Network Connectivity* Ensure network connectivity by following the steps in Flowchart 2.

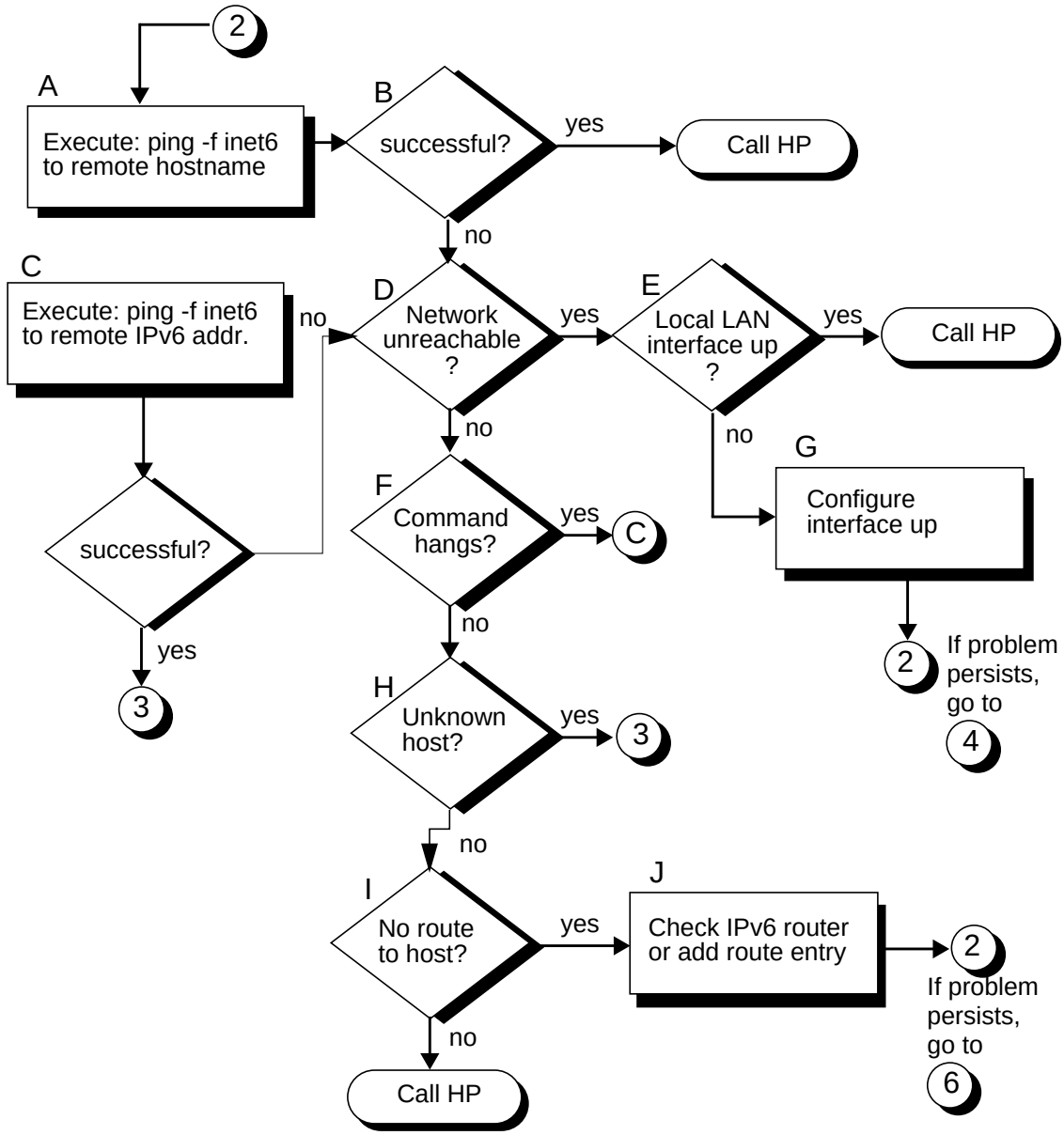
F. *Ensure IPv6 installed on remote node* Run

```
swlist -l bundle IPv6NCF11i
```

on the remote node to ensure IPv6 is installed. If `telnet` still fails, check the `inetd.conf` file.

I. Flowchart 2: Network Connectivity Test

Figure 3-2 Flowchart 2



Flowchart 2 Procedures

A. *Execute: ping -f inet6 to remote hostname.* Using `ping(1M)`, send an ICMPv6 message to the remote host with which you are having problems connecting. For example, for remote host name is `hpindon`. Enter:

```
ping -f inet6 hpindon
```

B. *ping successful?* A message is printed on `stdout` for each `ping` packet returned by the remote host. If packets are being returned, your system has network level connectivity to the remote host.

C. *Execute: ping to remote IPv6 address.* Using `ping(1M)`, send a message to the IPv6 address of the remote host. For example,:

```
ping -f inet6 8:7:6:5:4:3:2:1
```

D. *Network unreachable?* If so, check the status of the local LAN interface first.

E. *Local LAN interface up?* Execute `ifconfig` on the local interface to be sure it is configured up.

F. *Command hangs?* If a message is not returned after executing `ping`, go to Flowchart 4.

G. *Configure interface up.* If you find the local interface is not up, execute `ifconfig` with the appropriate flags set. Proceed to Flowchart 2. If the problem persists, go to Flowchart 4.

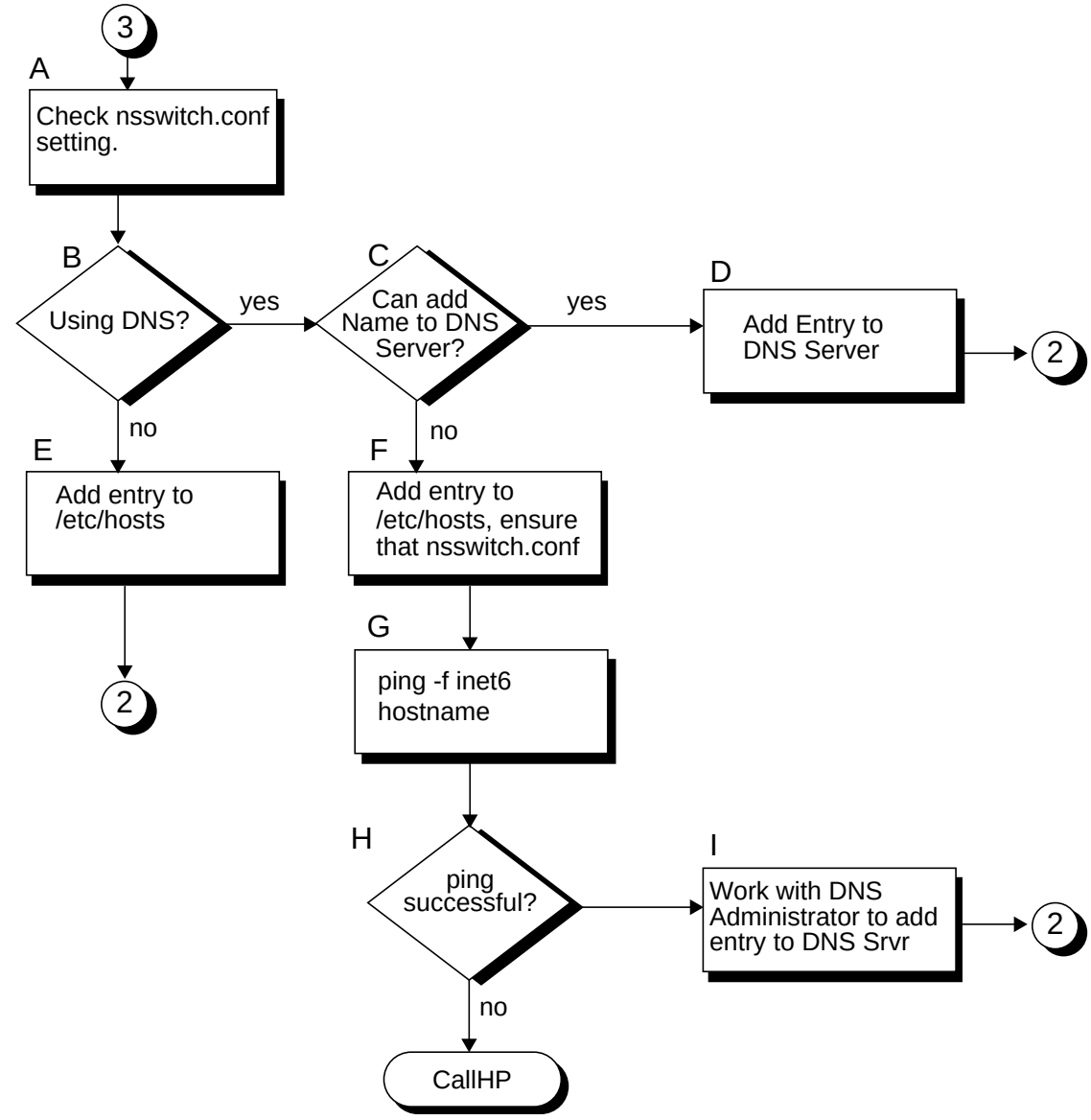
H. *Unknown host?* (Error= Unknown host *hostname*?) There is a problem with the IPv6 address configuration for the host *hostname* in the `/etc/hosts` file or on the name server. Go to Flowchart 3

I. *No route to host?* (Error= Sendto: No route to host?) Use `netstat -rn` to check the routing table. If there is no route to host, go to J. Otherwise, call your HP representative for help.

J. *Check IPv6 Router or add route table entry.* Add a route table entry to that host, or ensure that the IPv6 router advertises correct prefixes. Then try Flowchart 2 again. If the problem persists, go to Flowchart 6.

Flowchart 3: Name Service Test

Figure 3-3 Flowchart 3



Flowchart 3 Procedures

A. *Correct /etc/hosts configuration, check /etc/nsswitch.conf file.* Add the missing host name or IPv6 address. If the IPv6 address for the host is in /etc/host, ensure that you have an /etc/nsswitch.conf file entry of the form:
ipnodes: DNS [NOTFOUND=continue] files
and start again with Flowchart 3.

B. *Using DNS?* If your name and IPv6 address resolution policy uses DNS as the primary resolver, go to C.

C. *Can you add a Host Name to the DNS Server.?* Are you a DNS administrator? If so, proceed to D.

D. *Add Entry to DNS Server.* Refer to the BINDv9.1.3 Release Notes for details. Then retry Flowchart 2.

E. *Add entry to /etc/hosts* If your name and IPv6 address resolution policy uses /etc/hosts as the primary resolver, add a correct IPv6 address and host name to the local /etc/hosts file. Then retry Flowchart 2.

F. *Add entry to /etc/hosts. Ensure that nsswitch.conf is configured properly.* Add a correct IPv6 address and host name to the local /etc/hosts file. Ensure that your host name and IPv6 address resolution policy, specified in the file `/etc/nsswitch.conf`, uses `/etc/hosts`. Then retry Flowchart 2.

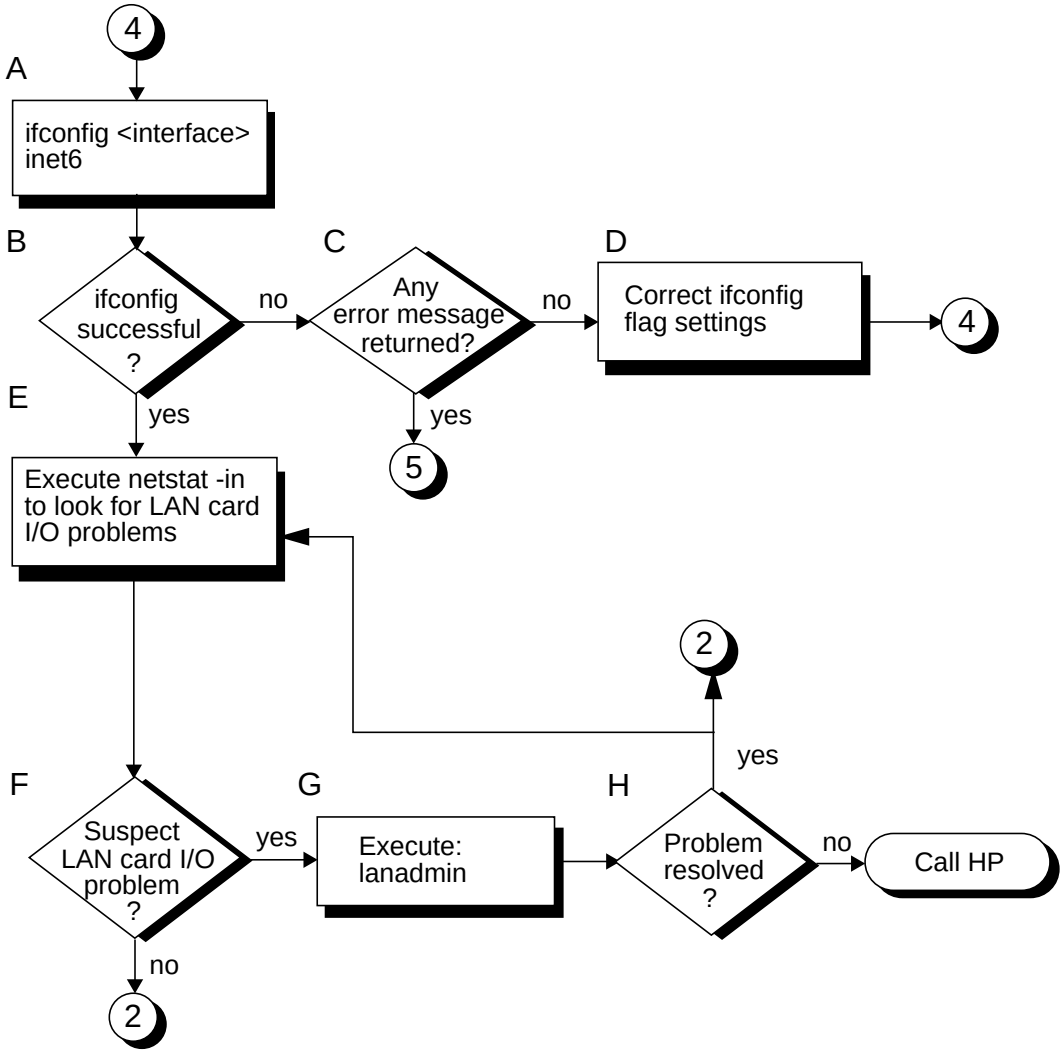
G. *ping -f inet6 hostname.* Test connectivity to the remote host using the `ping(1M)` command.

H. *ping successful?* If `ping -f inet6 hostname` succeeds using a host name and IPv6 address from `/etc/hosts`, DNS needs updating. If ping fails, check the `/etc/hosts`, `/etc/resolv.conf`, and `/etc/nsswitch.conf` files on both the local and remote hosts. If all look correct, call an HP representative for help.

I. *Work with DNS Administrator to add entry to DNS Server* Then retry Flowchart 2 to ensure that DNS correctly resolves host names and IPv6 addresses.

Flowchart 4: Interface Test

Figure 3-4 Flowchart 4



Flowchart 4 Procedures

A. *Execute: ifconfig <interface>inet6.* Execute `ifconfig` on the interface you want to test. For example, to check LAN interface `lan0`, enter:

```
ifconfig lan0 inet6
```

B. *ifconfig successful?* `ifconfig` succeeds when the output shows an Internet address and the flags: UP,RUNNING,MULTICAST,ONLINK.

C. *Any error message returned?* If `ifconfig` fails and displays an error message, go to Flowchart 5. Flowchart 5 shows common error messages and what to do for each.

D. *Correct ifconfig with non-default flag settings.* If `ifconfig` returns an unexpected flag setting, re-execute the command with the proper setting. For more information, refer to the `ifconfig(1M)` man page. Start again with Flowchart 4

E. *Execute: netstat -inf -inet6.* If `ifconfig` succeeds, then the network interface is configured correctly. `netstat -i` displays the number of incoming (`Ipkts`) and outgoing (`Opkts`) packets passed through an interface. No increase in the number of incoming or outgoing packets could indicated LAN card I/0 problems.

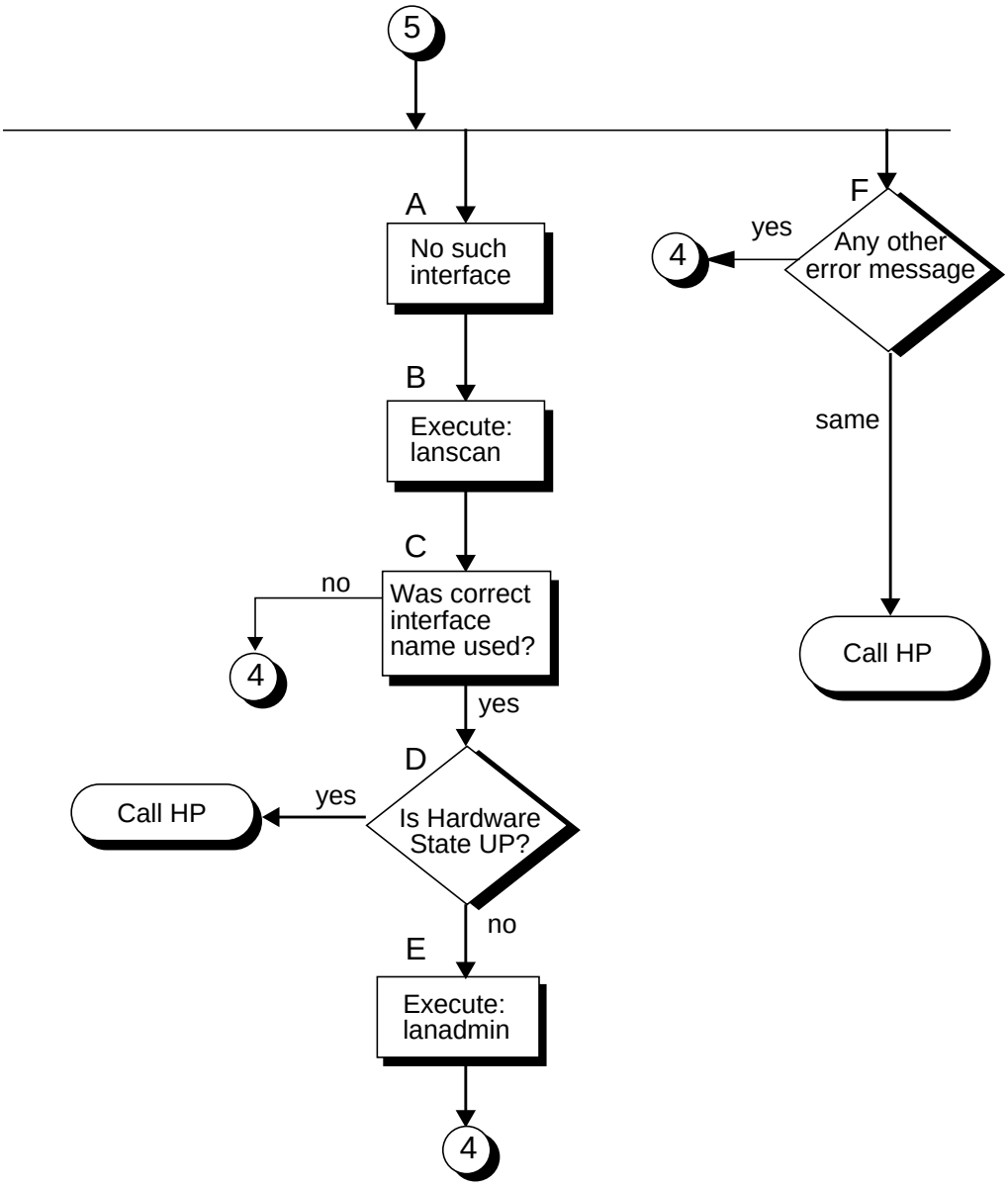
F. *Suspect LAN card I/O problems?* If the statistics indicate possible LAN card problems, go to G, otherwise go to Flowchart 2 to test Network Connectivity.

G. *Execute: lanadmin.* Use `lanadmin` to ensure the LAN card is operational. A substantial increase in the number of the `Ierrs` and `Oerrs` during a file transfer attempt might indicate transmission problems.

H. *Problem resolved?* If you found and corrected the LAN card problem, return to step E to verify the correction. If corrected, reexecute `ifconfig` to bring up the interface, then go to Flowchart 2. If the problem is still unresolved, call HP. Prepare to discuss the problem as described in “Contacting Your HP Representative” in chapter 4, “HP-UX 11i IPv6 Utilities.”

