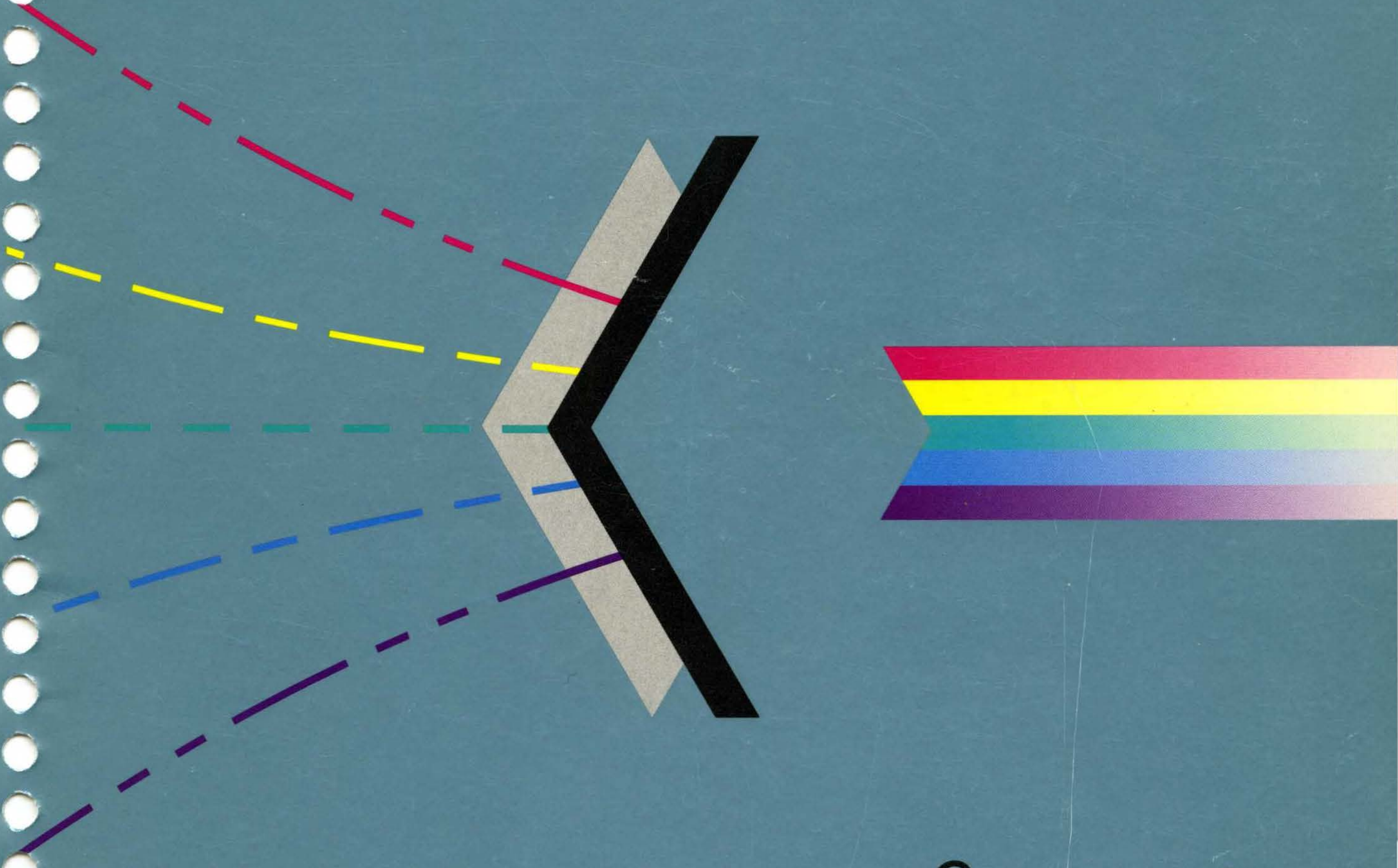
WELLFLEET communications



OPERATOR'S GUIDE

Wellfleet Operator's Guide

Wellfleet Communications, Inc.

Part # 103607A

Please address questions about technical matters to our 24-Hour Customer Support Line:

Inside Massachusetts:	617-275-2400
Outside Massachusetts:	1-800-2LANWAN

Please address comments about this manual to:

Technical Publications
Wellfleet Communications, Inc.
15 Crosby Drive
Bedford, MA 01730
Tel: (617) 275-2400
Fax: (617) 275-5001

Table of Contents

List of Figures

List of Tables

Preface

The material contained in this guide provides the information needed to monitor and control a system. Monitoring is enabled by a series of statistical screens that present a dynamic representation of on-going system activities, and by an event log that stores system-generated event messages. Control is enabled by the Network Control Language Interpreter (NCL), a command language that manages specific network entities and provides access to the system's management information base.

Audience

This guide is intended for experienced network operators and administrators who understand communications bridging and routing.

Organization

The *Operator's Guide* contains 13 chapters, a Table of Contents, List of Figures, List of Tables, and Index. The guide contains the following chapters:

Using the Statistics Screens

Tells you how to access Statistics Screens from the Main Menu and how to interpret the data presented by each available screen.

Using Network Control Language

Describes the functions of the Network Control Language (NCL) Interpreter and the format of NCL commands and explains how, generally, to execute them.

Managing Files

Tells you how to use the NCL file management commands, such as **`copy`**, **`del`**, **`dir`**, **`rename`**, and **`type`**.

Using TFTP

Tells you how to use TFTP to transfer executable (binary) and text files to and from the system.

Managing System Resources

Tells you how to control system resources using commands such as **`boot`**, **`dis`**, **`exit`**, **`help`**, **`log`**, and **`password`**.

Accessing Application-Specific Tables

Tells you how to use the NCL commands that work with SNMP and IP router software to provide access to application-specific routing and configuration tables.

Managing OSPF

Tells you how to use the OSPF NCL commands to obtain information about OSPF neighbors, link state database, routing table, interfaces, timer, queue, and error counts.

Using the Event Log

Describes the format of the system's event log and tells you how to interpret event log entries.

Event Messages, A to H

Defines each event log message, arranged by managed object, beginning with the letters **a** to **h**.

Event Messages, I to Z

Defines each event log message, arranged by managed object, beginning with the letters **i** to **z**.