Flowchart 5: Interface Test continued

Figure 3-5 Flowchart 5



Flowchart 5 Procedures

A. *No such interface name*. The interface name passed to `ifconfig` does not exist on the system. Check spelling and names of interfaces on the system using `lanscan`.

If the system contains more than one LAN card, make sure the correct number of LAN cards was configured into the kernel and that an `ifconfig` command was executed for each interface.

B. *Execute lanscan*. Execute `lanscan` to display information about the LAN cards in your system.

C. *Was correct interface name used*? Configure interface using ifconfig with the correct interface name. After reconfiguring using the correct interface name, start again with Flowchart 4.

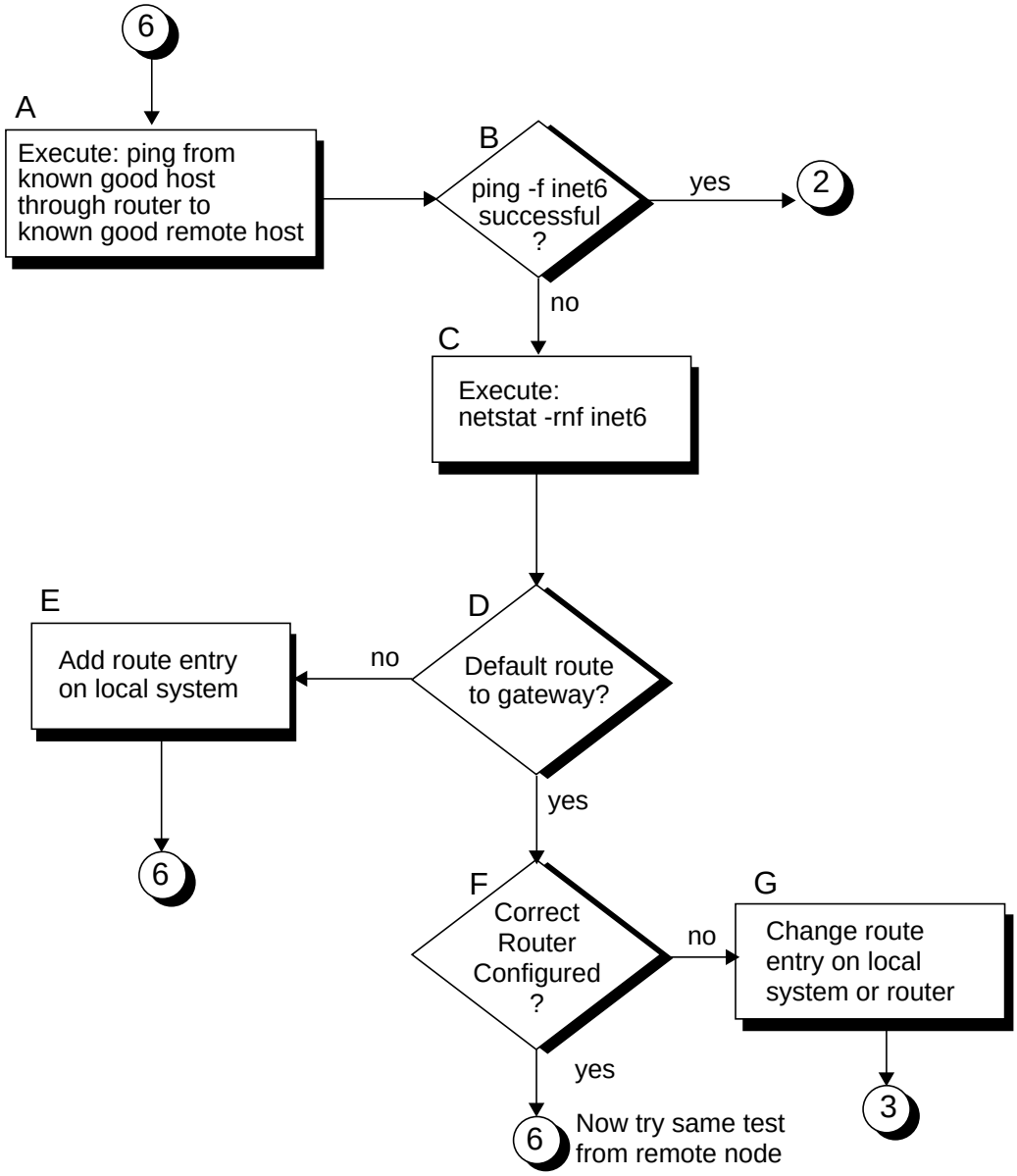
D *Is Hardware State UP?* Verify the state of the hardware with the output from the `lanscan` command. If the Hardware State is up call HP, otherwise go to E.

E. *Execute lanadmin*. Use the `lanadmin` command to reset the LAN card. Go to Flowchart 4.

F. *Any other error message*. Interpret any other error message and take the appropriate action. Then repeat flowchart 4. If you receive the same error message again, call HP. For assistance, call an HP representative. Prepare to discuss the problem as described in “Contacting Your HP Representative” in chapter 4, “HP-UX 11i IPv6 Utilities.”

Flowchart 6: Router Remote Loopback Test

Figure 3-6 Flowchart 6



Flowchart 6 Procedures

A. *Execute: ping from known good host through gateway to known good host on remote network.* This tests router connectivity to the remote network. For more information on `ping(1M)`, refer to the ping(1M) man page.

B. *ping successful?* If `ping -f inet6` succeeded, return to Flowchart 2. If `ping -f inet6` failed, the problem may exist in the routing table for the problem host. Go to C.

C. *Execute netstat -rn.* To display gateway routing information in numerical form, execute:

```
netstat -rnf inet6
```

D. *Direct route to remote or default route to gateway?* If the route exists, go to F. If not, go to E to add a new route.

E. *Add route entry on local system.* Use the `route(1M)` command to add a route entry to the route table on the local system. Refer to `route(1M)` for a complete description of the command. Or if an IPv6 router on the LAN advertises default routes, wait a few minutes to see a route advertisement is added to the default router list.

F. *Correct router configured?* If your local host has a route to the correct router, then retry Flowchart 6 from the remote node. If the remote node's routing is configured properly, and both the local and remote nodes can connect to their respective routers, then contact your ISP or network administrator to verify network-to-network connectivity.

G. *Change route entry on local system or router.* If the routing information is incorrect, correct it using the `route(1M)` command, or verify that the IPv6 router is advertising proper subnet prefixes. Then retry Flowchart 2 to test network connectivity.

4 HP-UX 11i IPv6 Utilities

HP-UX 11i IPv6 enhances standard IPv4 network utilities. This section summarizes the improvements for HP-UX 11i IPv6.

Configuration Utilities

ifconfig inet6 address family

Use `ifconfig(1M)` to assign an IPv6 address to an interface and configures parameters, such as the network prefix which replaces netmask.

The new `ifconfig` **inet6** address family is required to configure IPv6 interfaces. It is not required to examine IPv6 interfaces. Refer to the ifconfig(1M) man page for details.

Neighbor Discovery Protocol replaces arp in IPv6

The Neighbor Discovery Protocol (`ndp`) replaces `arp` in IPv6. Refer to "Neighbor Discovery" in Chapter 5 for details.

route inet6

`route(1M)` adds and deletes entries to the network routing table, allowing your system to communicate through a router. An IPv6 router that advertises routes and network prefixes can perform all the functions of an IPv6 route command, except configure a tunnel. Refer to the route(1M) man page for details.

Network Diagnostic Utilities

- `lanadmin(1M)` resets or reports status of the LAN card.
- `lanscan(1M)` displays LAN device configuration and status.
- `ndd(1M)` displays and modifies network driver parameters.
- `ndp(1M)` displays and modifies the IPv6 neighbor discovery cache
- `netstat(1)` provides network statistics and information about network connections.
- `ping(1M)` verifies network connectivity through the Network Layer

and reports round-trip time of communication time between hosts.

- `traceroute`(1M) traces the path between hosts at the Network Layer.

IPv6 Additions to Network Tracing and Logging

Use `nettl` to trace traffic through new IPv6 Subsystems, or use nettladm.

Table 4-1 New Network Trace Subsystems

Description	Subsystem Name
IPv6 Packets	NS_LS_IPV6
ICMPV6 Packets	NS_LS_ICMPV6
IPv6 Loopback packets	NS_LS_LOOPBACK6

Use `netfmt` to format trace records captured by `nettl` from the new IPv6 subsystems. `netfmt` can also filter `nettl` output according to the following IPv6 criteria:

Table 4-2 New IPv6 Network Filter Criteria

Filter Description	Entry in netfmt config. file
IPv6 Packets	NS_LS_IPV6
ICMPV6 Packets	NS_LS_ICMPV6
IPv6 Source Address	`filter ip6_saddr ::abcd` where ::abcd is the source address
IPv6 Destination Address	`filter ip6_daddr ::fedc` where ::fedc is the source address
Connection per port and IPv6 address	filter connection6 <local_IPv6addr> | <port> <remote_IPv6addr | port

Contacting Your HP Representative

If you do not have a service contract with HP, you may follow the procedure described below, but you will be billed accordingly for time and materials.

If you have a service contract with HP, document the problem as a Service Request (SR) and forward it to your HP representative. Include the following information where applicable:

- A characterization of the problem. Describe the events and symptoms leading up to the problem. Attempt to describe the source of the problem.

  Your characterization should include: HP-UX commands; communication subsystem commands; functionality of user programs; result codes and messages; and data that can reproduce the problem.

- Obtain the version, update, and fix information for all software. To check your Internet Services or HP-UX 11i IPv6 Software version, execute the command:

  ```
  what /stand/vmunix >> /tmp/filename
  ```

  To check the version of your kernel, execute the command:

  ```
  uname -a >> /tmp/filename
  ```

  This allows HP to determine if the problem is already known, and if the correct software is installed at your site.

- Illustrate as clearly as possible the context of any message(s). Record all error messages and numbers that appear at the user terminal and the system console.

- Prepare a listing of the HP-UX I/O configuration you are using for your HP representative to further analyze.

- Try to determine the general area within the software where you think the problem exists. Refer to the appropriate reference manual and follow the guidelines on gathering information for that product.

- Document your interim, or “workaround,” solution. The cause of the problem can sometimes be found by comparing the circumstances in which it occurs with the circumstances in which it does not occur.

- Create copies of any Internet Services or HP-UX 11i IPv6 Software link trace files that were active when the problem occurred for your HP representative to further analyze.
- *In the event of a system failure, obtain a full memory dump.* If the directory `/var/adm/crash` exists, the HP-UX utility `/usr/sbin/savecore` automatically executes during reboot to save the memory dump. Hewlett-Packard recommends that you create the `/var/adm/crash` directory after successfully installing this product. Send the output of your system failure memory dump to your HP representative.
- Prepare copies of the name service files such as `/etc/hosts`, etc/`nsswitch.conf`, `named.conf` and `resolv.conf`. Prepare a copy of the IPv6 configuration file /`etc/rc.config.d/netconf-ipv6` files.
- For EISA only: Run the `show board` command of the `eisa_config` utility on the LAN card and record the output.
- Verify the software: `/usr/sbin/swverify > /tmp/swv-out`
- Execute the `display` command of the `lanadmin` diagnostic on the LAN interface and record the output.
- Record the troubleshooting flowchart number and step number where you are unable to resolve the problem.
- Save all network log files. Make sure that ERROR and DISASTER log classes are enabled when log files are collected in `/var/adm/nettl.LOG000`.
- Execute the following commands and record the output:

```
uname -a >> /tmp/filename
what /stand/vmunix >> /tmp/filename
lanscan >> /tmp/filename
netstat -sf inet6 >> /tmp/filename
netstat -inf inet6 >> /tmp/filename
netstat -rnf inet6 >> /tmp/filename
ndp -an >> /tmp/filename
ndd -get /dev/tcp6 tcp_status >> /tmp/filename
ndd -get /dev/ip6 ip6_ill_status >> /tmp/filename
ndd -get /dev/ip6 ip6_ipif_status >> /tmp/filename
ndd -get /dev/ip6 ip6_ire_status >> /tmp/filename
ndd -get /dev/ip6 ip6_ill_config_status >> /tmp/filename
```

Prepare the formatted output and a copy of the log file for your HP representative to further analyze.

5 IPv6 Addressing and Concepts

This chapter introduces network addressing concepts for IPv6. It contains the following sections:

- Where to Get IPv6 Addresses

- IPv6 Address Formats
- Autoconfiguration Summary
- Stateless Address Autoconfiguration
- Networking Terminology

Where to get IPv6 Addresses

Contact a local ISP or the Regional Internet Registries below

ARIN - American IPv6 registration services
APNIC- Asia Pacific Network Information Center
RIPE - European Regional Internet Registry

The amount of addresses allocated varies according to your network requirements. Small Internet Service Providers (ISPs) or end nodes acquire IPv6 addresses from their upstream provider. Large ISPs, for example can receive from ARIN a minimum prefix of /48 with a second-level allocation of 16 bits for subnets. The remaining 64 bits are for a network interface

For Development Networks

Contact the 6bone (www.6bone.net). The 6bone provides a testbed for deployment of IPv6.

IPv6 Address Formats

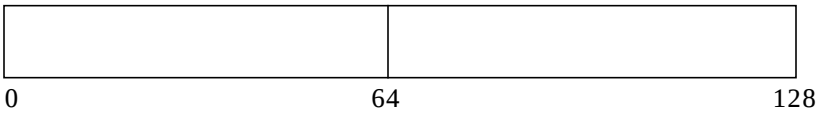
IPv6 addresses are 128 bit entities. IPv4 addresses are 32-bit addresses normally written as four decimal numbers (dotted decimal), one for each byte of the address.

Example: 10.1.3.7

IPv6 Node Addresses are 128-bit records represented as eight fields of up to four hexadecimal digits. A colon separates each field (:). Example: 8888:7777:6666:5555:4444:3333:2222:1111

To indicate a subnetwork address, IPv6 uses subnet prefixes similar to IPv4 CIDR format.Figure 5-1 shows a 128-bit IPv6 node address with a 64-bit subnet prefix.

Figure 5-1 IPv6 128-bit Addresses; HP-UX Default Prefix 64



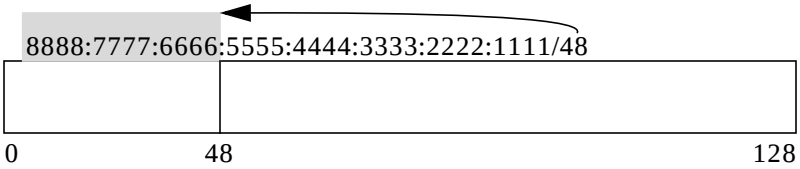
An IPv6 node address and its subnet prefix length can be combined in the format:

<IPv6-Node-Address>/<Prefix-Length>

Where <*IPv6-Node-Address*> is an IPv6 address and <*Prefix-Length*> is a decimal value specifying how many of the leftmost contiguous bits of the address compose the subnet prefix.

In Figure 5-2, prefix length 48 specifies that the leftmost 48 bits of the IPv6 address compose the subnet prefix.

Figure 5-2 Example Prefix Length 48



Address Scope

Link-local An IPv6 address used on a single link.

Site-local	An IPv6 address used inside a single site.
Global	An IPv6 address that uniquely identifies a node on the Internet such that packets can be routed to the node from any other node on the Internet.

Address Type

Unicast	Identifies a single interface. Notable unicast addresses are:
	Loopback `::1` Address internal to IPv6 stack
	Unspecified `::` Not a legally defined address
Anycast	Identifies a group of interfaces, possibly belonging to different nodes. A packet sent to an anycast address is delivered to only one of the interfaces in the group. Anycast addresses are currently not supported by HP-UX 11i IPv6.
Multicast	Identifies a group of interfaces, possibly belonging to different nodes. A packet sent to a multicast address is delivered to all the interfaces in this group.

Neighbor Discovery

IPv6 hosts and routers use the IPv6 Neighbor Discover Protocol to:

- advertise their link-layer address on the local link
- find neighbors' link-layer addresses on the local link
- find neighboring routers able to forward IPv6 packets
- actively track which neighbors are reachable
- search for alternate routers when a path to a router fails

The IPv6 Neighbor Discovery Protocol (ndp) uses ICMPv6. An IPv6-only utility, ndp(1M) and the Neighbor Discovery Protocol encompass the functionality of the IPv4 Address Resolution Protocol (ARP)and arp(1m) utility. ndp also provides some of the address-configuration functionality found in protocols BOOTP and DHCP.

A network device connecting to a network for the first time can learn all parameters necessary to function, solely through Neighbor Discovery information. Both IPv6 hosts and routers advertise their presence using neighbor advertisements and route advertisements, respectively. When an IPv6 host first comes up, it advertises its link-layer address, and solicits neighbor and router information.

For more information, see the ndp(1m) and ndp(7p) man pages and RFC 2461, “Neighbor Discovery for IP Version 6 (IPv6).”

Stateless Address Autoconfiguration

Stateless auto-configuration requires no manual configuration of hosts, minimal configuration of routers, and no additional servers. The primary interface (lanX:0) is automatically assigned a link-local address by the system when the interface is configured. This allows each IPv6 interface to have at least one source address that can be used by Neighbor Discovery. Therefore, it is not advisable to assign other addresses to the primary interface besides the link-local address. See the RFC 2373 for details.

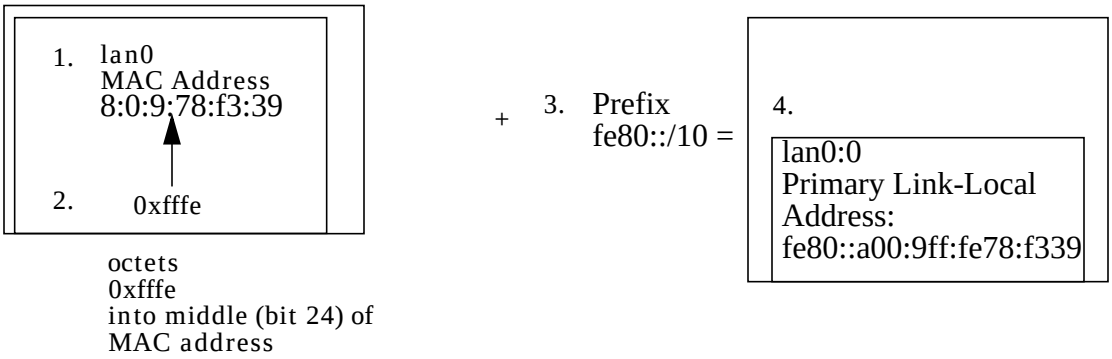
Link-local Address Assigned Automatically

A link-local address is formed by prepending the well-known link-local prefix FE80::0 to the interface identifier which is typically 64 bits long and based on EUI-64 identifiers. Link-local addresses are sufficient for allowing communication among IPv6 hosts attached to the same link.

Figure 5-3 shows the Primary Interface Autoconfiguration steps taken after using the `ifconfig` command

```
ifconfig lan0 inet6 up
```

Figure 5-3 Primary Interface Address Autoconfiguration



If you mark an interface “up” without assigning a primary address, the system derives a link-local address by:

1. Taking the LAN card’s 48-bit link-level address (“MAC address” `8:0:9:78:f3:39`)

   `0000 1000 0000 0000 0000 1001 0111 1000 1111 0011 0011 1001`

   and putting it into an EUI-64 identifier by:

2. Putting two bytes (0xffee) into the middle (bit 24) of the 48-bit link-level address `8:0:9:`**`ff:fe`**`:78:f3:39`;

`0000 1000 0000 0000 0000 1001` **`1111 1111 1111 1110`** `0111 1000 1111 0011 0011 1001`

   and flipping the Universal/local bit (as described in RFC 2373) to form a 64-bit EIU-64 **interface identifier** `a:0:9:ff:fe:78:f3:39`

`0000 1010 0000 0000 0000 1001 1111 1111 1111 1110 0111 1000 1111 0011 0011 1001`

3. Prepend the well-known prefix fe80::/10

4. Forming a 128-bit link-local unicast address for the primary interface `fe80::a00:9ff:fe78:f339`

5. Check the configuration by typing

```
ifconfig lan0 inet6
```

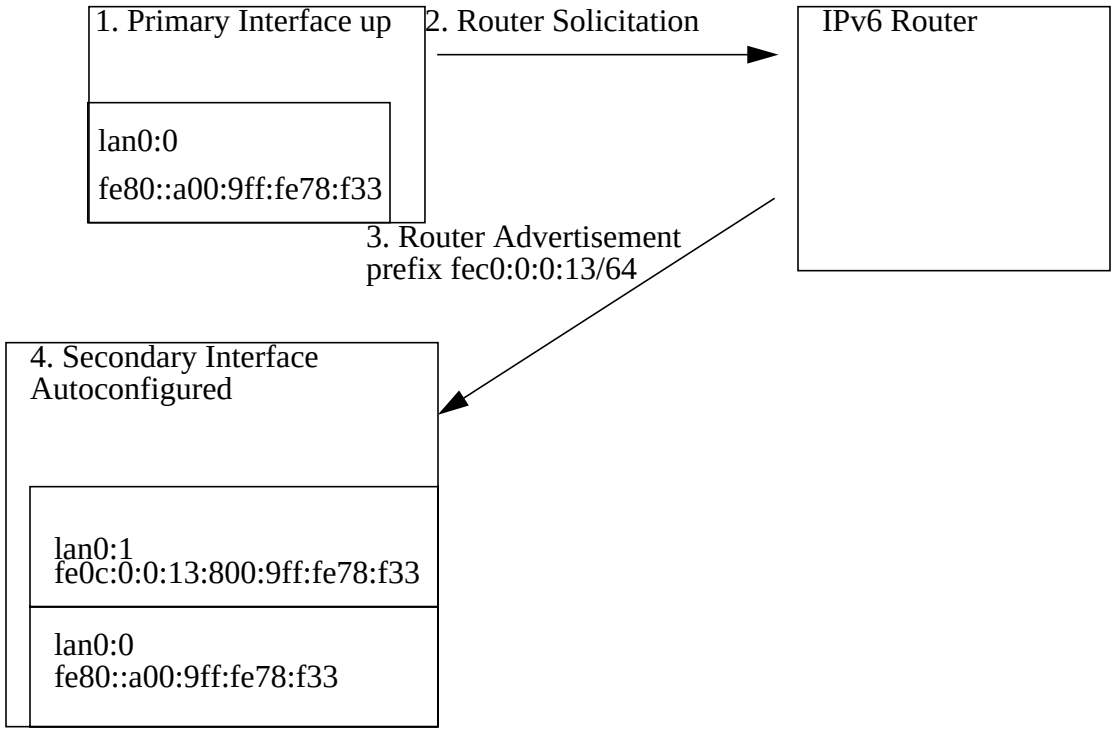
```
lan0: flags=4800841<UP,RUNNING,MULTICAST,ONLINK>
inet6 fe80::a00:9ff:fe78:f339 prefix 10
```

Secondary Interface Autoconfiguration

If an IPv6 router on the network advertises network prefixes in router advertisements, IPv6 derives a second IPv6 address based on the interface identifier. IPv6 assigns this address to a secondary interface for the network interface. The host adds the router as one of its default gateways.

Figure 5-4 shows a general example of Secondary Interface Autoconfiguration

Figure 5-4 Secondary Interface Autoconfiguration from IPv6 Router



1. Primary interface comes up. Link-local address autoconfigured.
2. Host multicasts Router Solicitation
3. IPv6 Router sends Router Advertisement to host
4. Host autoconfigures secondary interface (lan0:1) by prepending prefix (fec0:0:0:13/64) sent by router to interface identifier (a00:9ff:fe78:f33). Refer to RFC 2461 for details.

Manual Configuration and Router Advertisements

Note that even if a primary interface is manually configured, if the host receives prefixes from router advertisements, then secondary interfaces are autoconfigured. In this case, the addresses on the secondary interfaces is derived from the interface ID portion of the manually specified primary interface address.

Manual Configuration Overwriting Autoconfiguration

Manual configuration can overwrite autoconfiguration. When a secondary interface is configured with a manually assigned address, and if the user chooses an interface index number that has been used for an already autoconfigured secondary interface, the manual configuration overwrites the autoconfiguration. When this happens, network connectivity through the overwritten autoconfigured IP address is temporarily lost. At a later time, when the host receives the next router advertisement, the host will bring up another secondary interface with a different IP index number, but with the same IP address, and network connectivity through that IP address is restored. Normally, a user can avoid this by checking used IP index numbers. However, there is always a possibility that address autoconfiguration due to router advertisement is happening concurrently while the user manually configures secondary interfaces

Disabling Specific IPv6 Interfaces

To disable communication through a specific IP address on an autoconfigured secondary interface, that secondary interface should be marked down, not removed or overwritten with a different IP address. If that interface is removed or overwritten, the host will reconfigure another secondary interface with the same IP address when it receives the next router advertisement. Alternatively, the router can be configured to stop advertising the prefix that corresponds to the offending IP address.

Removing Interfaces

A primary interface cannot be removed from the system until all secondary logical interfaces are removed. You can remove secondary interfaces from the system using the `ifconfig inet6` command, as in the following example:

```
ifconfig lan1:1 inet6 ::
```

The primary interface (for example, lan1) can then be removed from the system with the `ifconfig` command, as in the following example:

```
ifconfig lan1 inet6 unplumb
```

A loopback interface does not have a hardware device associated with it. The name of the loopback interface is lo0. A loopback interface is automatically created by the system. You cannot delete it.

Networking Terminology

The following are descriptions of important IPv6 networking terms.

Nodes

A **node** is a computer on the network. A **Local node** (or host) is the computer or host where you have logged-in. A **remote node** is a computer on the network where you are not logged-in. A remote node does not have to be directly attached to your terminal.

Network Interface Name

A **network interface** is a communication device through which messages can be sent and received. An IPv6 address is associated with an interface name. Find the interface name(s) for a network interface by running the lanscan command and looking at the “Net-Interface Name PPA” field. For example

```
lanscan
Hardware Station         Crd Hdw   Net-Interface  NM  MAC       HP-DLPI DLPI
Path     Address         In# State NamePPA        ID  Type      Support Mjr#
2/0/2    0x08000978F339 0   UP     lan0 snap0     1   ETHER     Yes     119
```

The interface name may include a colon (:), followed by an interface index number that denotes the interface number. The interface index number 0 is the first interface number for a card/encapsulation type and is known as the primary interface. The interface name lan0 is equivalent to lan0:0. The syntax is:

nameX[:*interface-index-number*]

name is the class of the interface. Valid name is lan (Ethernet LAN). *X* is the Physical Point of Attachment (PPA). *interface-index-number* is the number of the interface.

You must configure the **primary interface** for a LAN card before you can configure subsequent interfaces, known as **secondary interfaces**, for the same card. For example, you must configure lan0:0 (or lan0) before you configure lan0:1 and lan0:2.

Router

A **Router** is a device that can forward packets between two or more networks. An IPv6 Router can advertise prefixes. IPv6 Router guidelines are beyond the scope of this manual. Refer to RFC 2461 for IPv6 Router guidelines.

6 IPv6 Software and Interface Technology

The following topics concern HP-UX 11i IPv6 deployment and migration.

Name and Address lookup for IPv6

The IETF [RFC 2893] recommends adding IPv6 addresses (know as A6/AAAA records) to a DNS Name Server only when the following conditions are true:

- The IPv6 address is assigned to the interface on the node
- The address is configured on the interface
- The interface is on a link which connects to the IPv6 infrastructure

HP recommends beginning with IPv6 addresses and host names in the /etc/hosts file on a development network; then adding IPv6 addresses and hosts to a Domain Name Service when moving IPv6 to a production backbone network.

/etc/nsswitch.conf (nsswitch.conf(4)) is a configuration file for the name service switch. A new entity, ipnodes, specifies which name services resolve addresses and host names.

For HP-UX 11i IPv6, the new keyword “ipnodes” specifies the resolver policy for the library functions getnameinfo(3N) and getaddrinfo(3N) for both IPv4 and IPv6 addresses. The existing keyword “hosts” is used to specify the resolver policy for the library functions gethostbyname() and gethostbyaddr() for IPv4 addresses.

If /etc/nsswitch.conf is not configured, the default ipnodes policy is

```
dns [NOTFOUND=return] files
```

The policy “dns [NOTFOUND=return] files” implies “if dns is UNAVAIL, continue on to files, and if dns returns NOTFOUND, return to the caller; in other words, treat dns as the authoritative source of information and try files only if dns is down.”

WARNING **If the “ipnodes” entry is not added to the /etc/nsswitch.conf file after the HP-UX 11i IPv6 product is installed, then address resolution for IPv6 commands or services will often fail.**

This policy matches the default “hosts” resolver policy. However, IPv6 Dual Stack commands or applications cannot resolve host names and IPv6 addresses using /etc/hosts with this policy. So Hewlett-Packard

recommends the policy

```
ipnodes: dns [NOTFOUND=continue] files
```

After HP-UX 11i IPv6 is installed, `telnet` and the `r*` commands will use ipnodes search criteria for all host name and address lookups.

Migrating Name and IPv6 Address Lookup

Most sites test IPv6 on a development subnetwork before deploying it on a larger scale. These sites typically add IPv6 address and host names to the `/etc/hosts` files on IPv6 hosts, then change their hosts lookup policy to search files.

Hewlett-Packard recommends that you maintain at least a minimal /etc/hosts file that includes important addresses like gateways, diskless boot servers and root servers, and your host's own IP address. Hewlett-Packard also recommends that you include the word files in the hosts line to help ensure a successful system boot using the `/etc/hosts` file when DNS is not available.

Migrating from IPv4 to IPv6

To successfully migrate to IPv6, maintain compatibility with the large installed base of IPv4 hosts and routers. Staying compatible with IPv4 when deploying IPv6 eases the task of moving the Internet to IPv6. HP-UX 11i IPv6 supports three of the many transitions mechanisms recommended by the IETF:

- Dual-Stack
- Automatic and Configured Tunneling IPv6 traffic through IPv4 networks
- “6to4” Connection of IPv6 Domains via IPv4 Clouds (RFC 3056)

Dual Stack

Dual Stack nodes support both IPv6 or IPv4 functionality. HP-UX IPv6 supports a dual IPv6/IPv4 protocol stack. IPv6 coexists with IPv4 applications. The Dual-Stack does not affect existing IPv4 source or binary files. Refer to RFC 2893 for details

Tunneling

Tunneling encapsulates IPv6 packets within IPv4 packets. IPv6 transmission across the IPv4 Internet is transparent. Refer to RFC 2893

Tunneling

To help migrate to IPv6, the IPv6 standards bodies recommend tunneling transition mechanisms:

IPv6 tunneling enables IPv6/IPv4 hosts and routers to connect with other IPv6/IPv4 hosts and routers over the existing IPv4 Internet. IPv6 tunneling encapsulates IPv6 datagrams within IPv4 packets. The encapsulated packets travel across an IPv4 Internet until they reach their destination host or router. The IPv6-aware host or router decapsulates the IPv6 datagrams, forwarding them as needed. IPv6 tunneling eases IPv6 deployment by maintaining compatibility with the large existing base of IPv4 hosts and routers.

IPv6/IPv4 hosts and routers can tunnel IPv6 datagrams over regions of IPv4 routing topology by encapsulating them within IPv4 packets. Tunneling can be used in a variety of ways:

Router-to-Router

IPv6/IPv4 routers interconnected by an IPv4 infrastructure can tunnel

IPv6 packets between themselves. In this case, the tunnel spans one segment of the end-to-end path that the IPv6 packet takes.

Host-to-Router IPv6/IPv4 hosts can tunnel IPv6 packets to an intermediary IPv6/IPv4 router that is reachable via an IPv4 infrastructure. This type of tunnel spans the first segment of the packet's end-to-end path.

Host-to-Host IPv6/IPv4 hosts that are interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves. In this case, the tunnel spans the entire end-to-end path that the packet takes.

Host-to-Host IPv6/IPv4 routers can tunnel IPv6 packets to their final destination IPv6/IPv4 host. This tunnel spans only the last segment of the end-to-end path.

Configured Tunneling

Tunneling techniques are classified by the way the encapsulating node determines the address of the node at the end of the tunnel. In router-to-router and host-to-router tunneling methods, the IPv6 packet is tunneled to a router. The tunnel endpoint is an intermediary router. The intermediary router decapsulates the IPv6 packet, then forwards it to its final destination. When tunneling to a router, the tunnel endpoint differs from the tunneled packet's destination. So the addresses in the tunneled IPv6 packet do not provide the IPv4 address of the tunnel endpoint. Instead, the node performing the tunneling provides configuration information that determines the tunnel endpoint address. The IETF calls this type of tunneling "configured tunneling."

Automatic Tunneling

In the host-to-host and router-to-host tunneling methods, the IPv6 packet is tunneled all the way to its final destination. The tunnel endpoint is the node to which the IPv6 packet is addressed. Because the tunnel endpoint is the IPv6 packet's destination, the IPv6 packet's destination address determines the tunnel endpoint: if that address is an IPv4-compatible IPv6 address, then the low-order 32-bits hold the destination node's IPv4 address. That can be used as the tunnel endpoint address. This technique avoids the need to explicitly configure the tunnel endpoint address. Deriving the tunnel endpoint address from the embedded IPv4 address of the packet's IPv6 address is called "automatic tunneling."

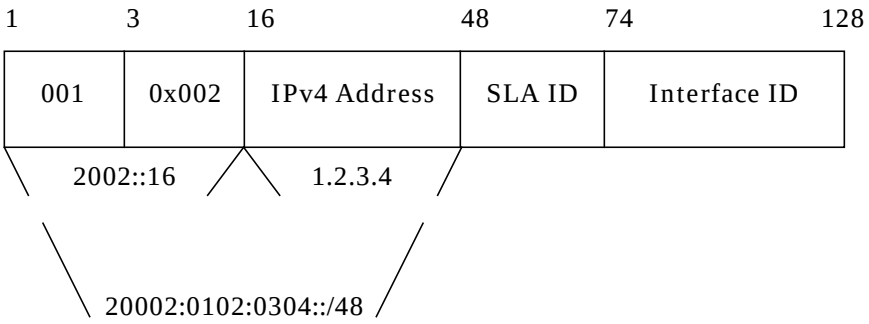
Connecting IPv6 Domains over IPv4 Clouds (6to4)

Isolated IPv6 nodes and networks can communicate over the IPv4 Internet without explicitly configuring tunnels, by implementing the “6to4” standard (RFC 3056). 6to4 effectively uses the IPv4 wide area network as a unicast point-to-point layer. 6to4 requires no end-node reconfiguration and minimal router configuration.

6to4 Well-known Prefix

The IANA permanently assigned one IPv6 address prefix for 6to4. The IPv6 prefix 2002:/16 prepends a host or router’s globally-unique 32-bit IPv4 address (<IPv4-Addr> to form a 48-bit 6to4 prefix 2002:<IPv4-Addr>. The 6to4 prefix provides a network prefix for the local IPv6 host or network. The IPv4 address is the endpoint for all external IPv4 connections.

Figure 6-1 **6to4 Prefix**



Encapsulating in IPv4

IPv6 packets from a 6to4 site are encapsulated in IPv4 packets when they leave the site over its external IPv4 connection. IPv4 packets are transmitted in IPv4 packets with an IPv4 protocol type of 41, the same protocol type set when IPv6 packets tunnel inside IPv4 frames according to RFC 2893.

Example 6to4 Topology

Figure 6-2 shows two IPv6 subnetworks. The end nodes have their routers globally unique IPv4 addresses embedded in their network prefixes. The routers have 6to4 addresses and corresponding globally unique IPv4 addresses. From the IPv6 end-node view, each host's subnetwork is connected to the other's through a 6to4 router. All IPv4 tunneling is transparent to the IPv6 end nodes.

Figure 6-2 **6to4 IPv6 End Node View**

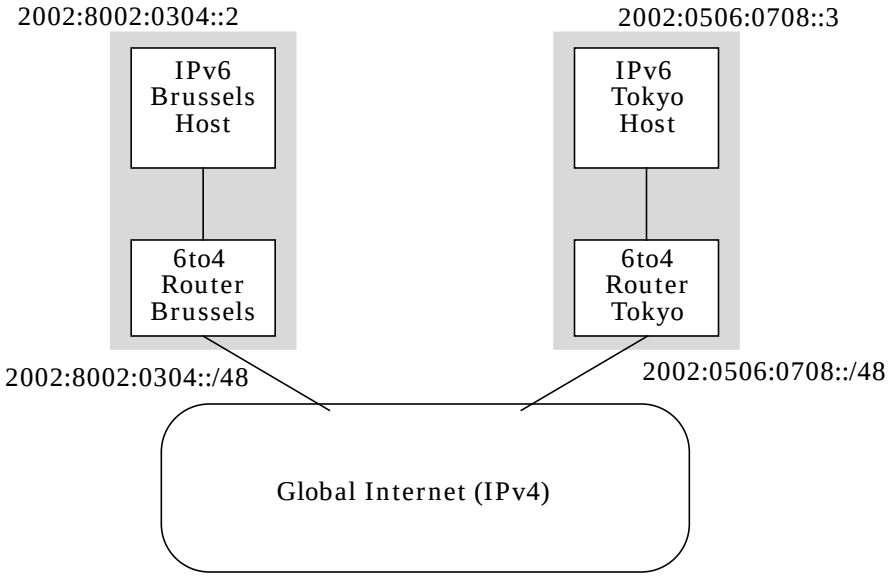
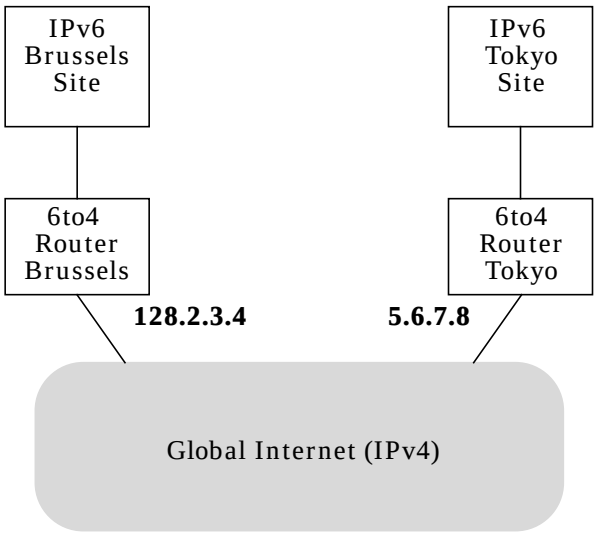


Figure 6-3 shows the same routers from the IPv4 internet perspective. Router Brussels and Router Tokyo are IPv4 Routers. All IPv6 traffic is tunneled transparently through the IPv4 internet.

Figure 6-3 6to4 IPv4 Internet View



A IPv6 ndd Tunable Parameters

The following new IPv6 tunable parameters allow advanced fine-tuning of HP-UX IPv6 11i performance. From the HP-UX prompt type

```
ndd -h
```

for a detailed description of supported IPv6 ndd parameters.

```
ip6_ill_status                (read only)
ip6_ipif_status               (read only)
ip6_ifhash_status             (read only)
ip6_ire_status                (read only)
ip6_raw_status                (read only)
ip6_tcp_status                (read only)
ip6_udp_status                (read only)
ip6_ire_hash                  (read only)
ip6_ire_hash_summary          (read only)
ip6_rput_pullups              (read and write)
ip6_fragment_timeout          (read and write)
ip6_ill_config_command        (write only)
ip6_ill_config_status         (read only)
ip6_reass_mem_limit           (read and write)
ip6_forwarding                (read and write)
ip6_send_redirects            (read and write)
ip6_debug                     (read and write)
ip6_ire_reachable_interval    (read and write)
ip6_ire_cleanup_interval      (read and write)
ip6_ire_redirect_interval     (read and write)
ip6_ire_pathmtu_interval      (read and write)
ip6_def_hop_limit             (read and write)
ip6_wroff_extra               (read and write)
ip6_dl_sap                    (read and write)
ip6_dl_snap_sap               (read and write)
ip6_bogus_sap                 (read and write)
ip6_encap_hop_limit           (read and write)
ip6_nd_dad_solicit_count      (read and write)
ip6_nd_multicast_solicit_count(read and write)
```

```
ip6_nd_unicast_solicit_count  (read and write)
ip6_nd_advertise_count        (read and write)
ip6_rd_solicit_count          (read and write)
ip6_nd_transmit_interval      (read and write)
ip6_nd_anycast_delay          (read and write)
ip6_nd_probe_delay            (read and write)
ip6_rd_solicit_delay          (read and write)
ip6_rd_transmit_interval      (read and write)
ip6_min_random_factor         (read and write)
ip6_max_random_factor         (read and write)
ip6_icmp_interval             (read and write)
```

NAME

ifconfig - configure network interface parameters

SYNOPSIS

ifconfig *interface* [*address_family*] [*address* [*dest_address*]] [*parameters*]

ifconfig *interface* [*address_family*]

DESCRIPTION

The first form of the **ifconfig** command assigns an address to a network interface and/or configures network interface parameters. **ifconfig** must be used at boot time to define the network address of each interface present on a machine. It can also be used at other times to redefine an interface's address or other operating parameters. If the *address_family* is not specified, the address family defaults to IPv4.