Using the Management Information Base

Describes the general structure of the management information base (MIB) and the format of the MIB variables, and tells you how to access them.

Management Information Base Variables, A to H

Describes the structure of specific parts of the management information base and defines each variable and object, beginning with the letters **a** to **h**, stored in the information base.

Management Information Base Variables, I to Z

Describes the structure of specific parts of the management information base and defines each variable and object, beginning with the letters **i** to **z**, stored in the information base.

Related Documents

This section lists the Wellfleet document set.

Installation Guide

Explains how to install the system hardware and how to boot the system.

Configuration Guide

Explains how to create the *`config`* file and configure a network by describing the network to the system; includes information on configuring the network's current topology and future changes.

In addition to the usual document set, the following documents are available when the appropriate software is ordered:

SNMP-NMS User's Guide

Explains how to install SNMP application software in the network monitor.

NCU User Guide

Explains how to use NCU to configure your Wellfleet router.

Notation

Wellfleet documentation follows these standards for typography:

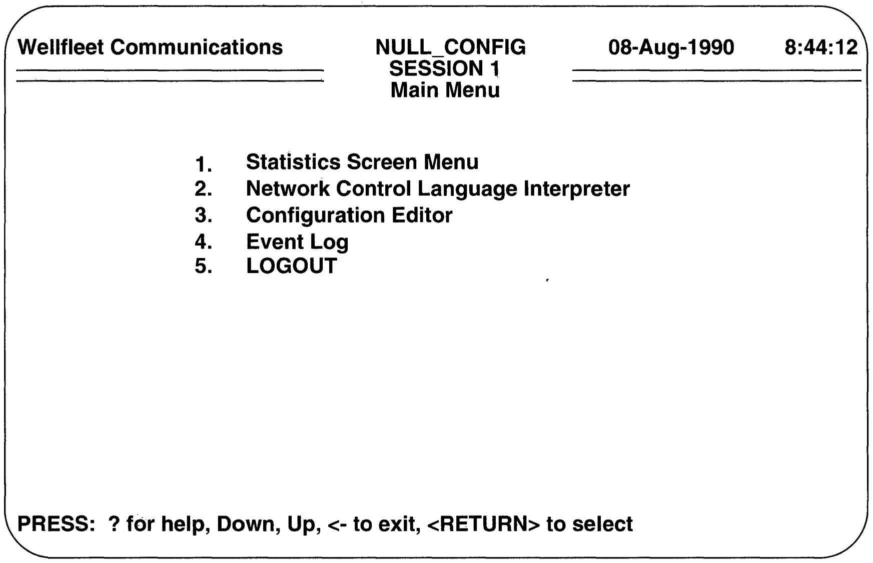
Type of Text	Components	Example
`user input`	Typewriter font bold in text	Use the **`dir`** command.
`user input`	Typewriter font regular in offset text	**sw->**`dir`
`Command names`	Typewriter font bold	Use the **`enable`** command.
`[KEYNAMES]`	Typewriter font bold in brackets []; (usually uppercase)	Press the **`[RETURN]`** key.
`Filenames`	Typewriter font oblique	Use the *`config`* file.
System output	Helvetica bold	**Zone name table full**
Command syntax:	Required arguments in angle brackets; optional arguments in curly braces divided by vertical bars	**`<addr>`** {...|...|...}
`[RETURN]` key symbol	Right angle arrow symbol	↵
Document titles	Italic	*Operator's Guide*
Chapter/section titles	Italic	Refer to the chapter entitled *Configuring the Bridge.*

1 Using the Statistics Screens

This chapter describes how to use some of the statistics recorded by the system during network operation. It tells you how to access the Statistics Screen Menu, how to display specified statistics screens on the console, and how to interpret statistical displays.

1.1 Accessing the Statistics Screen Menu

You begin displaying statistical data from the Main Menu (Figure 1-1).



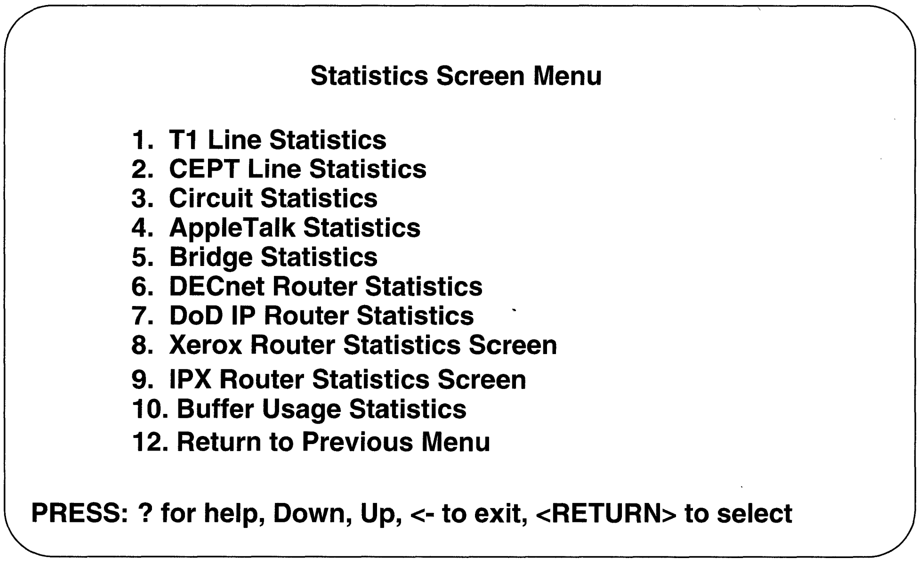
```
Wellfleet Communications        NULL_CONFIG        08-Aug-1990     8:44:12
                                 SESSION 1
                                 Main Menu

              1.   Statistics Screen Menu
              2.   Network Control Language Interpreter
              3.   Configuration Editor
              4.   Event Log
              5.   LOGOUT

PRESS:  ? for help, Down, Up, <- to exit, <RETURN> to select
```

Figure 1-1 Main Menu

Use the `[UPARROW]` (⇑) or `[DOWNARROW]` (⇓) key to position the cursor at **Statistics Screen Menu**, then press `[RETURN]` -- or, you may simply press the `<1>` key. After you press `[RETURN]` or type `<1>`, the console screen displays the Statistics Screen Menu on the console screen. Figure 1-2 shows a sample Statistics Screen Menu.



```
                    Statistics Screen Menu

      1. T1 Line Statistics
      2. CEPT Line Statistics
      3. Circuit Statistics
      4. AppleTalk Statistics
      5. Bridge Statistics
      6. DECnet Router Statistics
      7. DoD IP Router Statistics
      8. Xerox Router Statistics Screen
      9. IPX Router Statistics Screen
      10. Buffer Usage Statistics
      12. Return to Previous Menu

PRESS: ? for help, Down, Up, <- to exit, <RETURN> to select
```

Figure 1-2 Statistics Screen Menu

The sample in Figure 1-2 lists statistics screens that you can access, as follows:

T1 Line Statistics Screen provides summary data for each T1 line. It shows the number of alarms received and generated.