The second form of the command, without *address_family*, displays the current configuration for *interface*. If *address_family* is not specified, **ifconfig** reports the details on all supported address families.

Only a user with appropriate privileges can modify the configuration of a network interface. All users can run the second form of the command.

Arguments

ifconfig recognizes the following arguments:

address Either a host name present in the host name database (see *hosts*(4)), or a DARPA Internet address expressed in Internet standard dot notation (see *inet*(3N)) for an IPv4 address and in colon notation (see *inet6*(3N)) for an IPv6 address.

address_family Name of protocol on which naming scheme is based. An interface can receive transmissions in differing protocols, each of which may require separate naming schemes. The *address_family*, affects the interpretation of the remaining parameters on the command line. The only address families currently supported are **inet** (DARPA-Internet family) for IPv4 addresses, and **inet6** for IPv6 addresses.

dest_address Address of destination system. Consists of either a host name present in the host name database (see *hosts*(4)), or a DARPA Internet address expressed in Internet standard dot notation (see *inet*(3N)) for an IPv4 address, and in colon notation (see *inet6*(3N)) for an IPv6 address.

interface A string of the form *name unit*, such as **lan0**. (See the Interface Naming subsection given below.)

parameters One or more of the following operating parameters:

up Mark an interface "up". Enables interface after an **ifconfig down**. Occurs automatically when setting the address on an interface. Setting this flag has no effect if the hardware is "down". A secondary interface (see the Interface Naming subsection given below) can be marked up only if the primary interface is already up.

down Mark an interface "down". When an interface is marked "down", the system will not attempt to transmit messages through that interface. A primary interface (see the Interface Naming subsection given below) can be marked down only if all the secondary interfaces on the same physical device are already down.

broadcast (inet only) Specify the address that represents broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

metric *n* Set the routing metric of the interface to *n*. The default is 0. The routing metric is used by the routing protocol (see *gated*(1M)). Higher metrics have the effect of making a route less favorable; metrics are counted as additional hops to the destination network or host.

netmask *mask*
(inet only) Specify how much of the address to reserve for subdividing networks into sub-networks or aggregating networks into supernets. *mask* can be specified as a single hexadecimal number with a

leading **0x**, with a dot-notation Internet address, or with a pseudo-network name listed in the network table (see *networks*(4)). For subdividing networks into sub-networks, *mask* must include the network part of the local address, and the subnet part which is taken from the host field of the address. *mask* must contain 1's in the bit positions in the 32-bit address that are to be used for the network and subnet parts, and 0's in the host part. The 1's in the *mask* must be contiguous starting from the leftmost bit position in the 32-bit field. *mask* must contain at least the standard network portion, and the subnet field must be contiguous with the network portion. The subnet field must contain at least 1 bit. For aggregating networks into supernets, *mask* must only include a portion of the network part. *mask* must contain contiguous 1's in the bit positions starting from the leftmost bit of the 32-bit field.

prefix *n* (inet6 only) *n* indicates the length of the network prefix associated with this interface. The primary interface (see Interface Naming subsection given below) prefix length is always 10, and is not configurable. The *prefix* option can be used only with the *address* option, and only for secondary interfaces. Default: 64. Range: 1 to 128.

arp (inet only) Enable the user of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). If an interface already had the Address Resolution Protocol disabled, the user must "unplumb" the interface before it can be enabled for Address Resolution Protocol.

-arp (inet only) Disable the use of the Address Resolution Protocol. If an interface already had the Address Resolution Protocol enabled, the user must "unplumb" the interface before it can be disabled for Address Resolution Protocol.

plumb Setup the Streams plumbing needed for TCP/IP for a primary interface name. (See the Interface Naming subsection given below.). By default, the **plumb** operation is done automatically when an IP address is specified for an interface.

unplumb Tear down the Streams plumbing for a primary interface name. (See the Interface Naming subsection given below.) Secondary interface does not require "plumbing". A secondary IPv4 interface can be removed by assigning an IP address of 0.0.0.0 to it. Remove a secondary IPv6 interface by assigning an IP address of :: to it.

Interface Naming

The *interface* name associated with a network card is composed of the *name* of the interface (e.g. **lan** or **snap**), the *ppa number* which identifies the card instance for this interface, and an optional *IP index number* which allows the configuration of multiple IP addresses for an interface. For LAN cards, the *interface* name **lan** will be used to designate Ethernet encapsulation and **snap** for IEEE 802.3 encapsulation. The **lanscan** command can be used to display the *interface* name and *ppa number* of each interface that is associated with a network card (see *lanscan*(1M)).

IPv4 and IPv6 interfaces can coexist over the same physical network interface device using the same naming scheme. IPv6 interfaces are configured using the "inet6" ifconfig subcommand. (See the IPv6 subsection given below.)

IP Index Number

Multiple IP addresses assigned to the same *interface* may be in different subnets. An example of an interface name without an *IP index number* is **lan0**. An example of an interface name with a *IP index number* is **lan0:1**. Note: specifying **lan0:0** is equivalent to **lan0**.

A primary interface is an interface whose IP index number is zero. A secondary interface is an interface whose IP index number is non-zero.

Loopback Interface

The loopback interface (**lo0**) is automatically configured when the system boots with the TCP/IP software. The IP address and netmask of the primary IPv4 loopback interface are 127.0.0.1 and 255.0.0.0, respectively. The IP address and prefix of the primary IPv6 loopback interface are ::1 and 128 respectively. The user is not permitted to change the address of the primary loopback interface (**lo0:0**). It is permissible to assign other IP addresses to lo0 with non-zero *IP index numbers* (lo0:1, lo0:2, etc). This allows a system to have a "system IP" address that is available as long as one interface remains usable.

Supernets

(inet only) A supernet is a collection of smaller networks. Supernetting is a technique of using the netmask to aggregate a collection of smaller networks into a supernet.

This technique is particularly useful when the limit of 254 hosts per class C network is too restrictive. In those situations a netmask containing only a portion of the network part may be applied to the hosts in these networks to form a supernet. This supernet netmask should be applied to those interfaces that connect to the supernet using the *ifconfig* command. For example, a host can configure its interface to connect to a class C supernet, 192.6, by configuring an IP address of 192.6.1.1 and a netmask of 255.255.0.0 to its interface.

IPv6 Interfaces

inet6 must be specified when an IPv6 interface is configured. The address for an IPv6 interface can either be a hostname present in the host name database (see *hosts(4)),* or an address in the IPv6 colon notation.

Stateless Address Auto-configuration:

Unlike IPv4 interfaces, IPv6 interfaces can be configured without an address and/or a prefix. Stateless address autoconfiguration requires no manual configuration of hosts, minimal (if any) configuration of routers, and no additional servers. A primary interface **(lanX:0)** is automatically assigned a link-local address by the system when the interface is configured. A link-local address comprises the well-known link-local prefix FE80::0 and the interface identifier, which is typically 64 bits long and is based on EUI-64 identifiers. The link-local address allows automatic discovery of other hosts and routers on the same link, using the Neighbor Discovery Protocol (see *NDP*(7P)). The link-local address can be used as the source address to communicate with other nodes when no routers are present. If a router on the local link advertises prefixes in router advertisements, the host autoconfigures its secondary interfaces and its default gateway. The address of an autoconfigured secondary interface is formed by prepending the prefix received from the router to the interface identifier, the same interface identifier that is used in forming the primary interface.

Manual Address Configuration:

IPv6 interfaces can also be configured with manually assigned addresses and/or prefixes. A primary interface must be configured with a link-local address and the prefix must not be specified. The prefix is always 10. The universal/local bit, the U bit, of the interface identifier must be 0, per section 2.5.1 of RFC 2373. Accordingly, a manually assigned address for a primary interface must have the following pattern: FE80::xMxx:xxxx:xxxx:xxxx where x is any hexadecimal digit, and M must be 0, 1, 4, 5, 8, 9, C, or D.

When a primary interface is configured with a manually assigned address, secondary interfaces will be autoconfigured if the host receives prefixes from router advertisements. The addresses on the secondary interfaces will be derived from the interface identifier portion of manually configured address in the primary interface.

When a secondary interface is configured with a manually assigned address, and if the user chooses an IP index number that has been used for an autoconfigured secondary interface, the manual configuration overwrites the autoconfiguration. When this happens, network connectivity through the overwritten autoconfigured IP address is temporarily lost. At a later time, when the host receives the next router advertisement, the host will bring up another secondary interface with a different IP index number, but with the same IP address, and network connectivity through that IP address is restored. Normally, a user can avoid this by checking used IP index numbers. However, there is always a possibility that address autoconfiguration due to router advertisement is happening concurrently while the user manually configures secondary interfaces.

To disable communication through a specific IP address on an autoconfigured secondary interface, that secondary interface should be marked down, not removed or overwritten with a different IP address. If that interface is removed or overwritten, the host will reconfigure another secondary interface with the same IP address when it receives the next router advertisement. Alternatively, the router can be configured to stop advertising the prefix that corresponds to the offending IP address.

Tunneling interface:

The tunneling interface, tu0, enables both automatic and configured tunneling when it is marked up. Tunneling allows dual stack IPv6/IPv4 nodes to communicate over IPv4 routing infrastructures, by encapsulating the IPv6 packet inside an IPv4 packet. In automatic tunneling, the tunnel endpoint address is determined by the IPv4-compatible destination address of the IPv6 packet being tunneled. In configured tunneling, the tunnel endpoint address is configured by the user (see *route(1m)).*

IPv6 interface flags displayed:

An IPv6 interface may have three new flags that are not present in an IPv4 interface: TUNNEL, AUTO, and ONLINK. The TUNNEL flag is set for the tunnel interface (tu0). The AUTO flag is set for autoconfigured secondary interfaces. The ONLINK flag is set for interfaces with IP addresses that can be reached without going through a router.

Examples:

Stateless address autoconfiguration with link-local address

```
ifconfig lan0 inet6 up
```

Manual configuration for a primary interface with link-local address

```
ifconfig lan0 inet6 fe80::1 up
```

Manual configuration for a secondary interface with link-local address

```
ifconfig lan0:1 inet6 fe80::3 up
```

Manual configuration for a secondary interface with site-local address

```
ifconfig lan0:2 inet6 fec0::6 up
```

Manual configuration for a secondary interface with global address

```
ifconfig lan0:3 inet6 2222::4 up
```

Tunnel interface configuration

```
ifconfig tu0 inet6 up
```

DIAGNOSTICS

Messages indicate if the specified interface does not exist, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

AUTHOR

ifconfig was developed by HP and the University of California, Berkeley.

SEE ALSO

netstat(1), lanscan(1M), route(1M), inet(3N), inet6(3N), hosts(4), routing(7), NDP(7P).

IP VErsion 6 Addressing Architecture, RFC2373, Hinden, Derring.

NAME

netstat - show network status

SYNOPSIS

netstat [**-an**] [**-f** *address-family*] [*system* [*core*]]

netstat [**-Mnrsv**] [**-f** *address-family*] [**-p** *protocol*] [*system* [*core*]]

netstat [**-f** *address-family*] [**-gin**] [**-I** *interface*] [*interval*] [*system* [*core*]]

DESCRIPTION

netstat displays statistics for network interfaces and protocols, as well as the contents of various network-related data structures. The output format varies according to the options selected. Some options are ignored or invalid when used in combination with other options.

Generally, the **netstat** command takes one of the three forms shown above:

- The first form of the command displays a list of active sockets for each protocol.
- The second form displays the contents of one of the other network data structures according to the option selected.
- The third form displays configuration information for each network interface. It also displays network traffic data on configured network interfaces, optionally updated at each *interval*, measured in seconds.

Options are interpreted as follows:

-a Show the state of all sockets, including passive sockets used by server processes. When **netstat** is used without any options only active sockets are shown. This option does not show the state of X.25 programmatic access sockets. The option is ignored if the **-g**, **-i**, **-I**, **-M**, **-p**, **-r**, **-s** or *interval* option is specified.

-f *address-family*
Show statistics or address control block for only the specified *address-family*. The following address families are recognized: **inet** for **AF_INET**, **inet6** for **AF_INET6**, and **unix** for **AF_UNIX**. This option with **AF_UNIX** applies to the **-a** and **-s** options. This option with **AF_INET** or **AF_INET6** applies to the **-a**, **-i**, **-n**, and **-s** options.

-g Show multicast information for network interfaces. Only the address family **AF_INET** is recognized by this option. This option may be combined with the **-i** option to display both kinds of information. The option is ignored if the **-p** option is specified.

-i Show the state of network interfaces. Only the interfaces that have been configured with an IP address or the **plumb** option using the **ifconfig** command are shown. The output includes both the primary and logical interfaces. (See *ifconfig*(1M)). The counts for Ipkts and Opkts fields are for IP packets only. This option is ignored if the **-p** option is specified.

-I *interface* Show information about the specified interface only. This option applies to the **-g** and **-i** options.

-M Show the multicast routing tables. When **-s** is used with the **-M** option, **netstat** displays multicast routing statistics instead. This option is ignored if the **-p** option is specified.

-n Show network addresses as numbers. Normally, **netstat** interprets addresses and attempts to display them symbolically. This option applies to the **-a**, **-i**, **-r** and **-v** options.

-p *protocol* Show statistics for the specified protocol. The following protocols are recognized: **tcp**, **udp**, **ip**, **icmp**, **igmp**, **ipv6**, and **icmpv6**.

-r Show the routing tables. When **-v** is used with the **-r** option, **netstat** also displays the network masks in the route entries. This option is ignored if the **-g**, **-i**, **-I**, **-p** or *interval* option is specified and is invalid if the **-s** option is specified.

n

-s Show statistics for all protocols. When this option is used with the **-M** option, **netstat** displays multicast routing statistics instead. This option is ignored if the **-g**, **-i**, **-I**, **-p** or *interval* option is specified and is invalid if the **-r** option is specified.

-v Show additional routing information. When **-v** is used with the **-r** option, **netstat** also displays the network masks in the route entries. This option only applies to the **-r** option.

The arguments *system* and *core* allow substitutes for the defaults, **/stand/vmunix** and **/dev/kmem**.

If no options are specified, **netstat** displays the status of only active sockets. The display of active and passive sockets status shows the local and remote addresses, send and receive queue sizes (in bytes), protocol, and the internal state of the protocol. Address formats are in two forms: *host***.***port*, or *network***.***port* if the host portion of a socket address is zero. When known, the host and network addresses are displayed symbolically by using **gethostbyname()** and **getnetbyname()**, respectively (see *gethostent*(3N) and *getnetent*(3N)) for IPv4, and *getnameinfo()* for IPv6 (see *getaddrinfo*(3N)). If a symbolic name for an address is unknown, the address is displayed numerically according to the address family. For more information regarding the Internet "dot format" for IPv4 addresses, refer to *inet*(3N). For more information regarding the Internet "colon format" for IPv6 addresses, refer to *inet6*(3N). Unspecified or "wildcard" addresses and ports appear as an asterisk (*****).

The interface display provides a table of cumulative statistics regarding packets transferred, both inbound and outbound. The network addresses of the interface and the maximum transmission unit (MTU) are also displayed. When the *interval* argument is specified, **netstat** displays a running count of statistics related to network interfaces. This display consists of a column for the first interface found during autoconfiguration and a column summarizing information for all interfaces. To display a running count of statistics for a specific interface, use the **-I** option. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.

The routing table display indicates the available routes and their status. Each route consists of a destination host or network, a netmask and a gateway to use in forwarding packets. The **Flags** field shows whether the route is up (**U**), whether the route is to a gateway (**G**), or whether the route is a host or network route (with or without **H**).

The **Netmask** field shows the mask to be applied to the destination IP address of an IP packet to be forwarded. The result will be compared with the destination address in the route entry. If they are the same, then the route is one of the candidates for routing this IP packet. If there are several candidate routes, then the route with the longest **Netmask** field (contiguous 1's starting from the left-most bit position) will be chosen. (see *routing*(7).)

The **Gateway** field shows the address of the immediate gateway for reaching the destination. It can be the address of the outgoing interface if the destination is on a directly connected network.

The **Interface** field identifies which network interface is used for the route.

The **Pmtu** field displays the path maximum transmission unit (PMTU). If the route is created with a static PMTU value (see *route*(1M)), the corresponding PMTU value permanently overrides the interface MTU. Otherwise, the PMTU value is the same as the MTU of the network interface used for the route.

The **Prefix** field is for IPv6 only. Its format is similar to the CIDR notation in IPv4. The prefix is an integer between 0 and 128 inclusive. It specifies how many of the leftmost contiguous bits of the address comprise the prefix. A host route has a prefix of 128. A default route has a prefix of 0 (see *route*(1M)). The prefix is also used in selecting a route to forward an IPv6 packet.

DEPENDENCIES

X.25:

-a option does not list X.25 programmatic access information.

AUTHOR

netstat was developed by HP and the University of California, Berkeley.

SEE ALSO

ifconfig(1M), lanscan(1M), lanadmin(1M), route(1M), inet(3N), inet6(3N), gethostent(3N), getnetent(3N), getaddrinfo(3N). hosts(4), networks(4), protocols(4), services(4), routing(7).

NAME

ping - send ICMP Echo Request packets to network host

SYNOPSIS

ping [**-oprv**] [**-f** *address-family*] [**-i** *address*] [**-t** *ttl*] *host* [**-n** *count*]

ping [**-oprv**] [**-f** *address-family*] [**-i** *address*] [**-t** *ttl*] *host packet-size* [[**-n**] *count*]

DESCRIPTION

The **ping** command sends ICMP Echo Request (ECHO_REQUEST) packets to the *host* once per second. Each packet that is echoed back via an ICMP Echo Response packet is written to the standard output, including round-trip time.

ICMP Echo Request datagrams ("pings") have an IP and ICMP header, followed by a **struct timeval** (see *gettimeofday*(2)) and an arbitrary number of "pad" bytes used to fill out the packet. The default datagram length is 64 bytes, but this can be changed by using the *packet-size* option.

Options

The following options and parameters are recognized by **ping**:

-i *address* If *host* is a multicast address, send multicast datagrams from the interface with the local IP address specified by *address* in "dot" notation (see *inet*(3N)). If the **-i** option is not specified, multicast datagrams are sent from the default interface, which is determined by the route configuration.

-o Insert an IP Record Route option in outgoing packets, summarizing routes taken when the command terminates.

It may not be possible to get the round-trip path if some hosts on the route taken do not implement the IP Record Route option. A maximum of 9 Internet addresses can be recorded due to the maximum length of the IP option area.

-p The new Path MTU information is displayed when a ICMP **Datagram Too Big** message is received from a gateway. The **-p** option must be used in conjunction with a large *packetsize* and with the **-v** option.

-r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-connected network, an error is returned. This option can be used to ping the local system through an interface that has no route through it, such as, after the interface was dropped by **gated** (see *gated*(1M)).

-t *ttl* If *host* is a multicast address, set the time-to-live field in the multicast datagram to *ttl*. This controls the scope of the multicast datagrams by specifying the maximum number of external systems through which the datagram can be forwarded.

If *ttl* is zero, the datagram is restricted to the local system. If *ttl* is one, the datagram is restricted to systems that have an interface on the network directly connected to the interface specified by the **-i** option. If *ttl* is two, the datagram can be forwarded through one multicast router at the most; and so forth. *Range*: zero to 255. The default value is 1.

-v Verbose output. Show ICMP packets other than Echo Responses that are received.

-f *address-family*
The *address-family* determines whether the *host* is an IPv4 or IPv6 host. The address families currently supported are **inet** for IPv4 addresses and **inet6** for IPv6 addresses.

host Destination to which the ICMP Echo Requests are sent. *host* can be a hostname or an IPv4 or IPv6 Internet address. All symbolic names specified for *host* are looked up by using **gethostbyname**() (see *gethostent*(3N)) for IPv4, and **getaddrinfo**() (see *getaddrinfo*(3N)) for IPv6. If *host* is an Internet address, it must be in "dot" notation (see *inet*(3N)) for IPv4, and in "colon" notation (see *inet6*(3N)) for IPv6.

If the *address-family* is specified, and *host* is an Internet address, the address family of the Internet address must be the same as that specified in the *address-family* option. If the *address-family* is not specified, and *host* is a symbolic name, an attempt will be made to resolve *host* into an IPv4 address first. If that fails, a second attempt will be made to resolve *host* into an IPv6 address.

The ping command does not accept IPv4-mapped IPv6 addresses. To ping an IPv4 node, an IPv4 address should be used. IPv4-mapped IPv6 addresses are used to address IPv4-only nodes from an IPv6 node in a socket program only. IPv4-mapped IPv6 addresses are always converted to an IPv4 address before they are used in packets sent over the network.

If a system does not respond as expected, the route might be configured incorrectly on the local or remote system or on an intermediate gateway, or there might be some other network failure. Normally, *host* is the address assigned to a local or remote network interface.

(inet only) If *host* is a broadcast address, all systems that receive the broadcast should respond. Normally, these are only systems that have a network interface on the same network as the local interface sending the ICMP Echo Request.

If *host* is a multicast address, only systems that have joined the multicast group should respond. These may be distant systems if the **-t** option is specified, and there is a multicast router on the network directly connected to the interface specified by the **-i** option.

packet-size The size of the transmitted packet, in bytes. By default (when *packet-size* is not specified), the size of transmitted packets is 64 bytes. The minimum value allowed for *packet-size* is 8 bytes, and the maximum value is 4095 bytes. If *packet-size* is smaller than 16 bytes, there is not enough room for timing information. In that case, the round-trip times are not displayed.

count The number of packets **ping** will transmit before terminating. *Range*: zero to 2147483647. The default is zero, in which case **ping** sends packets until interrupted.

Using ping for Fault Isolation

When using **ping** for fault isolation, first specify a local address for *host* to verify that the local network interface is working correctly. Then specify host and gateway addresses further and further away to determine the point of failure. **ping** sends one datagram per second, and it normally writes one line of output for every ICMP Echo Response that is received. No output is produced if there are no responses. If an optional *count* is given, only the specified number of requests is sent. Round-trip times and packet loss statistics are computed. When all responses have been received or the command times out (if the *count* option is specified), or if the command is terminated with a **SIGINT**, a brief summary is displayed.

This command is intended for use in testing, managing and measuring network performance. It should be used primarily to isolate network failures. Because of the load it could impose on the network, it is considered discourteous to use **ping** unnecessarily during normal operations or from automated scripts.

AUTHOR

ping was developed in the Public Domain.

FILES

/etc/hosts

SEE ALSO

getaddrinfo(3N), gethostent(3N), inet(3N), inet6(3N).

NAME

route - manually manipulate the routing tables

SYNOPSIS

/usr/sbin/route [**-f**] [**-n**] [**-p** *pmtu*] **add** [**net**|**host**] *destination* [**netmask** *mask*] *gateway* [*count*]

/usr/sbin/route inet6 [**-f**] [**-n**] [**-p** *pmtu*] [**-t**] **add** [**net**|**host**] *v6destination* [/ *prefix*] *v6gateway* [*count*]

/usr/sbin/route [**-f**] [**-n**] **delete** [**net**|**host**] *destination* [**netmask** *mask*] *gateway* [*count*]

/usr/sbin/route inet6 [**-f**] [**-n**] [**-t**] **delete** [**net**|**host**] *v6destination* [/ *prefix*] *v6gateway* [*count*]

/usr/sbin/route -f [**-n**]

/usr/sbin/route inet6 -f [**-n**]

DESCRIPTION

The **route** command manipulates the network routing tables manually. You must have appropriate privileges.

Subcommands

The following subcommands are supported.

add Add the specified host or network route to the network routing table. If the route already exists, a message is printed and nothing changes.

delete Delete the specified host or network route from the network routing table.

Options and Arguments

route recognizes the following options and arguments.

inet6 Specifies an IPv6 route. When this option is used, the destination and the gateway must have IPv6 addresses. When this option is not used, the command defaults to an IPv4 route and the destination and the gateway must have IPv4 addresses.

-f Delete all route table entries that specify a remote host for a gateway. If this is used with one of the subcommands, the entries are deleted before the subcommand is processed.

-n Print any host and network addresses in Internet "dot" notation for IPv4 and in "colon" notation for IPv6, except for the default network address, which is printed as **default**.

-t (inet6 only) Specifies this route as a configured tunnel. A tunnel route may be created for a single host or for a subnet. The options **inet6** and **-t** are both required for configured tunnels. The *v6gateway* address must be an IPv4-compatible IPv6 address to which packets for the address range will be sent. *count* must be greater than zero (0) for configured tunnels. When an IPv6 packet is transmitted over a configured tunnel, the tunnel endpoint address configured for that tunnel is used as the destination address for the encapsulating IPv4 header. IPv6-in-IPv6 tunnels are not yet supported.

-p *pmtu* Specifies a path maximum transmission unit (MTU) value for a static route. The minimum value allowed is 68 bytes for IPv4 and 1280 bytes for IPv6; the maximum is the MTU of the outgoing interface for this route. This option can be applied to both host and network routes.

net or **host** The type of *destination* address. If this argument is omitted, routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. For IPv4, if the *destination* has a local address part of **INADDR_ANY(0)** , the route is assumed to be to a network; otherwise, it is treated as a route to a host. For IPv6, if the *destination* has an address that is less than 128 bits, including any leading and trailing 0's, the route is assumed to be a network; otherwise, it is treated as a route to a host. An exception is the IPv6 "Unspecified Address", typically represented as ::, which is always interpreted as the default

network route.

destination (inet only) The destination host system where the packets will be routed. *destination* can be one of the following:

- A host name (the official name or an alias, see *gethostent*(3N)).
- A network name (the official name or an alias, see *getnetent*(3N)).
- An Internet address in "dot" notation (see *inet*(3N)).
- The keyword **default**, which signifies the wildcard gateway route (see *routing*(7)).

v6destination (inet6 only) The destination host system where the packets will be routed. *v6destination* can be one of the following:

- A host name (the official name or an alias, see *getipnodebyname*(3N)).
- An IPv6 address in "colon" notation (see *inet6*(3N)).
- The keyword **default**, which signifies the wildcard gateway route.

prefix (inet6 only) The *prefix* is an integer between 0 and 128 inclusive. It specifies how many of the leftmost contiguous bits of the v6destination address comprise the prefix. Its format is similar to the CIDR notation in IPv4. A *prefix* of 0 would be a default route. If the *prefix* is omitted when adding a network route, then the *prefix* would be 64 by default. It is advisable to specify the *prefix* when an IPv6 network route is added. The *prefix* option can be applied to network routes only.

netmask
mask (inet only) The mask that will be bit-wise ANDed with *destination* to yield a net address where the packets will be routed. *mask* can be specified as a single hexadecimal number with a leading **0x**, with a "dot-notation" Internet address, or with a pseudo-network name listed in the network table (see *networks*(4)). The length of the *mask*, which is the number of contiguous 1's starting from the left-most bit position of the 32-bit field, can be shorter than the default network mask for the *destination* address. (see *routing*(7)). If the *netmask* option is not given, *mask* for the route will be derived from the *netmasks* associated with the local interfaces. (see *ifconfig*(1M)). *mask* will be defaulted to the longest *netmask* of those local interfaces that have the same network address. If there is not any local interface that has the same network address, then *mask* will default to the default value of network mask of *destination.*

gateway (inet only) The gateway through which the destination is reached. *gateway* can be one of the following:

- A host name (the official name or an alias, see *gethostent*(3N)).
- An Internet address in "dot" notation (see *inet*(3N)).

v6gateway (inet6 only) The gateway through which the destination is reached. *v6gateway* can be one of the following:

- A host name (the official name or an alias, see *getipnodebyname*(3N)).
- An IPv6 address in "colon" notation (see *inet6*(3N)).

count An integer that indicates whether the gateway is a remote host or the local host. If the route leads to a destination through a remote gateway, *count* should be a number greater than 0. If the route leads to *destination* and the gateway is the local host, *count* should be 0. The default for *count* is zero. The result is not defined if *count* is negative.

Operation

All symbolic names specified for a *destination* or *gateway* are looked up first as a host name using **gethostbyname()** for IPv4 and **getaddrinfo()** for IPv6; if the host name is not found, the *destination* is searched for as a network name using **getnetbyname()** for IPv4 only. *destination* and *gateway* can be in "dot" notation (see *inet*(3N)). *v6destination* and *v6gateway* can be in "colon" notation (see *inet6*(3N)).

If the **-n** option is not specified, any host and network addresses are displayed symbolically according to the name returned by **gethostbyaddr()** and **getnetbyaddr()**, respectively, except for the default network address (printed as **default**) and addresses that have unknown names. Addresses with unknown names are printed in Internet "dot" notation (see *inet*(3N)).

If the **-n** option is specified, any host and network addresses are printed in Internet "dot" notation except for the default network address which is printed as **default**.

If the **-f** option is specified, **route** deletes all route table entries that specify a remote host for a gateway. If it is used with one of the subcommands described above, the entries are deleted before the subcommand is processed.

Path MTU Discovery is a technique for discovering the maximum size of an IP datagram that can be sent on an internet path without causing datagram fragmentation in the intermediate routers. In essence, a source host that utilizes this technique initially sends out datagrams up to the the size of the outgoing interface. The Don't Fragment (DF) bit in the IP datagram header is set. As an intermediate router that supports Path MTU Discovery receives a datagram that is too large to be forwarded in one piece to the next-hop router and the DF bit is set, the router will discard the datagram and send an ICMP Destination Unreachable message with a code meaning "fragmentation needed and DF set". The ICMP message will also contain the MTU of the next-hop router. When the source host receives the ICMP message, it reduces the path MTU of the route to the MTU in the ICMP message. With this technique, the host route in the source host for this path will contain the proper MTU.

The **-p** *pmtu* option is useful only if you know the network environment well enough to enter an appropriate *pmtu* for a host or network route. IP will fragment a datagram to the *pmtu* specified for the route on the local host before sending the datagram out to the remote. It will avoid fragmentation by routers along the path, if the *pmtu* specified in the **route** command is correct.

ping can be used to find the *pmtu* information for the route to a remote host. The *pmtu* information in the routing table can be displayed with the **netstat -r** command (see *netstat*(1)).

The loopback interface **(lo0)** is automatically configured when the system boots with the TCP/IP software. For IPv4, the default IP address and netmask of the loopback interface are 127.0.0.1 and 255.0.0.0, respectively. For IPv6, the default IP address and prefix of the loopback interface are ::1 and 128, respectively.

When **lo0** is configured, the 127.0.0.0 loopback route for IPv4 and the ::1 loopback route for IPv6 are set up automatically so that packets for any 127.*.*.* address and ::1 will loop back to the local host. Users cannot add or delete any 127.*.*.* or ::1 loopback routes.

IPv6 Operation

The keyword **inet6** is required for adding or deleting IPv6 routes.

Examples

add a direct IPv6 host route

```
route inet6 add 2345::1 4444::3
```

add an indirect IPv6 (sub)network route

```
route inet6 add net 2222::/64 4567::8 1
```

delete an indirect IPv6 (sub)network route

```
route inet6 delete net 2222::/64 4567::8 1
```

add a configured tunnel for a remote network

```
route inet6 -t add net 2233::/66 ::12.13.14.15 1
```

Output

add *destination* **: gateway** *gateway*

The specified route is being added to the tables.

delete *destination* **: gateway** *gateway*

The specified route is being deleted from the tables.

Flags

The values of the *count* and *destination* type fields in the **route** command determine the presence of the **G** and **H** flags in the **netstat -r** display and thus the route type, as shown in the following table.

Count	Destination Type	Flags	Route Type
=0	network	**U**	Route to a network directly from the local host
>0	network	**UG**	Route to a network through a remote host gateway
=0	host	**UH**	Route to a remote host directly from the local host
>0	host	**UGH**	Route to a remote host through a remote host gateway
=0	**default**	**U**	Wildcard route directly from the local host
>0	**default**	**UG**	Wildcard route through a remote host gateway

DIAGNOSTICS

The following error diagnostics can be displayed:

add a route that already exists

The specified entry is already in the routing table.

delete a route that does not exist

The specified route was not in the routing table.

cannot update loopback route

Routes for any 127.*.*.* loopback destination cannot be added or deleted.

WARNINGS

Reciprocal **route** commands must be executed on the local host, the destination host, and all intermediate hosts if routing is to succeed in the cases of virtual circuit connections or bidirectional datagram transfers.

The HP-UX implementation of **route** does not presently support a **change** subcommand.

AUTHOR

route was developed by the University of California, Berkeley.

FILES

/etc/networks
/etc/hosts

SEE ALSO

netstat(1), ifconfig(1M), ndd(1M), ping(1M), getsockopt(2), recv(2), send(2), getaddrinfo(3N), gethostent(3N), getnetent(3N), inet(3N), inet6(3N), routing(7).

r

NAME

ndp - IPv6 Neighbor Discovery cache display and control

SYNOPSIS

ndp *host*

ndp [**-i** *interface*] [**-n**] **-a**

ndp [**-i** *interface*] [**-n**] **-A** *interval*

ndp [**-i** *interface*] **-d** *host*

ndp [**-i** *interface]* **-F**

ndp [**-i** *interface*] **-p**

ndp -s *interface host hw_addr* [**pub**]

ndp -f *filename*

DESCRIPTION

The **ndp** command displays and modifies the IPv6 Neighbor Discovery cache as specified in the IPv6 Neighbor Discovery (ND) protocol.

Options

ndp recognizes the following options and arguments:

host Display the current Neighbor Discovery cache entries for the host specified by *host*, which is either a name present in the hostname database (see *hosts*(4)), or an IPv6 address expressed in colon notation (see *inet6*(3N)).

-i *interface*
Select the Neighbor Discovery cache entries for the specified *interface*. There is no distinction between primary and secondary interfaces. Therefore, specifying "-i lan1:1" is the same as specifying "-i lan1".

-n Display host addresses in IPv6 colon notation. If this option is not specified, **ndp** attempts to display host addresses symbolically first, and falls back to displaying the host addresses in IPv6 colon notation if that failed.

-a Display all Neighbor Discovery cache entries.

-A *interval*
Continuously display all Neighbor Discovery cache entries, updated at each *interval*, measured in seconds.

-d *host* Deletes Neighbor Discovery cache entries with IP addresses that are not associated with local interfaces for the host specified by *host*.

-F Deletes all Neighbor Discovery cache entries with IP addresses that are not associated with local interfaces. These are entries with the "L" flag set.

-p Display the prefix list in the Neighbor Discovery cache table. The prefix list defines a set of IP addresses that are on-link which can be reached directly without going through a router. The prefix flags are L for onlink, and A for autonomous. The autonomous flag indicates that the prefix came from stateless autoconfiguration.

-s *interface host hw_addr* **[pub]**
Create a Neighbor Discovery cache entry for the interface specified by *interface*, the host specified by *host*, and the hardware address (link-layer address) specified by *hw_addr*. The *hw_addr* is specified as xx:xx:xx:xx:xx:xx, where each "x" is a hexadecimal digit. If **pub** is specified, the entry is published, which means that this system will respond to Neighbor Solicitation for the specified "host" even though the host address is not its own.

ndp -f *filename*
Create Neighbor Discovery cache entries from the specifications found in the file specified by *filename*. Each entry in this file specifies the interface, host, hw_addr, and optionally the pub flag. For example, the content of this file can be:

```
nodea 1:2:3:4:5:6 lan0
nodeb 2:3:4:5:6:7 lan1 pub
```

The use of **-d**, **-F**, **-s**, and **-f** options requires root privileges.

Contents

A Neighbor Discovery cache entry includes the following fields:

- host (neighbor's host name or IP address)
- hardware address (link layer address) of host
- interface name
- state
- flags

The state of an entry can be INCOMPLETE, REACHABLE, STALE, or PROBE. An entry is in an INCOMPLETE state if address resolution is in progress and the hardware address of the neighbor has not been determined. An entry is in a REACHABLE state if the neighbor is known to have been reachable recently. An entry is in a STALE state if the neighbor is no longer known to be reachable. However, no attempt has been made to verify its reachability because no traffic has been sent to this neighbor. An entry is in a PROBE state if the neighbor is no longer known to be reachable, and unicast Neighbor Soliciation probes have been sent to verify reachability. The flags can be D (deprecated), L (local), or P (published). A deprecated address can be used for receiving packets, but it should not be used for sending packets because its validity is expected to expire soon. The local flag indicates that this Neighbor Discovery cache entry corresponds to an interface on this host. The published flag indicates that the host will respond to Neighbor Solicitations on this IPv6 address.

DIAGNOSTICS

ndp returns a non-zero value to indicate errors. A zero return value indicates success.

EXAMPLES

The following **netstat** output shows the local interfaces and the IP addresses assigned to them.

```
# netstat -inf inet6
```

```
Name     Mtu Address/Prefix                Ipkts   Opkts
lan1    1500 fe80::210:83ff:fef7:3a21/10     982     759
lan1:1  1500 fec0::9:210:83ff:fef7:3a21/64     0       0
lan3    1500 fe80::210:83ff:fef7:7a9d/10       0       0
lo0     4136 ::1/128                          57      57
```

To display the entire Neighbor Discovery cache:

```
# ndp -a -n
```

```
Destination            Physical Address  Interface State    Flags fe80::202:fdff:fe36:8720  0:2:fd:36:87:20
lan1       STALE      C    fec0::9:210:83ff:fef7:3a21   0:10:83:f7:3a:21   lan1:1      REACHABLE   LP
fe80::210:83ff:fef7:3a21      0:10:83:f7:3a:21     lan1        REACHABLE    LP    fe80::210:83ff:fef7:7a9d
0:10:83:f7:7a:9d  lan3     REACHABLE LP
```

To show Neighbor Discovery cache entries for a host:

```
# ndp fe80::210:83ff:fef7:3a21
```

```
Destination            Physical Address  Interface State    Flags fe80::210:83ff:fef7:3a21  0:10:83:f7:3a:21
lan1     REACHABLE LP
```

To show Neighbor Discovery cache entries for an interface:

```
# ndp -in lan3 -a
```

```
Destination            Physical Address  Interface State    Flags fe80::210:83ff:fef7:7a9d  0:10:83:f7:7a:9d
lan3     REACHABLE LP
```

To delete a Neighbor Discovery cache entries for a host and an interface:

```
# ndp -i lan1 -d fe80::202:fdff:fe36:8720
```

```
fe80::202:fdff:fe36:8720 (fe80::202:fdff:fe36:8720) deleted.
```

To show the prefix list:

```
# ndp -p
```

```
Prefix List          Interface  Valid     Preferred  Flags
 Entries                        Lifetime  Lifetime
 fec0:0:0:9::/64       lan1      167       107      A
```

n

```
fe80::/10          lan1    inf     inf     LA
fe80::/10          lan3    inf     inf     LA
```

To add an entry in the Neighbor Discovery cache:

```
# ndp -s lan3 nodeb 0:1:2:3:4:5
```

nodeb (2001::1) added.

```
# ndp -a -n -i lan3
```

Destination Physical Address Interface State Flags fe80::210:83ff:fef7:7a9d 0:10:83:f7:7a:9d lan3 REACHABLE LP 2001::1 0:1:2:3:4:5 lan3 REACHABLE C

To flush all remote entries:

```
# ndp -F
```

nodea (fe80::202:fdff:fe36:8720) deleted.
nodeb (2001::1) deleted.

AUTHOR

ndp was developed by HP.

SEE ALSO

hosts(4), inet6(3N), NDP(7P).

Neighbor Discovery for IP Version 6 (IPv6), RFC2461, Narten, Nordmark, Simpson.

IPv6 Stateless Address Autoconfiguration, RFC2462, Thomson, Narten.

n

NAME

nettl - control network tracing and logging

SYNOPSIS

/usr/sbin/nettl -start

/usr/sbin/nettl -stop

/usr/sbin/nettl -firmlog 0|1|2 -card *dev_name ...*

/usr/sbin/nettl -log *class ...* **-entity** *subsystem ...*

/usr/sbin/nettl -status [**log** |**trace** |**all**]

/usr/sbin/nettl -traceon *kind ...* **-entity** *subsystem ...* [**-card** *dev_name ...*]
[**-file** *tracename*] [**-m** *bytes*] [**-size** *portsize*] [**-tracemax** *maxsize*] [**-n** *num_files*]

/usr/sbin/nettl -traceoff -entity *subsystem ...*

DESCRIPTION

The **nettl** command is a tool used to capture network events or packets. Logging is a means of capturing network activities such as state changes, errors, and connection establishment. Tracing is used to capture or take a snapshot of inbound and outbound packets going through the network, as well as loopback or header information. A subsystem is a particular network module that can be acted upon, such as **ns_ls_driver**, or **SX25L2**. **nettl** is used to control the network tracing and logging facility.