CEPT Line Statistics Screen provides summary data for each E1 (CEPT) line. It shows the number of alarms received and generated.

Circuit Statistics Screen provides summary data for each individual circuit. It shows the number of bytes and frames received and transmitted, and the number of received and transmitted frames that contained errors.

AppleTalk Statistics Screen provides summary data for each AppleTalk network interface.

Bridge Statistics Screen provides summary data for each Bridge circuit group. It shows the number of frames that were received, forwarded, flooded, and dropped.

DECnet Router Statistics Screen provides summary data for each DECnet router circuit group. It shows the number of frames received, forwarded, or dropped.

DoD IP Router Statistics Screen provides summary data for each IP Router network interface. It shows the number of IP datagrams received, forwarded, handled within the router, and dropped.

Xerox Router Statistics Screen provides summary data for each Xerox network interface. It shows the number of datagrams received, forwarded, handled within the router, and dropped.

IPX Router Statistics Screen provides summary data for each IPX network interface. It shows the number of datagrams received, forwarded, handled within the router, and dropped.

Buffer Usage Statistics Screen provides information on buffer allocation and use.

The actual Statistics Screen Menu displayed on your console reflects the system's line configuration and resident application modules. The Buffer Usage Statistics Screen is always available for display.

All statistics screens display cumulative information gathered since the system last booted. If you wish, you can reset (set to zero) values from individual statistics screens (refer to the section entitled *Resetting Statistics Screen Values* in this chapter), or you can use the Network Command Language (NCL) `RESET` command to reset values.

1.2 Displaying Help Text

You can display help text on the contents of any statistics screen from the Statistics Screen Menu. To display help text, use the `[UPARROW]` (⇑) or `[DOWNARROW]` (⇓) to position the cursor to the immediate left of the desired screen, and type `<?>`. The console screen then displays a brief summary of the screen's contents. After examining this summary, you press any key to return to the Statistics Screen Menu.

1.3 Displaying a Statistics Screen

You display any statistics screen from the Statistics Screen Menu. To display a screen, press the `[DOWNARROW]` (⇓) or `[UPARROW]` (⇑) to position the cursor to the immediate left of the desired screen and then press `[RETURN]` -- or, you may simply type the menu item number that appears to the left of the screen name. After you press `[RETURN]` or type a number, the console displays the specified screen.

1.3.1 Refreshing the Statistics Screens

All statistics screens are dynamic and are updated periodically. The update cycle is configurable and ranges from a minimum of one second to a maximum of one minute. The default cycle is three seconds. Cycle duration is determined by the **Screen Refresh Rate** parameter (refer to the *Configuration Guide* for details) which is set during system configuration.

1.3.2 Resetting Statistics Screen Values

Occasionally when examining a screen you may see a number prefixed with an asterisk (for example, ***234345677**). The asterisk indicates that the number is too large to be displayed, and that the system has truncated the number's most significant digits. At this point you may wish to reset the value.

When you reset a displayed parameter value, you set to zero that value as well as all other values displayed on the same horizontal line. (Parameter values displayed on the same horizontal line refer to the same circuit, circuit group, network interface, line, or slot.)

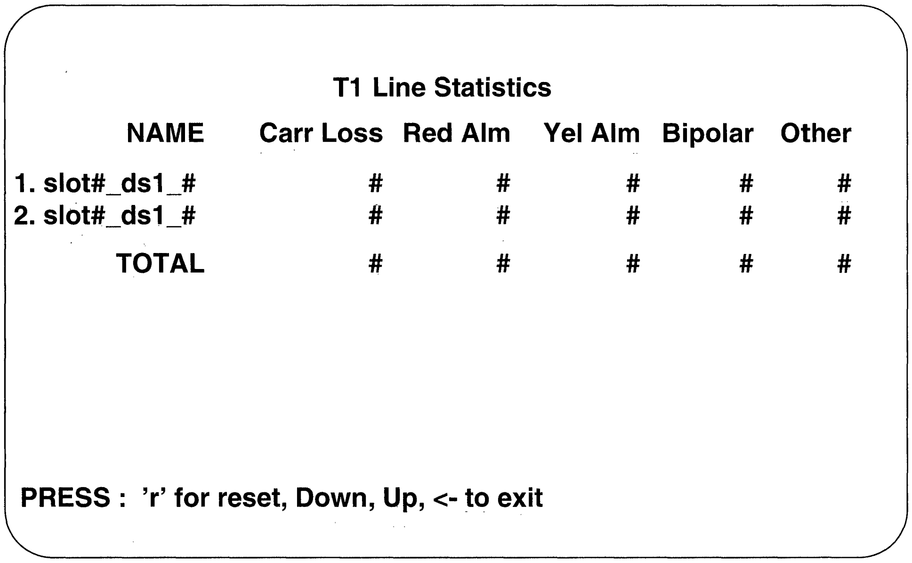
To reset a displayed value, use the `[DOWNARROW]` (⇓) or `[UPARROW]` (⇑) to position the cursor on the line containing the value to be reset; then type `<r>`, and press `[RETURN]`.

1.3.3 Leaving a Statistics Screen

To leave a statistics screen and return to the main Statistics Screen Menu, press the `[LEFTARROW]` (⇐) key. The system returns you to the Statistics Screen Menu.

1.4 T1 Line Statistics Screen

Availability of the T1 Line Statistics Screen (Figure 1-3) is configuration dependent. The system provides this screen only if you have established one or more T1 lines during configuration. This screen provides a line by line analysis of T1 alarm conditions. If you wish to examine T1 alarms at a greater level of detail than provided by the T1 Line Statistics Screen, you can use the NCL GET command to obtain a complete listing of T1 alarm statistics maintained by the system.



```
                         T1 Line Statistics
         NAME      Carr Loss  Red Alm   Yel Alm  Bipolar  Other

1. slot#_ds1_#            #         #         #        #      #
2. slot#_ds1_#            #         #         #        #      #
        TOTAL             #         #         #        #      #






PRESS :  'r' for reset, Down, Up, <- to exit
```

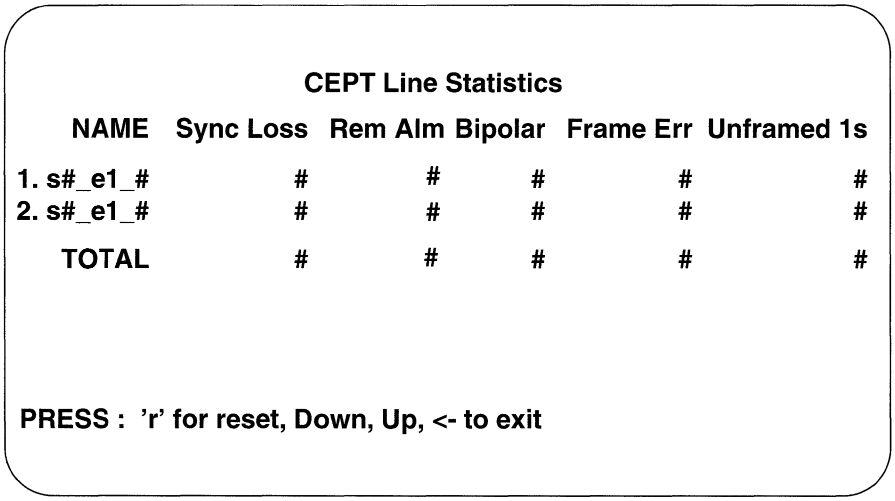
Figure 1-3 T1 Line Statistics Screen

Values provided by the T1 Line Statistics Screen are as follows:

Name	Lists each T1 line by slot number and connector designator.
Carr Loss	Contains the number of instances of loss of T1 carrier signal.
Red Alm	Contains the number of Red alarms (locally detected frame losses).
Yel Alm	Contains the number of Yellow alarms (remotely detected frame losses).
Bipolar	Contains the number of bipolar errors.
Other	Contains the number of other line errors.
TOTAL	Contains an aggregate count for each reporting metric.

1.5 CEPT Line Statistics Screen

Availability of the CEPT Line Statistics Screen (Figure 1-4) is configuration dependent. The system provides this screen only if you have established one or more E1 (CEPT) lines during configuration. This screen provides a line by line analysis of E1 alarm conditions. If you wish to examine E1 alarms at a greater level of detail than provided by the CEPT Line Statistics Screen, you can use the NCL **GET** command to obtain a complete listing of E1 alarm statistics maintained by the system.



```
                         CEPT Line Statistics
      NAME   Sync Loss  Rem Alm  Bipolar  Frame Err  Unframed 1s
1. s#_e1_#          #        #        #          #            #
2. s#_e1_#          #        #        #          #            #
     TOTAL          #        #        #          #            #



PRESS :  'r' for reset, Down, Up, <- to exit
```

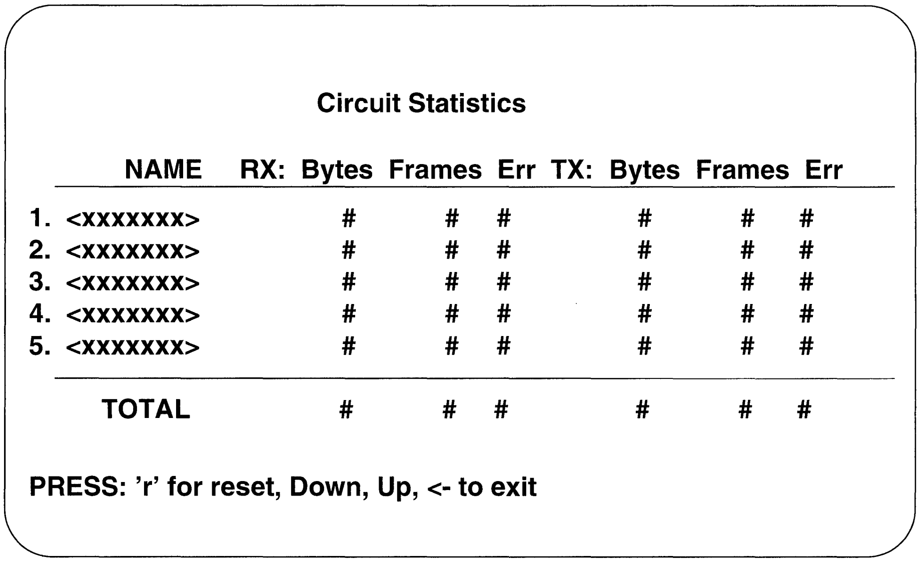
Figure 1-4   CEPT Line Statistics Screen

Values provided by the CEPT Line Statistics Screen are as follows:

Name	Lists each CEPT line by slot number and connector designator.
Sync Loss	Contains the number of instances of synchronization loss.
Rem Alm	Contains the number of instances of receiving an alarm indication from the end of the E1 line.
Bipolar	Contains the number of bipolar violations.
Frame Err	Contains the number of received E1 frames that contained errors.
Unframed 1s	Contains the number of instances of receiving an unframed, all 1s signal.
TOTAL	Contains an aggregate count for each reporting metric.