Except for the **-status** option, **nettl** can be used only by users who have an effective user ID of 0.

Options

nettl recognizes the following options, which can be used only in the combinations indicated in SYNOPSIS. Some option and argument keywords can be abbreviated as described below. All keywords are case-insensitive.

-start (Abbr.: **-st**)
Used alone without other options.

Initialize the tracing and logging facility, start up default logging, and optionally start up console logging. Logging is enabled for *all* subsystems as determined by the **/etc/nettlgen.conf** file. Log messages are sent to a log file whose name is determined by adding the suffix **.LOG000** to the log file name specified in the **/etc/nettlgen.conf** configuration file. Console logging is started if console logging has been configured in the **/etc/nettlgen.conf** file. See *nettlconf*(1M) and *nettlgen.conf*(4) for an explanation of the configuration file. If the log file (with suffix) already exists, it is opened in *append* mode; that is, new data is added to the file. The default name is **/var/adm/nettl** (thus logging starts to file **/var/adm/nettl.LOG000**). See "Data File Management" below for more information on how the log file is handled.

A **nettl -start** command is performed during system startup if the **NETTL** variable in the **/etc/rc.config.d/nettl** file has a value of **1**.

Note: It is strongly recommended that the tracing and logging facility be turned on before any networking is started and remain on as long as networking is being used. Otherwise, information about disasters will be lost. To minimize the impact on the system, all subsystems can be set with the **-log** option to capture only **disaster**-class log messages.

-stop (Abbr.: **-sp**)
Used alone without other options.

Terminate the trace/log facility. Once this command is issued, the trace/log facility is no longer able to accept the corresponding trace/log calls from the network subsystems.

Note: See note for the **-start** option.

-card *dev_name ...*
(Abbr.: **-c**)
This option is required by the X.25 subsystems; it is optional for other subsystems. Some subsystems do not support this option.

Limit the trace information gathered to only the data that comes from the specified network interface card. More than one *dev_name* can be specified at a time in order to trace multiple network interfaces.

dev_name specifies a device which corresponds to a network interface card that has been installed and configured. It can be either an integer representing the network interface, or the device file name of the network interface. Some subsystems do not support both types of *dev_name*. For example, the X25 subsystems require that *dev_name* be a device file name. The product documentation for the subsystems should explain if the **-card** option is applicable and how to choose an appropriate *dev_name*.

If *dev_name* is not an integer it is assumed to be a device file name. The path prefix **/dev/** will be attached in front of *dev_name* if it is not an absolute path name to form the device file name, **/dev/***dev_name*. *dev_name* must refer to a valid network device file.

-entity all
-entity *subsystem ...*
(Abbr.: **-e**)

Limit the action of **-log**, **-traceoff**, or **-traceon** to the specified protocol layers or software modules specified by *subsystem*.

The number and names of *subsystem*s on each system are dependent on the products that have been installed. Use the command **nettlconf -status** to obtain a full listing of supported subsystems and the products that own them.

Examples of OSI subsystems:

acse_pres	**ftam_init**	**mms**
asn1	**ftam_resp**	**network**
cm	**ftam_vfs**	**ots**
em	**ftp_ftam_gw**	**transport**
ftam_ftp_gw	**hps**	**ula_utils**

Examples of LAN subsystems:

ns_ls_driver	**ns_ls_loopback**	**ns_ls_x25**
ns_ls_icmp	**ns_ls_tcp**	**ns_ls_igmp**
ns_ls_nfs	**ns_ls_udp**	**ns_ls_ip**
ns_ls_ipv6	**ns_ls_icmpv6**	

Two X.25-specific subsystems are used for tracing only:

SX25L2 **SX25L3**

-file *tracename*
(Abbr.: **-f**)
Used with the first **-traceon** option only.

The first time the **-traceon** keyword is used, it initializes tracing, creating a file *tracename***.TRC000** which receives the binary tracing data. If a trace file of the name *tracename***.TRC000** already exists the binary trace data is appended to the end of the file.

To start a fresh trace file, first turn off tracing then turn it back on again using a different *tracename*. See "Data File Management" below for more information on file naming.

If **-file** is omitted, *binary* trace output goes to standard output. If standard output is a terminal device, an error message is issued and no tracing is generated.

-firmlog 0|1|2
(Abbr.: **-fm**)
Requires the **-card** option.
Series 800 and X.25 only.

Set the X.25/800 interface card logging mask to level 0, 1, or 2. The default level is 0. The X.25/800 interface logs a standard set of messages. A level of 1 specifies cautionary messages as well as the default messages. A level of 2 specifies information messages in addition to the cautionary and default messages. This option is recognized only by the **ns_ls_x25** subsystem.

-log *class* ... (Abbr.: **-l**)
Requires the **-entity** option.

Control the class of log messages that are enabled for the subsystems specified by the **-entity** option.

class specifies the logging class. Available classes are:

Full	Abbr.	Mask
informative	**i**	1
warning	**w**	2
error	**e**	4
disaster	**d**	8

informative Describes routine operations and current system values.

warning Indicates abnormal events possibly caused by subsystem problems.

error Signals an event or condition which was *not* affecting the overall subsystem or network operation, but may have caused an application program to fail.

disaster Signals an event or condition which *did* affect the overall subsystem or network operation, caused several programs to fail or the entire node to shut down.

Classes can be specified as keywords or as a single numeric mask depicting which classes to log. The mask is formed by adding the individual masks of the log classes. If you choose to indicate several classes at once, be sure to separate each log class with a space.

disaster logging is always on. The default logging classes for each subsystem is configured into the configuration file, **/etc/nettlgen.conf**. When the tracing/logging facility is started, the information in the configuration file is read and subsystems are enabled for logging with the specified classes. To change the log class, use the "**nettl -log** *class* **-entity** *subsystem*" command with a new log class value. If desired, the command can be run for different log classes and different entities.

n

-m *bytes* Specify the number of bytes (*bytes*) of each trace record to trace. This option allows the user to specify the number of bytes to be captured in the trace packet. The user may prefer not to capture an entire PDU trace, such as when the user is only interested in the header.

The maximum value for *bytes* is 2000. By default, the entire packet is traced. A value of 0 will also cause the entire packet to be traced. This option currently applies only to kernel subsystems.

-size *portsize*
(Abbr.: **-s**)
Used with first **-traceon** option only.

Set the size in kilobytes (KB) of the trace buffer used to hold trace messages until they are written to the file. The default size for this buffer is 68 KB. The possible range for *portsize* is 1 to 1024. Setting this value too low increases the possibility of dropped trace messages from fast subsystems.

-status log
-status trace
-status [**all**]
(Abbr.: **-ss**)
Used alone without other options.

Report the tracing and logging facility status. The facility must be operational, that is, **nettl -start** has been completed. The additional options define the type of trace or log information that is to be displayed. The default value is **all**.

log Log status information

trace Trace status information

all Trace and log status information

-tracemax *maxsize*
(Abbr.: **-tm**)
Used with first **-traceon** option only.

Tracing uses a circular file method such that when one file fills up, another file is used. The number of trace files that can exist on a system at any given time can be specified using the **-n** option. See "Data File Management" below for more information on file behavior.

maxsize specifies the maximum size in kilobytes (KB) of any two trace files combined. Therefore, the maximum size of each trace file will be approximately half of *maxsize* kilobytes (KB). The default value for the combined file sizes is 1000 KB. The possible range for *maxsize* is 100 to 99999.

-n *num_files* Used with first **-traceon** option only.

Specifies the number of trace files that can exist on a system at any given time. However, nettl can reduce the number of trace files depending on the available disk space. If the option is not specified, the default value is two trace files.

-traceoff (Abbr.: **-tf**)
Requires the **-entity** option.

Disable tracing of *subsystem*s specified by the **-entity** option. If **all** is specified as an argument to the **-entity** option, all tracing is disabled. The trace file remains, and can be formatted by using the **netfmt** command to view the trace messages it contains (see *netfmt*(1M)).

-traceon all
-traceon *kind* ...
(Abbr.: **-tn**)
Requires the **-entity** option. The **-card** option is required for X.25 subsystems. Other options are not required.

Start tracing on the specified subsystems. The tracing and logging facility must have been initialized by **nettl -start** for this command to have any effect. The default trace file is standard output; it can be overridden by the **-file** option. If standard output is a terminal device, then an informative message is displayed and no trace data is produced.

When tracing is enabled, every operation through the subsystems is recorded if the *kind* mask is matched.

kind defines the trace masks used by the tracing facility before recording a message. If **-traceon all** is specified, all trace masks are enabled. *kind* can be entered as one or several of the following keywords or masks:

keyword	mask	keyword	mask
hdrin	**0x80000000**	**state**	**0x04000000**
hdrout	**0x40000000**	**error**	**0x02000000**
pduin	**0x20000000**	**logging**	**0x01000000**
pduout	**0x10000000**	**loopback**	**0x00800000**
proc	**0x08000000**		

hdrin Inbound Protocol Header.

hdrout Outbound Protocol Header.

pduin Inbound Protocol Data Unit (including header and data).

pduout Outbound Protocol Data Unit (including header and data).

proc Procedure entry and exit.

state Protocol or connection states.

error Invalid events or condition.

logging Special kind of trace that contains a log message.

loopback Packets whose source and destination system is the same.

For multiple *kind*s, the masks can be specified separately or combined into a single number. For example, to enable both **pduin** and **pduout** (to trace all packets coming

into and out of the node) use either **pduin pduout** or **0x10000000 0x20000000** or the combination **0x30000000**.

Not all subsystems support all trace *kind*s. *No* error is returned if a given subsystem does not support a particular trace *kind*.

If a **-traceon** is issued on a subsystem that is already being traced, the tracing mask and optional values are changed to those specified by the new command, but the new **-file**, **-size**, and **-tracemax** options are ignored and a message is issued.

If **-entity all** is specified, all recognized subsystems are traced except X.25-specific subsystems. To turn on tracing for X.25, use the command

nettl -traceon *kind* **-e** *x.25_subsys* **-card** *dev_name*

where the value of *x.25_subsys* is **SX25L2** or **SX25L3**.

Data File Management

Data files created by the tracing and logging facility require special handling by the facility that the user must be aware of. When files are created, they have the suffix **.LOG000** or **.TRC000** appended to them, depending on whether they are log or trace files, respectively. This scheme is used to keep the files distinct for cases where the user specifies the same name in both places. Also, the files implement a type of circular buffer, with new data always going into the file appended with **.LOG000** or **.TRC000**. When a *logname***.LOG000** or *tracename***.TRC000** file is full, each log or trace is renamed to the next higher number in its sequence; that is, a file with sequence number **N** is renamed as a file with sequence number **N+1** and a new file named *logname***.LOG000** or *tracename***.TRC000** is created. The number of files that can exist simultaneously on the system can be specified by the **-n** option.

Note: The file name prefix (*logname* or *tracename*) specified by the user must not exceed eight characters so that the file name plus suffix does not exceed fourteen characters. Longer names are truncated. To see the actual name of the trace or log file, use the **nettl -status all** command.

Console Logging

Console logging is used to display significant log events on the system console. The values in the **/etc/nettlgen.conf** file determine if console logging is to be started and the entries in the **/var/adm/conslog.opts** file determine what log messages will be reported to the console. The **nettlconf** command can be used to configure and maintain the information in the **/etc/nettlgen.conf** file (see *nettlconf*(1M)). If changes are made to these files, **nettl** must be stopped and restarted for the new information to take effect.

All log messages written to the console as a result of this configuration information are in a special short form. If more information is desired on the console, the **netfmt** formatter can be used to direct output to the console device. This may be most useful in an X windows environment.

Console logging may be disabled if conservation of system resources is valued more than notification of log events.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multibyte character code sets are supported in data; single-byte character code sets are supported in file names.

EXAMPLES

1. Initialize the tracing/logging facility:

   nettl -start

   (See note for the **-start** option.)

2. Display the status of the tracing/logging facility.

   nettl -status all

3. Change log class to **error** and **warning** for all the subsystems. **disaster** logging is always on for all subsystems.

   nettl -log e w -e all

4. Turn on inbound and outbound PDU tracing for the **transport** and **session** (OTS/9000) subsystems and send binary trace messages to file **/var/adm/trace.TRC000**.

```
nettl -traceon pduin pduout -entity transport session \
     -file /var/adm/trace
```

5. Turn on outbound PDU tracing for X.25 level two and subsystems **ns_ls_ipv6** and **ns_ls_ip**. Trace messages go to the trace file set up in the previous example. This example also uses the abbreviated options. Tracing for X.25 requires a **-card** option to indicate which X.25 card to trace.

   ```
   nettl -tn pduout -e SX25L2 ns_ls_ipv6 ns_ls_ip -c x25_0
   ```

6. Determine status of tracing from the previous two examples.

   ```
   nettl -status trace
   ```

   The output should resemble the following:

   ```
   Tracing Information:
   Trace Filename:                  /var/adm/trace.TRC*
   Trace file size(Kbytes): 1000
   User's ID:               0       Buffer Size:            32768
   Messages Dropped:        0       Messages Queued:        0

   Subsystem Name:       Trace Mask:                    Card:
   TRANSPORT             0x30000000
   SESSION               0x30000000
   NS_LS_IP              0x10000000
   NS_LS_IPV6            0x10000000
   SX25L2                0x10000000                     x25_0
   ```

7. Stop tracing for all subsystems.

   ```
   nettl -traceoff -e all
   ```

8. Enable **pduin** and **pduout** tracing for **ns_ls_driver** (LAN driver) subsystem. Binary trace data goes to file **/var/adm/LAN.TRC000**.

   The **-file** option of this command is only valid the first time tracing is called. The trace file is not automatically reset with the **-file** option. To change the trace output file, stop tracing and start up again. This example assumes that the **-traceon** option is being used for the first time.

   ```
   nettl -tn pduin pduout -e ns_ls_driver -file /var/adm/LAN
   ```

9. Terminate the tracing and logging facility.

   ```
   nettl -stop
   ```

   (See note for the **-start** option.)

WARNINGS

Although the **nettl** command allows the specification of all log classes and all trace kinds for all subsystems, many subsystems do not support all log classes and all trace kinds. No error or warning will be issued if a subsystem does not support a log class or trace kind. Refer to the product documentation of the subsystem for information on supported log classes and trace kinds.

Tracing to a file that resides on a NFS file system can impact system performance and result in loss of trace data. It is recommended that NFS file systems not be used to contain tracing output files.

Tracing to a file may not be able to keep up with a busy system, especially when extensive tracing information is being gathered. If some data loss is encountered, the trace buffer size can be increased. Be selective about the number of subsystems being traced, as well as the kinds of trace data being captured.

The **nettl** and **netfmt** commands read the **/etc/nettlgen.conf** file each time they are run (see *nettl*(1M) and *netfmt*(1M)). If the file becomes corrupted, these commands will no longer be operational.

FILES

/dev/netlog	Kernel log pseudo-device file.
/dev/nettrace	Kernel trace pseudo-device file.
/etc/nettlgen.conf	Tracing and logging subsystem configuration file.
/etc/rc.config.d/nettl	Contains variables which control the behavior of **nettl** during system startup.

/var/adm/conslog.opts	Default console logging options filter file as specified in **/etc/nettlgen.conf**.
/var/adm/nettl.LOG000	Default log file as specified in **/etc/nettlgen.conf**.

AUTHOR
nettl was developed by HP.

SEE ALSO
netfmt(1M), nettlconf(1M), nettlgen.conf(4).

NAME

netfmt - format tracing and logging binary files

SYNOPSIS

/usr/sbin/netfmt [**-k**] **-s** [**-t** *records*] [[**-f**] *file_name*]

/usr/sbin/netfmt [**-k**] **-p** [**-c** *config_file*]

/usr/sbin/netfmt [**-c** *config_file*] [**-F**] [**-t** *records*] [**-v**] [**-l**] [**-n**] [**-N** | [**-l** [**-L**] [**-T**]]] [[**-f**] *file_name*]

/usr/sbin/netfmt -k [**-c** *config_file*] [**-F**] [**-t** *records*] [**-v**] [[**-f**] *file_name*]

DESCRIPTION

netfmt is used to format binary trace and log data gathered from the network tracing and logging facility (see *nettl*(1M)) and the kernel logging facility (see *kl*(1M)). The binary trace and log information can be read from a file or from standard input (if standard input is a tty device, an informative message is given and **netfmt** quits). Formatted data is written to standard output.

Formatting options are specified in an optional filter configuration file. Message inclusion and format can be controlled by the filter configuration file. If no configuration commands are specified, all messages are fully formatted.

There are two types of global formatting done by **netfmt**. The first one is global filtering for *NetTL's* trace/log packets and the other is for *KL's* log packets. A description of the filter configuration file follows the option descriptions.

Options

netfmt recognizes the following command-line options and arguments:

-k This option tells **netfmt** that the input file is a *KL* log file. This option should be specified if the user needs to log messages got from *KL* subsystems. This option cannot be specified anywhere except as the first option in the command line.

-s Display a summary of the input file. The summary includes the total number of messages, the starting and ending timestamps, the types of messages, and information about the system that the data was collected on. The contents of the input file are not formatted; only a summary is reported.

-t *records* Specifies the number of records from the tail end of the input file to format. This allows the user to bypass extraneous information at the beginning of the file, and get to the most recent information quickly. The maximum number of *records* that can be specified is 1000. If omitted, all records are formatted. The **-t** option is not allowed when the input file is a FIFO (pipe).

-f *file_name* Specifies the input file containing the binary log or trace data. *file_name* may not be the name of a tty device. Other options may impose additional restrictions on the type of the input file allowed. If omitted, data is read from standard input.

-p Parse input: this switch allows the user to perform a syntax check on the *config_file* specified by the **-c** parameter. All other parameters are ignored. If the syntax is correct, **netfmt** terminates with no output or warnings.

-c *config_file* Specifies the file containing formatter filter configuration commands. Syntax for the commands is given below. When **-c** is omitted the file **$HOME/.netfmtrc** is read for both logging and tracing filter configuration commands if it exists.

-F Follow the input file. Instead of closing the input file when end of file is encountered, **netfmt** keeps it open and continues to read from it as new data arrives. This is especially useful for watching events occur in real time while troubleshooting a problem. Another use would be for recording events to a console or hard-copy device for auditing. (Note that console logging is controlled by the configuration files **/etc/nettlgen.conf** and **/var/adm/conslog.opts**; see *nettlgen.conf*(4).) The **-F** option is not allowed when the input file is redirected.

The following options are not supported by all subsystems. If a subsystem does not support an option, that option is ignored during formatting of data from that subsystem. Consult the product documentation of the subsystem for information regarding the support of these options.

-v Enables output of verbose information. This includes additional cause and action text with formatted output. This information describes the possible cause of the message and any actions that may be required by the subsystem.

After the contents of the input file have been formatted a summary of the file is displayed. When this option is used with the **-t** option, only a summary of the last *records* is reported. No summary is produced when this option is used in conjunction with the **-F** option or if formatting is interrupted.

-l (*ell*) Turn off inverse video highlighting of certain traced fields. Use this flag when sending formatted trace data to a line printer. By default, certain fields in the trace file are highlighted in inverse video when viewing the formatted trace format at a terminal that supports highlighting.

-n Shows port numbers and network addresses(such as IP and x121) as numbers (normally, **netfmt** interprets numbers and attempts to display them symbolically).

-N Enables "nice" formatting where Ethernet/IEEE802.3, SLIP, IP, ICMP, IGMP, TCP, UDP, and RPC packets are displayed symbolically. All remaining user data is formatted in hexadecimal and ASCII.

-1 *(one)* Attempts to tersely format each traced packet on a single line. If **-L** and/or **-T** options are used, the output lines will be more than 80 characters long.

-T Places a time stamp on terse tracing output. Used with the **-1** (*minus one*) option.

-L Prefixes local link address information to terse tracing output. Used with the **-1** (*minus one*) option.