1.6 Circuit Statistics Screen

The Circuit Statistics Screen (Figure 1-5) provides summary traffic volume data for each circuit within the system. If you wish to examine circuit usage at a greater level of detail than provided by the Circuit Statistics Screen, you can use the NCL **GET** command to obtain a complete listing of circuit statistics maintained by the system.



```
                          Circuit Statistics

         NAME    RX: Bytes  Frames  Err  TX: Bytes  Frames  Err
      ----------------------------------------------------------
 1. <xxxxxxx>          #        #    #          #        #    #
 2. <xxxxxxx>          #        #    #          #        #    #
 3. <xxxxxxx>          #        #    #          #        #    #
 4. <xxxxxxx>          #        #    #          #        #    #
 5. <xxxxxxx>          #        #    #          #        #    #
      ----------------------------------------------------------
       TOTAL           #        #    #          #        #    #

 PRESS: 'r' for reset, Down, Up, <- to exit
```

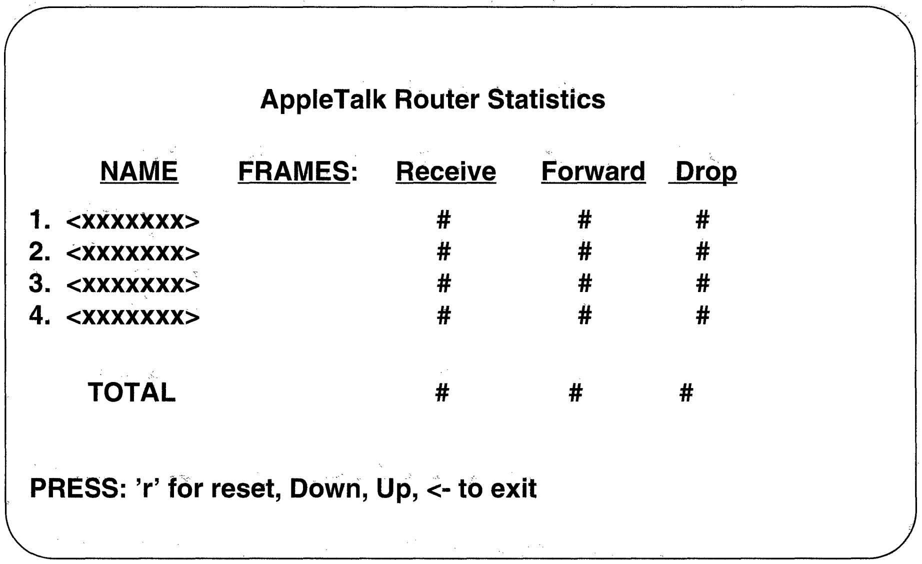
Figure 1-5 Circuit Statistics Screen

Values provided by the Circuit Statistics Screen are as follows:

NAME	Lists each individual circuit as contained in the configuration file.
RX: Bytes	Contains the number of bytes of data received by the circuit.
RX: Frames	Contains the number of frames received by the circuit.
RX: Err	Contains the number of faulty frames (frames that contained an error) received by the circuit.
TX: Bytes	Contains the number of bytes of data transmitted by the circuit.
TX: Frames	Contains the number of frames transmitted by the circuit.
TX: Err	Contains the number of frames that were not transmitted because of errors.
TOTAL	Contains an aggregate count for each reporting metric.

1.7 AppleTalk Statistics Screen

The AppleTalk Statistics Screen (Figure 1-6) provides a circuit group by circuit group summary analysis of AppleTalk Router operations. Availability of the screen is configuration dependent; the Wellfleet system provides this screen only if you have loaded the AppleTalk Router software during the configuration process.



```
                    AppleTalk Router Statistics

    NAME      FRAMES:   Receive    Forward   Drop

1. <xxxxxxx>               #          #        #
2. <xxxxxxx>               #          #        #
3. <xxxxxxx>               #          #        #
4. <xxxxxxx>               #          #        #

   TOTAL                   #         #        #


PRESS: 'r' for reset, Down, Up, <- to exit
```

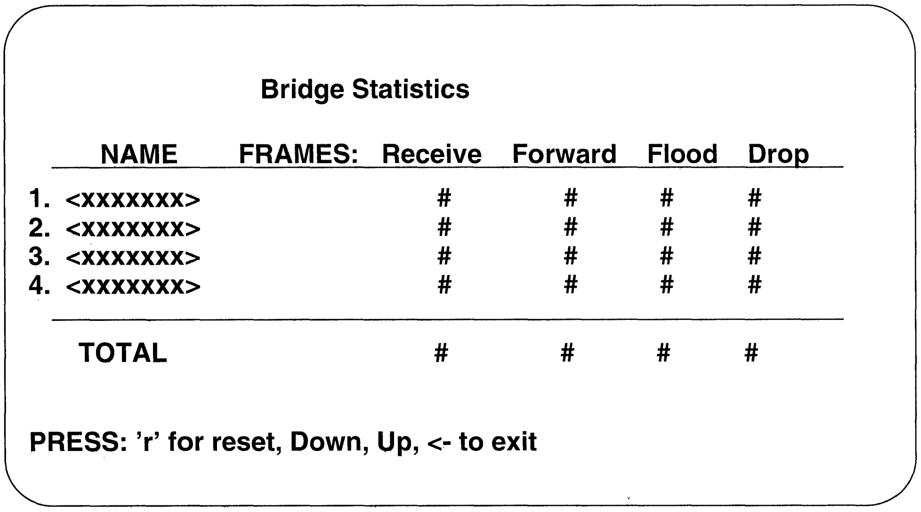
Figure 1-6 AppleTalk Statistics Screen

Values provided by the AppleTalk Statistics Screen are as follows:

NAME	Lists each individual AppleTalk circuit group
Receive	Contains the number of AppleTalk packets received across the circuit group.
Forward	Contains the number of received AppleTalk packets that were relayed by the AppleTalk Router to an adjacent system or to another router.
Drop	Contains the number of received AppleTalk packets dropped by the Appletalk Router for whatever reason.
TOTAL	Contains an aggregate count for each reporting metric.

1.8 Bridge Statistics Screen

Availability of the Bridge Statistics Screen is configuration dependent. The system provides this screen only if you have loaded the Bridge application software during configuration process. This screen provides a circuit group by circuit group analysis of Bridge operations. The Bridge Statistics Screen is shown in Figure 1-7.



```
              Bridge Statistics

   NAME      FRAMES:  Receive  Forward  Flood  Drop
1. <xxxxxxx>             #        #       #      #
2. <xxxxxxx>             #        #       #      #
3. <xxxxxxx>             #        #       #      #
4. <xxxxxxx>             #        #       #      #

  TOTAL                  #        #       #      #

PRESS: 'r' for reset, Down, Up, <- to exit
```

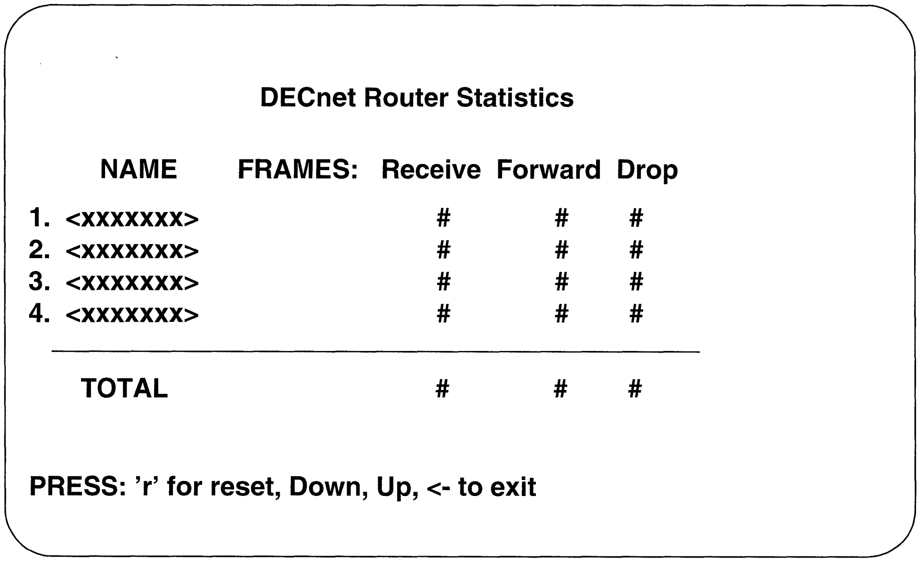
Figure 1-7 Bridge Statistics Screen

Values provided by the Bridge Statistics Screen are as follows:

NAME	Lists each individual circuit group.
Receive	Contains the number of frames received by the circuit group.
Forward	Contains the number of received frames that were forwarded by the Bridge. Forwarding requires that the Bridge "learned" the destination address.
Flood	Contains the number of received frames that were flooded by the Bridge. Flooding indicates: (1) that the Bridge had not "learned" the destination address at the time of packet reception, or (2) the packet contained a multicast address.
Drop	Contains the number of received frames that were dropped by the Bridge. Reasons for dropping packets could be that the packet is directed to a *Blocked* port, or because of protocol/source address filtering.
TOTAL	Contains an aggregate count for each reporting metric.

1.9 DECnet Phase IV Router Statistics Screen

Availability of the DECnet Phase IV Router Statistics Screen (Figure 1-8) is configuration dependent. The system provides this screen only if you have loaded the DECnet Phase IV Router application software during the configuration process. This screen provides a circuit group by circuit group analysis of DECnet Phase IV Router operations. If you wish to examine DECnet Phase IV Router statistics at a greater level of detail than provided by the DECnet Phase IV Router Statistics Screen, you can use the NCL **GET** command to obtain a complete listing of DECnet statistics maintained by the system.



```
                    DECnet Router Statistics

      NAME      FRAMES:  Receive  Forward  Drop

 1. <xxxxxxx>               #        #      #
 2. <xxxxxxx>               #        #      #
 3. <xxxxxxx>               #        #      #
 4. <xxxxxxx>               #        #      #
   ______________________________________________

    TOTAL                   #        #      #


 PRESS: 'r' for reset, Down, Up, <- to exit
```

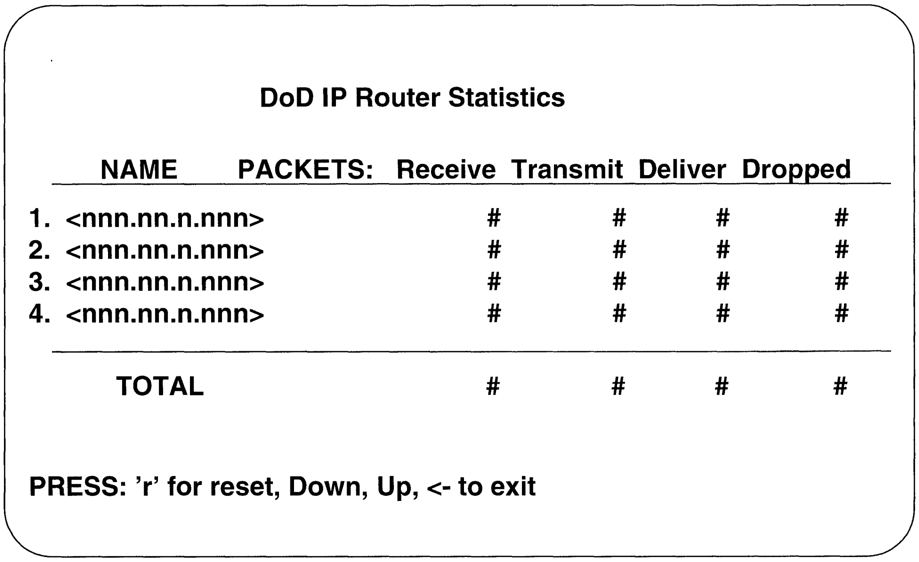
Figure 1-8 DECnet Router Statistics Screen

Values provided by the DECnet Phase IV Router Statistics Screen are as follows:

Name	Lists the name of the DECnet circuit group.
Receive	Contains the number of data frames received on the circuit group.
Forward	Contains the number of data frames transmitted on the circuit group.
Drop	Contains the number of data frames dropped by the circuit group.
TOTAL	Contains an aggregate count for each reporting metric.