Filter Configuration File

Note: Filter configuration file syntax converges the syntax used with the obsolete **nettrfmt** network trace formatter and **netlogfmt** network log formatter commands with new **netfmt** syntax for controlling formatter options. The first section below describes the general use and syntax of the filter configuration file. Specific options for subsystem Naming and Filtering are listed in the *Subsystem Filtering* section below.

n

The filter configuration file allows specification of two types of information:

- Specify options in order to control how the input data is to be formatted. These options determine what the output looks like and allow a user to select the best format to suit their needs.
- Specify filters in order to precisely tailor what input data is to be discarded and what is to be formatted. **Global filters** control all subsystems; **subsystem filters** pertain only to specific subsystems. There are two types of **Global filters** that **netfmt** supports. The global filtering can start with either the word **formatter**, which means it is global to all the *NetTL's* subsystems and the second type starts with the word **kl_formatter**, which is used to filter *KL's* subsystems.

A filter is compared against values in the input data. If the data matches a filter, the data is formatted; otherwise, the input data is discarded. A filter can also specify *NOT* by using **!** before the filter value in the configuration file. If the input data matches a *NOT* filter, it is discarded. A filter can also be a "wildcard" (matching any value) by specifying an asterisk ***** before the filter value in the configuration file. "Wild card" filters pass all values of the input data. Specifying **!*** as the filter means *NOT ALL*.

Filter Configuration File Syntax

- The formatter ignores white space, such as spaces or tabs. However, newlines (end of line characters) are important, as they terminate comments and filter specifications.
- The formatter is not case sensitive. For example **error** and **ERROR** are treated as equivalent.
- To place comments in the file, begin each comment line with a **#** character. The formatter ignores all remaining characters on that line. There are no inline comments allowed.
- An exclamation point (**!**) in front of an argument indicates *NOT*. This operator is not supported for timestamp, log instance, and ID filtering.
- The asterisk (*****), when used as an argument, indicates *ALL*. Since the default for all formatting options is *ALL*, it is unnecessary to use the asterisk alone. It can be used along with the exclamation point, (**!***) to indicate *NOT ALL*. This operator is not available for timestamp, log instance, and ID filtering.

Global Filtering: For NetTL's Subsystems
The below explained global filtering options apply only to **NetTL's** subsystems. **NetTL's** global filtering commands start with the word **formatter**, followed by the keywords **verbosity**, **mode**, **option**, or **filter**.

formatter verbosity *value*,
value should be either of

high	Enables output of netfmt internal debugging information to standard error. Same as the **-v** option.
low	No internal debugging information is to be displayed.

formatter mode *value*,
value should be one of

raw	Dumps out the messages in hex format.
nice	Enables "nice" formatting. Same as **-N** option.
terse	Attempts to tersely format each traced packet on a single line. Same as **-1** (*minus one*) option.
normal	Normal formatting.

formatter option [**!**] *value*,
value should be

suppress	Normally repeated lines in hex output are condensed into a single line and a message stating that redundant lines have been skipped is displayed. Specifying **!suppress** will print all redundant data. This is useful when the formatted output is used as input into other commands.
highlight	Normally the formatter will highlight certain fields in its trace output in inverse video. Specifying **!highlight** will turn this feature off. Same as the **-l** (*minus ell*) option.

formatter filter *type* [**!**] *value* | *****
Six *types* of filtering are provided:

class	log classes
kind	trace kinds
id	connection, process, path, and user
log instance	specific thread of events
subsystem	subsystem names
time	specify ranges of time(s)

The following combinations are recognized:

formatter filter class *value* [*subsystem*]
value indicates the log class. This option allows the user to select one or more classes to be formatted. Initially all log classes are formatted. Only one class is allowed per line. Classes in multiple lines are logically "OR"ed. The optional *subsystem* name sets the class filter only for the specified subsystem. The log classes are:

INFORMATIVE	Describes routine operations and current system values.
WARNING	Indicates abnormal events possibly caused by subsystem problems.
ERROR	Signals an event or condition which was *not* affecting the overall subsystem or network operation, but may have caused an application program to fail.
DISASTER	Signals an event or condition which *did* affect the overall subsystem or network operation, caused several programs to fail or the entire node to shut down.

formatter filter Connection_ID *value*
formatter filter Device_ID *value*
formatter filter Path_ID *value*
formatter filter Process_ID *value*

formatter filter User_ID *value*
value specifies the ID number of the messages to format. Last-entered value has precedence over any previous ones. See the record header in the formatted output to determine which ID numbers to filter on. The **!** operator is *not* allowed in *value*.

formatter filter kind *value* [*subsystem*]
value can either be an established trace kind or a mask. A mask is a hexadecimal representation of a (set of) trace kind(s). Masks in multiple lines are logically "OR"ed. The optional *subsystem* name sets the kind filter only for the specified subsystem. Trace kinds and their corresponding masks are:

Name	Mask	Name	Mask
hdrin	0x80000000	**state**	0x04000000
hdrout	0x40000000	**error**	0x02000000
pduin	0x20000000	**logging**	0x01000000
pduout	0x10000000	**loopback**	0x00800000
proc	0x08000000		

hdrin Inbound Protocol Header.
hdrout Outbound Protocol Header.
pduin Inbound Protocol Data Unit (including header and data).
pduout Outbound Protocol Data Unit (including header and data).
proc Procedure entry and exit.
state Protocol or connection states.
error Invalid events or condition.
logging Special kind of trace that contains a log message.
loopback Packets whose source and destination system is the same.

formatter filter log_instance *value*
value specifies the log instance number of the messages to filter. Selecting a log instance allows the user to see the messages from a single thread of network events. Only one log instance is allowed per filter configuration file. The log instance can not be negated with the **!** operator.

formatter filter subsystem *value*
value specifies the subsystem name. Available subsystem names can be listed by using the command:

```
nettlconf -status
```

Only one subsystem name is allowed per line; multiple lines "OR" the request. To eliminate a given subsystem name, use the **!** operator, which formats all subsystems except those excluded by the list of negated subsystems. To include all subsystems (the default), use the ***** operator. To eliminate all subsystems, use the **!*** operator.

formatter filter time_from *value*
formatter filter time_through *value*
time_from indicates the inclusive starting time. **time_through** indicates the inclusive ending time. *value* consists of *time_of_day* and optionally *day_of_year*, (usually separated by one or more blanks for readability).

time_of_day specifies the time on the 24-hour clock in hours, minutes, seconds and decimal parts of a second (resolution is to the nearest microsecond). Hours, minutes and seconds are required; fractional seconds are optional. *time_of_day* format is *hh***:***mm***:***ss***.***ddddd*.

day_of_year specifies the day of the year in the form month/day/year in the format: *mm*/*dd*/[*yy*]*yy*. Specify month and day numerically, using one or two digits. For example, January can be specified as **1** or **01**; the third day of the month as **3** or **03**. Specify the year in four digits or by its last two digits. Only years in the ranges **1970-2037** are accepted. Two digit years in the range *70-99* are interpreted as being in the *20th century* (19*xx*) and those in the range *00-37* are interpreted as being in the *21st century* (20*xx*) (all ranges inclusive). *day_of_year* is an optional field; the current date is used as a default.

The **time_from** specification includes *only* those records starting from the resolution of time given. For example, if the *time_of_day* for **time_from** is specified as 10:08:00, all times before that, from 10:07:59.999999 and earlier, are excluded from the formatted output. Records with times of 10:08:00.000000 and later are included in the formatted output. Similarly, the **time_through** specification includes *only* up to the resolution of time given. For example, if the *time_of_day* for **time_through** is specified as 10:08:00, all records with times after that, from 10:08:00.000001 onward, are excluded from the formatted output.

Global Filtering: For KL's Subsystems

The below explained global filtering options apply only to *KL's* subsystems. *KL's* global filtering commands start with the word **kl_formatter**, followed by either **verbosity**, or **filter**.

kl_formatter verbosity *value*,
value should be either of

high	This will format the packets with the UDD, displayed along with the header of the **KL** packet
low	This will format only the header part of the **KL** packet. No UDD will be formatted. **verbosity** of This will format only the header part of the **KL** packet. No UDD will be formatted. **verbosity** of *low* is default.

kl_formatter filter *type* [**!**] *value* | *****
types of filtering are provided:

class	log classes
processor_id	specific CPU's
process_id	specific process id's
thread_id	specific thread id's
subsystem	subsystem names
time	specify ranges of time(s)

The following combinations are recognized:

kl_formatter filter class *value* [*subsystem*]
value indicates the log class. This option allows the user to select one or more classes to be formatted. Initially all log classes are formatted. Only one class is allowed per line. Classes in multiple lines are logically "OR"ed. The optional *subsystem* name sets the class filter only for the specified subsystem. The log classes are:

INFORMATIVE	Describes routine operations and current system values.
WARNING	Indicates abnormal events possibly caused by subsystem problems.
ERROR	Signals an event or condition which was *not* affecting the overall subsystem or network operation, but may have caused an application program to fail.
DISASTER	Signals an event or condition which *did* affect the overall subsystem or network operation, caused several programs to fail or the entire node to shut down.

kl_formatter filter Processor_ID *value*
kl_formatter filter Process_ID *value*
kl_formatter filter Thread_ID *value*
value specifies the ID number of the messages to format. Last-entered value has precedence over any previous ones. See the record header in the formatted output to determine which ID numbers to filter on. The **!** operator is *not* allowed in *value*.

kl_formatter filter subsystem *value*
value specifies the subsystem name. Available subsystem names can be listed by using the command:

```
nettlconf -status
```

Only one subsystem name is allowed per line; multiple lines "OR" the request. To eliminate a given subsystem name, use the **!** operator, which formats all subsystems except those excluded by the list of negated subsystems. To include all subsystems

(the default), use the ***** operator. To eliminate all subsystems, use the **!*** operator.

kl_formatter filter time_from *value*
kl_formatter filter time_through *value*
The functionality is same as in the case of **NetTL.**

Subsystem Filtering

Note: Global filtering described above takes precedence over individual subsystem tracing and logging filtering described below.

Subsystem filters are provided to allow filtering of data for individual subsystems or groups of subsystems. Their behavior varies among individual subsystems. Subsystem filters are valid only when the corresponding subsystems have been installed and configured on the system. See the subsystem documentation for a description of supported subsystem filters and their behavior.

Subsystem filtering commands start with the name of the subsystem followed by the subsystem filter keywords. However, to provide convenience and backwards compatibility, several other filter keywords are provided for the group of LAN subsystems: **NAME** and **FILTER**. Currently, four types of subsystem filters are provided: LAN, X25, STREAMS, and OTS. The collection of LAN subsystems use the subsystem filters identified by the **FILTER** and **NAME** keywords and the collection of OTS subsystems use the subsystem filters with the **OTS** keyword. The collection of X25 subsystems start their filter commands with the X25 subsystem names.

LAN Naming and Filtering

LAN naming can be used to symbolically represent numbers with more recognizable labels.

name *nodename value*
nodename is a character string to be displayed in place of all occurrences of *value*. *value* is a (IEEE802.3/Ethernet) hardware address consisting of 6 bytes specified in hexadecimal (without leading "0x"), optionally separated by **-**. **netfmt** substitutes all occurrences of *value* with *nodename* in the formatted output. The mapping is disabled when the **-n** option is used. This option applies to tracing output only.

LAN filtering is used to selectively format packets from the input file. There are numerous filter types, each associated with a particular protocol layer:

Filter Layer	Filter Type	Description
Layer 1	**dest**	hardware destination address
	source	hardware source address
	interface	software network interface
Layer 2	**ssap**	IEEE802.2 source sap
	dsap	IEEE802.2 destination sap
	type	Ethernet type
Layer 3	**ip_saddr**	IP source address
	ip_daddr	IP destination address
	ip_proto	IP protocol number
	ip6_saddr	IPv6 source address
	ip6_daddr	IPv6 destination address
	ip6_proto	IPv6 protocol number
Layer 4	**tcp_sport**	TCP source port
	tcp_dport	TCP destination port
	udp_sport	UDP source port
	udp_dport	UDP destination port
	connection	a level 4 (TCP, UDP) connection
	connection6	a level 4 (TCP, UDP) connection for IPv6
Layer 5	**rpcprogram**	RPC program
	rpcprocedure	RPC procedure
	rpcdirection	RPC call or reply

Filtering occurs at each of the five layers. If a packet matches any filter within a layer, it is passed up to the next layer. The packet must pass every layer to pass through the entire filter. Filtering starts with Layer 1 and ends with Layer 5. If no filter is specified for a particular layer, that layer is "open" and all packets pass through. For a packet to make it through a filter layer which has a filter specified, it must match the filter. Filters at each layer are logically "OR"ed. Filters between layers are logically "AND"ed.

n

LAN trace and log filters use the following format:

filter *type* [**!**] *value* | *****
filter is the keyword identifying the filter as a LAN subsystem filter.

The following filters are available for LAN tracing.

filter connection *value*
value takes the form:

local_addr:*port remote_addr*:*port*

where *local_addr* and *remote_addr* can be a hostname or a 4-byte Internet address specified in decimal dot notation (see *inet*(3N) for more information on Internet addresses and decimal dot notations). *port* can be a service name or an *integer*. *integer* represents a port and can be designated by a hexadecimal integer (**0x***digits*), an octal integer (**0***digits*), or base-10 integers (0 through 65 535).

filter connection6 *value*
value takes the form:

local_IPv6addr | *port remote_IPv6addr* | *port*

where *local_IPv6addr* and *remote_IPv6addr* can be a hostname or a 16-byte Internet address specified in colon notation (see *inet6*(3N) for more information on IPv6 Internet addresses and colon notations). *port* can be a service name or an *integer*. *integer* represents a port and can be designated by a hexadecimal integer (**0x***digits*), an octal integer (**0***digits*), or base-10 integers (0 through 65 535).

filter dest *value*
filter source *value*
value is a hardware address consisting of 6 bytes specified in hexadecimal (without leading **0x**), optionally separated by **-**.

filter dsap *value*
filter ssap *value*
value is a hexadecimal integer of the form: **0x***digit*; an octal integer of the form: **0***digits*; or a base-ten integer, 0 through 255.

filter interface *value*
value identifies a network interface and takes the form: **lan***n* for LAN interface, or **lo***n* for loopback interface, where *n* is the logical unit number, as in **lan0**.

filter ip_daddr *value*
filter ip_saddr *value*
value is a hostname or a 4-byte Internet address specified in decimal dot notation (see *inet*(3N) for more information on Internet addresses and decimal dot notations).

filter ip6_daddr *value*
filter ip6_saddr *value*
value is a hostname or a 16-byte Internet address specified in colon notation (see *inet6*(3N) for more information on Internet addresses and colon notations).

filter ip_proto *value*
filter ip6_proto *value*
value is a hexadecimal integer of the form: **0x***digit*; an octal integer of the form: **0***digits*; or a base-ten integer, 0 through 255 (see *protocols*(4) for more information on protocol numbers).

filter tcp_dport *value*
filter tcp_sport *value*
filter udp_dport *value*
filter udp_sport *value*
value is a port number designated as a 2-byte integer value or a service name. The integer value can be designated by a hexadecimal integer (**0x***digits*), an octal integer (**0***digits*), or a base-10 integer (0 through 65 535).

filter rpcprogram *value*
value is a RPC program name or an integer RPC program number (see *rpc*(4) for more information on RPC program names). The integer value can be designated by a hexadecimal integer (**0x***digits*), an octal integer (**0***digits*), or a base-10 integer (0 through 65 535).

filter rpcprocedure *value*
value is an integer RPC procedure number. The integer value can be designated by a hexadecimal integer (**0x***digits*), an octal integer (**0***digits*), or a base-10 integer (0 through 65535).

filter rpcdirection *value*
value can be either **call** or **reply**.

filter type *value*
value is a hexadecimal integer of the form: **0x***digits*; an octal integer of the form: **0***digits*; or a base-ten integer (0 through 65535).

LAN log filtering command has the following form:

filter subsystem *value*
value takes the form:

subsys_name **event** *event_list*

where *subsys_name* is a subsystem name obtained using the **nettlconf -status** command or one of the following abbreviations:

axin	**bufs**	**caselib**	**caserouter**
ip	**ipc**	**lan**	**loopback**
nsdiag	**nse**	**probe**	**pxp**
rlbdaemon	**sockregd**	**strlog**	**tcp**
timod	**tirdwr**	**udp**	**nfs**

event_list takes the form:

event_spec [**,***event_spec* **. . .**]

where *event_spec* takes one of the three forms:

[**!**] *integer* [**!**]*range* [**!**]*****

integer is an integer in hexadecimal (leading **0x**), octal (leading **0**), or decimal, which specifies a log event for the subsystem indicated.

range takes the form *integer***-***integer*, and indicates an inclusive set of events.

n

X25 Naming and Filtering
The X25 product provides capabilities to assign symbolic names to important numbers and to filter log events and trace messages. See *x25log*(1M) and *x25trace*(1M) for more information about X25 naming and filtering.

OTS Filtering
The OTS subsystem filter allows filtering of the message ID numbers that are typically found in the data portion of an OTS subsystem's log or trace record. The OTS subsystem filter is effective for any subsystem that is a member of the OTS subsystem group.

OTS trace filtering configuration commands have the following form in *config_file*:

OTS [*subsystem*] **msgid** [**!**] *message_ID* | *****

Keywords and arguments are interpreted as follows:

OTS Identifies the filter as an OTS subsystem filter.

subsystem One of the following group of OTS subsystems:

OTS	**ACSE_PRES**	**NETWORK**
TRANSPORT	**SESSION**	

Note: The absence of *subsystem* implies that the filter applies to all OTS subsystems.

message_ID is the value of the message ID to filter. A message ID is used by OTS subsystems to identify similar types of information. It can be recognized as a 4 digit number contained in brackets (**[ ]**) at the beginning of an OTS subsystem's trace or log record. Initially all *message_ID*s are enabled for formatting. To format records with specific *message_ID*s, turn off all message IDs using the **!*** operator, then selectively enable the desired message IDs. Only one *message_ID* is allowed on each line. Multiple lines are "OR"ed together.

STREAMS Filtering

The STREAMS subsystem filter allows filtering on some fields of the messages logged by STREAMS modules and drivers. See *strlog*(7) for more information.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported in data. Single-byte character codesets are supported in filenames.

DEPENDENCIES

netfmt only recognizes subsystems and filters from products which have been installed and configured.

WARNINGS

The syntax that was used for the obsolete LAN trace and log options has been mixed with the syntax for the **netfmt** command such that any old options files can be used without any changes. The combination of syntax introduces some redundancy and possible confusion. The global filtering options have the string **formatter filter** as the first two fields, while the LAN filtering options merely have the string **filter** as the first field. It is expected that the older LAN filtering options may change to become more congruent with the global filtering syntax in future releases.

The **nettl** and **netfmt** commands read the **/etc/nettlgen.conf** file each time they are executed. These commands will not operate if the file becomes corrupted (see *nettl*(1M) and *netfmt*(1M)).

DIAGNOSTICS

Messages describe illegal use of **netfmt** command and unexpected EOF encountered.

EXAMPLES

The first group of examples show how to use command line options.

1. Format the last 50 records in file **/var/adm/nettl.LOG000** (the default log file):

```
netfmt -t 50 -f /var/adm/nettl.LOG000
```

2. Use the follow option to send *all* log messages to the console (normally, only **DISASTER**-class log messages are sent to the console in console form):

```
netfmt -f  /var/adm/nettl.LOG000  -F  > /dev/console
```

3. Monitor all log messages in a **hpterm** window:

```
hpterm -e /usr/sbin/netfmt -F -f /var/adm/nettl.LOG000
```

4. Read file **/var/adm/trace.TRC000** for binary data and use **conf.file** as the filter configuration file:

```
netfmt -c conf.file -f /var/adm/trace.TRC000
```

The remaining examples show how to specify entries in the filter configuration file used with the **-c** option.