1.10 DoD IP Router Statistics Screen

Availability of the DoD IP Router Statistics Screen (Figure 1-9) is configuration dependent. The system provides this screen only if you have loaded the IP Router application software during the configuration process. This screen provides summary traffic volume data for each IP network interface.



```
                    DoD IP Router Statistics

      NAME      PACKETS:  Receive  Transmit  Deliver  Dropped
   ______________________________________________________________
1.  <nnn.nn.n.nnn>           #         #         #        #
2.  <nnn.nn.n.nnn>           #         #         #        #
3.  <nnn.nn.n.nnn>           #         #         #        #
4.  <nnn.nn.n.nnn>           #         #         #        #
   ______________________________________________________________
       TOTAL                 #         #         #        #


PRESS: 'r' for reset, Down, Up, <- to exit
```

Figure 1-9 DoD IP Router Statistics Screen

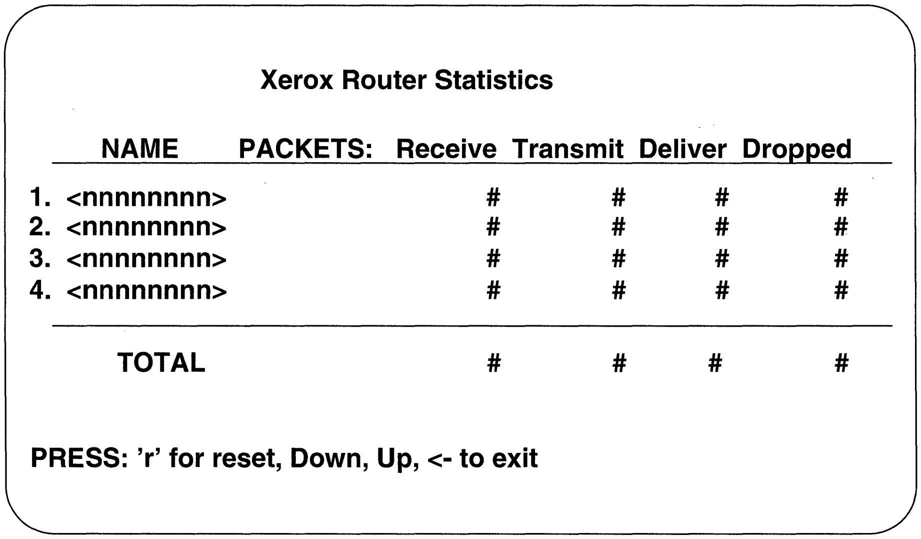
Values provided by the DoD IP Router Statistics Screen are as follows:

Name	Lists the network interface address in dotted decimal notation.
Receive	Contains the number of IP datagrams received by the network interface.
Transmit	Contains the number of IP datagrams transmitted by the network interface.
Deliver	Contains the number of IP datagrams addressed to the IP Router and delivered, by the router, to one of three upper layer protocols for processing. The three protocols are ICMP (Internet Control Message Protocol), TCP (Transmission Control Protocol), and UDP (User Datagram Protocol). If you desire more detailed information than provided by the DoD IP Router Statistics Screen, you can use the

	NCL GET command to obtain counts of received and transmitted ICMP datagrams by message type.
Dropped	Contains the number of IP datagrams dropped by the network interface. Dropped datagrams include (but are not limited to) datagrams with faulty checksums and datagrams requiring absent protocols. The interface also drops datagrams as directed by source address and destination address filters established during the configuration process. If you desire more detailed information than provided by the DoD IP Router Statistics Screen, you can use the NCL `GET` command to obtain a further breakdown of dropped messages.
TOTAL	Contains an aggregate count for each reporting metric.

1.11 Xerox Router Statistics Screen

Availability of the Xerox Router Statistics Screen (Figure 1-10) is configuration dependent. The system provides this screen only if you have loaded the Xerox Router application software during the configuration process. This screen provides summary traffic volume data for each Xerox network interface.



```
                  Xerox Router Statistics

    NAME      PACKETS:  Receive  Transmit  Deliver  Dropped
 ----------------------------------------------------------
 1. <nnnnnnnn>             #         #        #        #
 2. <nnnnnnnn>             #         #        #        #
 3. <nnnnnnnn>             #         #        #        #
 4. <nnnnnnnn>             #         #        #        #
 ----------------------------------------------------------
      TOTAL                #         #        #        #


 PRESS: 'r' for reset, Down, Up, <- to exit
```

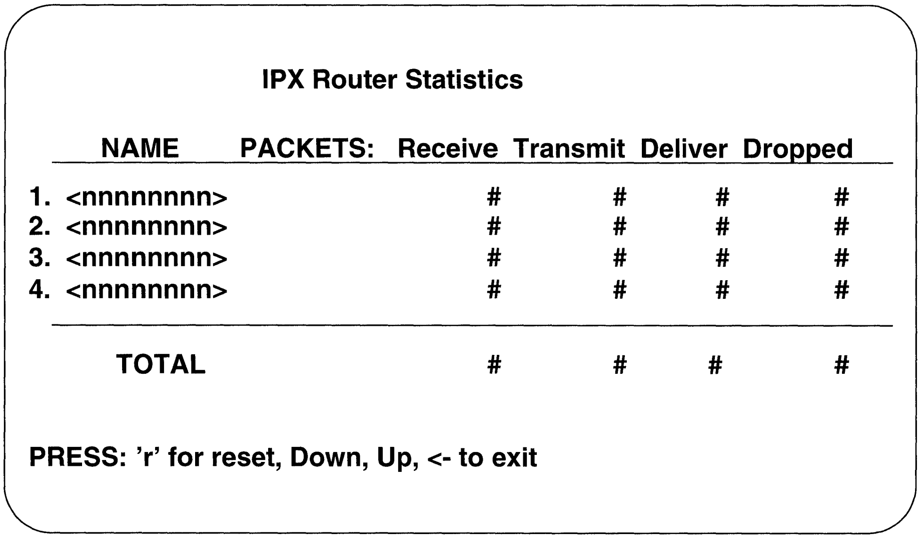
Figure 1-10 Xerox Router Statistics Screen

Values provided by the Xerox Router Statistics Screen are as follows:

Name	Lists the network interface address in 8-digit hexadecimal format.
Receive	Contains the number of XNS datagrams received by the network interface.
Transmit	Contains the number of XNS datagrams transmitted by the network interface.
Deliver	Contains the number of XNS datagrams delivered, by the router, to an upper layer protocol for processing.
Dropped	Contains the number of XNS datagrams dropped by the network interface. Dropped datagrams include (but are not limited to) datagrams with faulty checksums and datagrams with faulty header information.
TOTAL	Contains an aggregate count for each reporting metric.