1. Tell **netfmt** to format only **INFORMATIVE**-class log messages coming from the **NS_LS_IP** subsystem between 10:31:53 and 10:41:00 on 23 November 1993.

```
formatter       filter      time_from      10:31:53    11/23/93
formatter       filter      time_through   10:41:00    11/23/93
formatter       filter      class          !*
formatter       filter      class          INFORMATIVE
formatter       filter      subsystem      !*
formatter       filter      subsystem      NS_LS_IP
```

2. Map hardware address to name(LAN):

```
name            node1           08-00-09-00-0e-ca
name            node3           02-60-8c-01-33-58
```

3. Format only packets from either of the above hardware addresses:

```
filter          source          08-00-09-00-0e-ca
filter          source          02-60-8c-01-33-58
```

4. Format all packets transmitted from the local node, **local**, to the remote node, **192.6.1.3**, which reference local TCP service ports **login** or **shell**, or remote UDP port **777**:

```
filter          ip_saddr        local
filter          ip_daddr        192.6.1.3
filter          tcp_sport       login
filter          tcp_sport       shell
filter          udp_dport       777
```

5. Format a TCP connection from local node **node2** to **192.6.1.3** which uses **node2** service port **ftp** and remote port **1198**.

```
filter          connection      node2:ftp     192.6.1.3:1198
```

6. Format all packets except those that use interface **lan0**:

```
filter          interface       ! lan0
```

7. Format all logged events for subsystem **ip**. No other events are formatted. (By default, all events are formatted):

```
filter          subsystem       ip    event       *
```

8. Format only event **5003** for subsystem **ip**. Format all events except **3000** for subsystem **tcp**. No other events are formatted.

```
filter          subsystem       ip     event  5003
filter          subsystem       tcp    event  *,!3000
```

9. Format only events **5003**, **5004**, **5005**, and **5006** for subsystem **ip**. Format all events except events **3000**, **3002**, and **3003** for subsystem **tcp**. No other events are formatted:

```
filter       subsystem       ip     event  5003-5006
filter       subsystem       tcp    event  *,!3000,!3002-3003
```

10. Format only those records containing message IDs **9973** and **9974** for subsystem **session** and those not containing message ID **9974** for subsystem **transport**. All records from other subsystems are formatted:

```
ots session     msgid           !*
ots session     msgid           9973
ots session     msgid           9974
ots transport   msgid           !9974
```

11. Combine LAN and general filtering options into one configuration file. Format 15 minutes of pduin and pduout data starting at 3:00 PM on 2 April 1990 for data from **lan0** interface.

```
formatter       filter        kind           0x30000000
formatter       filter        time_from      15:00:00  04/02/90
formatter       filter        time_through   15:15:00  04/02/90
filter          interface     !*
filter          interface     lan0
```

AUTHOR

netfmt was developed by HP.

FILES

/etc/nettlgen.conf	default subsystem configuration file
/var/adm/conslog.opts	default console logging options filter file
$HOME/.netfmtrc	default filter configuration file if the **-c config_file** option is not used on the command line.

SEE ALSO

nettl(1M), kl(1M), nettlconf(1M), nettlgen.conf(4), strlog(7).

NAME

NDP - Neighbour Discovery Protocol

DESCRIPTION

NDP is a protocol used by hosts and routers to:

1. Find the link-layer address of the neighbours known to be attached to the same link.
2. Find the neighbouring routers that are willing to forward packets on their behalf.
3. Actively keep track of which neighbours are reachable and which are not.
4. Search for alternate routers when the path to a router fails.

NDP, to accomplish the above mentioned tasks defines the following processes:

1. Router and Prefix discovery.

Router discovery is a process through which hosts locate the neighbouring routers and learn prefix plus other parameters necessary for address autoconfiguration.

Prefix discovery is used by the hosts to learn the range of IPv6 addresses that reside on-link and can be reached without going through a router.

Routers send Router Advertisements which will make the hosts treat them as the default routers. The Router Advertisements will also contain prefix information options that will identify the range of IPv6 addresses that are on-link (Subnet prefix).

2. Router and Host requirements.

Router requirements in NDP specify a set of rules for host to act as a router. These rules include

- Router configuration variables.

These configuration variables include intervals between successive unsolicited router advertisements etc.

- How to make an interface, an advertising interface.

When an interface is made an advertising interface, it means that the node is going to send periodic router advertisements and is willing to forward packets on behalf of hosts on that link.

- Message content for router advertisements.

A router will send periodic as well as solicited Router Advertisements on an advertising interface. **NDP** specifies the format of these messages.

- Sending unsolicited router advertisements.

Apart from sending solicited router advertisements in response to router solicitations, routers can send unsolicited router advertisements. For example, unsolicited router advertisements can be sent to expire a prefix or to advertise a new prefix etc.

- Stopping router advertisements on an interface.

A router can stop advertising prefixes on an interface. This can happen due to system management decisions when a router may be stopped from being one. **NDP** specifies what the router should be doing under these circumstances.

- Processing router solicitation messages.

Hosts as part of the stateless autoconfiguration process will send Router Solicitations. Routers should respond to such solicitations with a router advertisement.

- Steps to be taken when the link-local address for the router changes.

Normally the link-local address of a Router should not change. However NDP still defines the steps should be taken by the router when its link-local address changes for any of its interfaces.

Host requirements are a set of rules that apply for a IPv6 host. They are,

- IPv6 variables that have to be maintained.

These variables include the time between retransmissions of neighbour solicitations, link MTU for each interface etc.

- Processing router advertisements.

This rule discusses what actions should be taken on receipt of router advertisements.

- Timing out prefixes and default routers.

Whenever routers send router advertisements they include the lifetime of the router as well as the prefixes that they advertise. **NDP** specifies what actions the host should take when these lifetimes expire.

- Selecting a default router

When there are more than one router in the link, the default router selection algorithm comes into picture. This algorithm will help selecting the default router based on factors like reachability etc.

- Sending a router solicitation.

When an interface is enabled a host need not wait for the unsolicited router advertisement. Instead, it can send a router solicitation and get a router advertisement as a response. This will help in receiving the default router and prefix information as soon as the interface is enabled.

3. Algorithm for sending a packet

Any IPv6 host is required to maintain some data structures that will be used by the algorithm for sending a packet. These data structures are:

Neighbour Cache

A set of entries that will maintain IPv6 Address to link-layer address mappings for neighbours to which a packet has been sent recently. In addition to that it maintains information needed for neighbour unreachability detection like the reachability state etc.

Destination Cache

A set of entries for hosts to whom packets have been sent recently. This includes hosts which are both on-link and off-link. It contains a level of indirection to the neighbour cache.

Prefix List

This is a list of prefixes which define the set of IPv6 address that are on-link. This information is maintained on a per interface basis. Typically this list is built from Router Advertisements received from the router.

Default Router List

A list of routers which will forward packets on behalf of this host. This list will again have a pointer to a neighbour cache entry for the respective router.

A host will use the above data structures while sending a packet to a host. Following is the conceptual algorithm for sending a packet to a unicast destination.

a. Before a packet is sent out the next hop should be determined. Normally next hop determination is not done on all packets. The results of a next hop determination are stored in the destination cache. The host should first check the destination cache for any entry that matches with the current destination address. If it finds a match then proceeds to step 'c'.

b. If there is no entry for the destination in the destination cache, a longest prefix match is made with all prefixes in the prefix list. If there is a match, the destination is determined to be on-link and the destination address will be considered as the next hop. Otherwise, the next hop is determined from the routing table.

c. Once the next hop is determined the address resolution process and neighbour unreachability detection are done for the next hop. This process is explained in the next section.

d. Once the neighbour is known to be reachable, the packet is sent to that destination.

4. Address Resolution And Neighbour Unreachability

Address resolution is a process used to determine the link-layer address of a neighbour. The IPv6 Address to link-layer address mapping found through this process is cached in the

Neighbour Cache. Following are the steps involved in Address Resolution.

a. First the neighbour cache is checked for an entry which matches the current destination address. If the entry is not present, the host sends a Neighbour Solicitation Message to the solicited-node multicast group. This multicast address is derived based on the destination IPv6 address and all nodes with the particular IPv6 address are required to join that group.

b. If a host with the specified IPv6 address is present in the network, it will reply this solicitation with a Neighbour Advertisement Message.

c. On receiving the Neighbour Advertisement the node will search for an entry in the neighbour cache for the sender's IPv6 address. A new entry is created in the neighbour cache and the reachability flag is set to REACHABLE.

Once the Address resolution is completed, neighbour unreachability detection will be performed. This process depends on the reachability field of the neighbour cache. An entry in the neighbour cache can have any of the following states:

a. INCOMPLETE. It means that the address resolution is in progress and the link-layer address of the destination is yet to be determined.

b. REACHABLE. It means that the destination is reachable until recently.

c. STALE. The destination is no longer known to be reachable, but reachability detection need not be made until a packet has to be sent to that destination.

d. DELAY. This state is an optimization that gives additional time for the upper layer protocols to provide the reachability confirmation.

e. PROBE. This state means that a reachability confirmation is actively requested by repeatedly sending Neighbour Solicitations.

During neighbour unreachability detection, the node checks for the state in the neighbour cache. If the state for the destination is REACHABLE, the packet is sent. Otherwise, the following are steps involved:

a. When an address resolution is made on a destination, an entry is created in the neighbour cache for that destination and the reachability state will be set to INCOMPLETE. If the address resolution fails, the entry is deleted.

b. When the address resolution passes, the entry will be filled with the destination's link-layer address and the state will be set to REACHABLE.

c. There is a timer maintained called the Reachability timer which will expire the state of an entry in the neighbour cache. Once this timer expires the reachability state changes from REACHABLE to STALE.

d. When a packet is being sent to a destination whose state is STALE in the neighbour cache, the node sets the state to DELAY and starts a timer associated with that state. By the time the timer expires if the node received reachability confirmation, the state is set to REACHABLE. Otherwise it is set to PROBE.

e. Once the entry's state is in PROBE, the node sends unicast neighbour solicitations to the link-layer address specified in the entry. If it receives a neighbour advertisement in response the state is set to REACHABLE. This solicitation will be sent repeatedly and the maximum number of times is configurable. If the reachability confirmation is not received after maximum solicitations, the entry is deleted from the neighbour cache and the address resolution is done again.

NOTE: Entries in the neighbour cache can also be created as a result of node receiving unsolicited Neighbour Advertisements, Router Advertisements and Router Solicitations etc. However, for the entry created under these circumstances the reachability state will always be set to STALE.

5. Redirect function

A router will send a host a redirect message when it finds that there is a better next-hop router on the same link. This is a requirement for a router.

Hosts on receiving a router redirect message, should update its destination cache with the new next hop address.

AUTHOR

NDP was developed by the IPng Working Group of Internet Engineering Task Force.

SEE ALSO

ifconfig(1M), lanconfig(1M), ndp(1M), ip6(7), lan(7).

Neighbour Discovery for IPv6, RFC2461, T. Narten et al. NDP Neighbour Discovery Protocol

NAME

IPv6, ipv6, ip6 - Internet Protocol Version 6

SYNOPSIS

```
#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET6, SOCK_DGRAM, 0);
s = socket(AF_INET6, SOCK_STREAM, 0);
```

DESCRIPTION

IPv6 is the next generation network-layer protocol designed to be the successor to the current Internet Protocol version 4 (IPv4). It provides the packet delivery service for TCP, UDP and ICMPv6.

IPv6 has significant advantages over IPv4 in terms of increased address space, simplified header format, integrated QoS support and mandatory security. IPv6 also allows optional internet-layer information to be encoded in separate headers called extension headers which are placed between the IPv6 header and upper layer headers. Extension headers currently supported are hop-by-hop option header, destination option header, fragment header and routing (type 0) header. An IPv6 packet may carry zero, one, or more extension headers, each identified by the next header field of the preceding header.

IPv6 has extended the address size from 32 bits to 128 bits and they are textually represented in hex-colon notation as **`x:x:x:x:x:x:x:x`**, where the **`x`**'s are the hexadecimal values of the eight 16-bit pieces of the address. For example **`fedc:83ff:fef6:417a:210:83ff:fef6:3dc0`**.

IPv6 has three types of addresses: **unicast**, **anycast**, and **multicast**.

- An **unicast address** is an identifier for a single interface. A packet sent to an unicast address is delivered to the interface identified by that address.
- An **anycast address** is an identifier for a set of interfaces. A packet sent to an anycast address is delivered to one of the interfaces identified by that address.
- A **multicast address** is an identifier for a set of interfaces. A packet sent to a multicast address is delivered to all interfaces identified by that address.

  There are no broadcast addresses in IPv6, their function is superseded by multicast addresses.

Every IPv6 address has a **scope** associated with it. A scope is a topological span within which the address may be used as an unique identifier for an interface or set of interfaces.

An unicast address has three defined scopes: link-local, site-local and global.

- Link-local address uniquely identifies interfaces within a single link and it has a fixed prefix of **`fe80::/10`**. For example, **`fe80::210:84c0:ef6f:cd30`**.
- Site-local address uniquely identifies interfaces within a single site only and it has a fixed prefix of **`fec0::/10`**. For example, **`fec0::210:84c0:ef6f:cd30`**.
- Global address uniquely identifies interfaces anywhere in the internet.

There are 2 special unicast addresses which hold an embedded IPv4 address in the low order 32-bits.

- The first type is termed as IPv4-compatible IPv6 address and is of the form **`0:0:0:0:0:0:d.d.d.d`**. This type of address is used by dual stack (IPv4/IPv6) nodes to perform automatic IPv6-over-IPv4 tunneling where the IPv4 tunnel endpoint address is determined from the IPv4 address embedded in the IPv4-compatible destination address of the IPv6 packet being tunneled.
- The second type is termed as IPv4-mapped IPv6 address and is of the form **`0:0:0:0:0:ffff:d.d.d.d`**. This address facilitates IPv6 applications to interoperate with IPv4 applications. Applications can automatically generate this address using **`getaddrinfo()`** (see *getaddrinfo*(3N)) when the specified host has only IPv4 address.

IPv6 Socket Options

New socket options are defined for IPv6 to send and receive extension headers and to exchange other optional information between the kernel and application. The options are supported at the **`IPPROTO_IPV6`** protocol level. The type of the variable pointed to by the **`optval`** parameter is indicated in parenthesis.

IPV6_UNICAST_HOPS (**integer**) Set or get the hop limit used in outgoing unicast packets. When this option is set using **setsockopt()** (see *setsockopt*(2)), the new option value specified is used as the hop limit for all subsequent unicast packets sent via that socket. Valid values are in the range 0-255 (both inclusive) and the default value is 64. For example,

```
int hoplimit = 50;
setsockopt(s, IPPROTO_IPV6, IPV6_UNICAST_HOPS,
        &hoplimit, sizeof(hoplimit));
```

This option can be used with **getsockopt()** (see *getsockopt*(2)) to determine the hop limit value the system will use for subsequent unicast packets sent via that socket.

IPV6_MULTICAST_HOPS
(**integer**) Set or get the hop limit used in outgoing multicast packets. When this option is set, the new option value specified is used as the hop limit for all subsequent multicast packets sent via that socket. Valid values are in the range 0-255 (both inclusive) and the default value is 1.

IPV6_MULTICAST_IF (**integer**) Sets the interface to use for outgoing multicast packets. The option value is the index of the selected outgoing interface. For example,

```
unsigned int index;
index = if_nametoindex("lan0");
setsockopt(s, IPPROTO_IPV6, IPV6_MULTICAST_IF,
        &index, sizeof(index));
```

IPV6_MULTICAST_LOOP
(**boolean**) Enables or disables loopback in the IP layer for multicast datagrams sent through this socket. The value of the variable pointed to by optval is zero (disable) or non-zero (enable). Default: enabled.

IPV6_JOIN_GROUP (**struct ipv6_mreq**) Join a multicast group on a specified local interface. The IPv6 multicast address of the group to join and the index of the interface on which to join should be specified using **struct ipv6_mreq** which is defined in <**netinet/in6.h**> as:

```
struct ipv6_mreq {
   struct in6_addr ipv6mr_multiaddr;
                   /* IPv6 multicast addr */
   unsigned int  ipv6mr_interface;
                   /* interface index */
};
```

If the interface index is specified as 0 then the default multicast interface is used.

IPV6_LEAVE_GROUP (**struct ipv6_mreq**) Leave a multicast group on a specified local interface. The IPv6 multicast address of the group to leave and the interface index should be specified using **struct ipv6_mreq**. The interface index should match the index used while joining the group. Set index to 0, to specify default interface.

IPV6_CHECKSUM (**integer**) When this option is set, kernel computes the checksum for outbound packets and verifies checksum on inbound packets. The option value is the byte offset of the checksum location in the user data. This option is not valid for **IPPROTO_ICMPV6** since checksum computation is mandatory for **IPPROTO_ICMPV6**. The default value is -1 (checksums not computed nor verified for protocols other than **IPPROTO_ICMPV6**).

IPV6_RECVPKTINFO (**boolean**) When this option is enabled, **PKTINFO** (destination IPv6 address and the arriving interface index) is returned as ancillary data by **recvmsg()**. (See *recvmsg*(2)). The information is returned in **struct**

in6_pktinfo structure and it is defined in <**netinet/in6.h**>**as:**

```
struct in6_pktinfo {
      struct in6_addr ipi6_addr;
      uint32_t        ipi6_ifindex;
};
```

By default this option is disabled.

IPV6_RECVHOPLIMIT (**boolean**) When this option is enabled, inbound packet's hoplimit is returned as ancillary data by **recvmsg()**. For example,

```
int on = 1;
setsockopt(s, IPPROTO_IPV6, IPV6_RECVHOPLIMIT,
              &on, sizeof(on));
```

By default this option is disabled.

IPV6_RECVDSTOPTS (**boolean**) When this option is enabled, the inbound packet's destination options (when present) is returned as ancillary data by **recvmsg()**. By default this option is disabled.

IPV6_RECVHOPOPTS (**boolean**) When this option is enabled, the inbound packet's hop-by-hop options (when present) is returned as ancillary data by **recvmsg()**. By default this option is disabled.

IPV6_RECVRTHDR (**integer; boolean**) When this option is enabled, the inbound packet's routing options (when present) is returned as ancillary data by **recvmsg()**. By default this option is disabled.

IPV6_RECVRTHDRDSTOPTS
(**integer; boolean**) When this option is enabled, the inbound packet's destination options appearing before a routing header (when present) is returned as ancillary data by **recvmsg()**. By default this option is disabled.

The next seven socket options can be used with both **setsockopt()** and as option name in ancillary data to **sendmsg()**. (See *sendmsg*(2))

IPV6_PKTINFO (**struct in6_pktinfo**) Used to set the source address and interface index for outgoing packets.

IPV6_HOPLIMIT (**integer**) Used to set the hop limit for outbound packets. This hop limit is valid for only a single output operation. To set hop limit for all unicast or multicast IPv6 packets use **IPV6_UNICAST_HOPS** or **IPV6_MULTICAST_HOPS** options respectively.

IPV6_NEXTHOP (**struct sockaddr_in6**) Used to set the next hop address. The node identified by this address must be a neighbor of the sending host. When this address is the same as the destination IPv6 address then this is equivalent to **SO_DONTROUTE** socket option.

IPV6_RTHDR (**variable length**) Used to specify the routing header for outgoing packets. Only Type 0 routing header is currently supported.

IPV6_DSTOPTS (**variable length**) Used to specify one or more destination options to be sent in subsequent IPv6 packets.

IPV6_HOPOPTS (**variable length**) Used to specify one or more hop-by-hop options to be sent in subsequent IPv6 packets.

IPV6_RTHDRDSTOPTS (**variable length**) Used to specify one or more destination options preceding a routing header. This option will be silently ignored when sending packets unless a routing header is also specified.

IPv6 uses the enhanced version of ICMP called ICMPv6 to report errors encountered in processing packets and for diagnostic purposes (like ping). ICMPv6 is an integral part of IPv6 and has a next header value of 58.

All the options and the associated structures are defined in <**netinet/in6.h**>, applications are not required to include this header file explicitly, it is automatically included by <**netinet/in.h**>.

DIAGNOSTICS

One of the following errors may be returned when a socket operation fails.

[EADDRINUSE]	The specified multicast group has been joined already.
[EADDRNOTAVAIL]	The specified IPv6 address is not a local interface address or there is no route for the specified multicast address or the specified multicast group has not been joined.
[EINVAL]	The parameter 'level' is not **IPPROTO_IPV6**, or **optval** is the NULL address, or the specified multicast address is not valid, or the specified hop limit is not in the range 0 <= x<= 255.
[ENOBUFS]	Insufficient memory is available for internal system data structures.
[ENOPROTOOPT]	The parameter optname is not a valid socket option for the **IPPROTO_IPV6** level.

AUTHOR

The socket interfaces to IP were developed by the University of California, Berkeley.

SEE ALSO

bind(2), getsockopt(2), recv(2), send(2), socket(2), inet(7F), ndp(7P).

RFC 2460 Internet Protocol Version 6.
RFC 2553 Basic Socket Interface Extensions for IPv6.
RFC 2292 Advanced Socket Interface Extensions for IPv6.

i