1.12 IPX Router Statistics Screen

Availability of the IPX Router Statistics Screen (Figure 1-11) is configuration dependent. The system provides this screen only if you have loaded the IPX Router application software during the configuration process. This screen provides summary traffic volume data for each IPX network interface.



```
                    IPX Router Statistics

    NAME      PACKETS:  Receive  Transmit  Deliver  Dropped
  ---------------------------------------------------------
1. <nnnnnnnn>              #         #        #        #
2. <nnnnnnnn>              #         #        #        #
3. <nnnnnnnn>              #         #        #        #
4. <nnnnnnnn>              #         #        #        #
  ---------------------------------------------------------
     TOTAL                 #         #        #        #

PRESS: 'r' for reset, Down, Up, <- to exit
```

Figure 1-11 IPX Router Statistics Screen

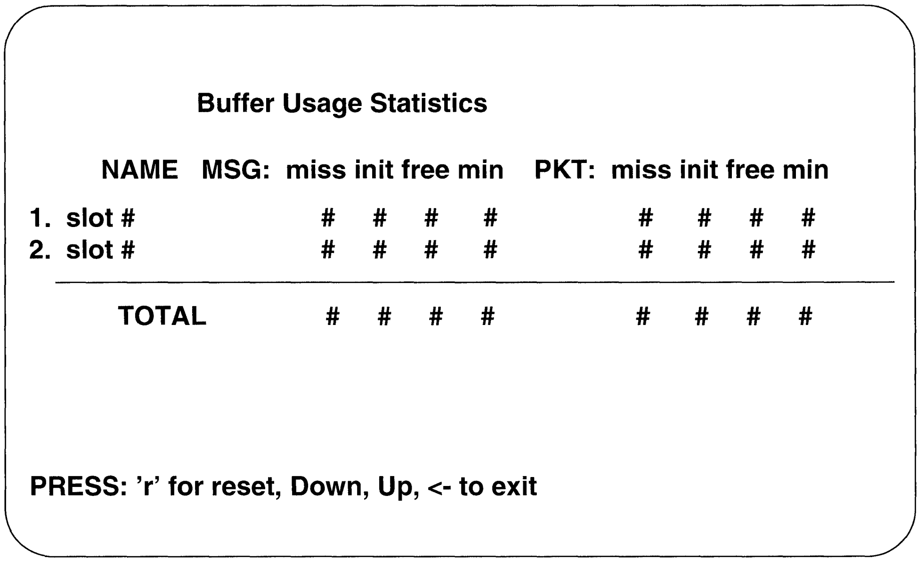
Values provided by the IPX Router Statistics Screen are as follows:

Name	Lists the network interface address in 8-digit hexadecimal format.
Receive	Contains the number of XNS datagrams received by the network interface.
Transmit	Contains the number of XNS datagrams transmitted by the network interface.
Deliver	Contains the number of XNS datagrams delivered, by the router, to an upper layer protocol for processing.
Dropped	Contains the number of XNS datagrams dropped by the network interface. Dropped datagrams include (but are not limited to) datagrams with faulty checksums and datagrams with faulty header information.
TOTAL	Contains an aggregate count for each reporting metric.

1.13 Buffer Usage Statistics Screen

The Buffer Usage Statistics Screen (Figure 1-12) is always available for inspection. This screen provides data on the allocation, usage, and availability of global memory buffers on each of the Advanced Communications Engine (ACE) processor boards within the system.

Global memory contains two types of buffers: message and packet. Message buffers facilitate internal, inter-processor communications that take place over the system's VME bus. Packet buffers facilitate external network communications by temporarily storing incoming or outgoing data packets.



```
          Buffer Usage Statistics

     NAME   MSG:  miss init free min   PKT:  miss init free min

1.  slot #              #    #    #    #           #    #    #    #
2.  slot #              #    #    #    #           #    #    #    #
 ----------------------------------------------------------------------
      TOTAL             #    #    #    #           #    #    #    #




PRESS: 'r' for reset, Down, Up, <- to exit
```

Figure 1-12 Buffer Usage Statistics Screen

Values provided by the Buffer Usage Statistics Screen are as follows:

NAME	Lists each slot, within the system cabinet, that contains an ACE processor board.
MSG: miss	Contains the number of times the system was unable to obtain a message buffer (that is, all buffers were in use).
MSG: init	Contains the number of message buffers allocated when the system booted.
MSG: free	Contains the number of message buffers available for use. Because system operations require some overhead, the number of buffers available is somewhat less than the number of allocated buffers.
MSG: min	Contains the lowest number of message buffers that were available since the system booted. Note that this count is directly related to the **MSG: miss** count. If message buffers were always available (**MSG: min** > 0), it follows that **MSG: miss** = 0.
PKT: miss	Contains the number of times the system was unable to obtain a packet buffer (that is, all buffers were in use).
PKT: init	Contains the number of packet buffers allocated when the system booted.

PKT: free	Contains the number of packet buffers available for use. Because system operations require some overhead, the number of packet buffers available is somewhat less than the number of allocated buffers.
PKT: min	Contains the lowest number of packet buffers that were available since the system booted. Note that this count is directly related to the **PKT: miss** count. If packet buffers were always available (**PKT: min** > 0), it follows that **PKT: miss** = 0.
TOTAL	Contains an aggregate count for each reporting metric.

2 Using Network Control Language

This chapter describes how to use the Network Control Language (NCL) Interpreter. NCL is a control command language that:

- ♦ Allows you to access the system's management information base in order to obtain detailed information about system operations.
- ♦ Allows you to access files and directories on the system diskette.
- ♦ Allows you to manage specific software entities (for example, application software such as the Bridge, or logical network connections such as circuits) within the system.
- ♦ Allows you to use the IP Router and Simple Network Management Protocol (SNMP) management-agent software to access either the standard Internet MIB (as defined in Internet Request for Comments 1156), the corporate (private) management information base of a local or remote network system, or application-specific bridging and routing tables from a local or remote system. (Refer to Chapters 11 to 13 for further information on MIB.)
- ♦ Allows you to use the Trivial File Transfer Protocol (TFTP) to transfer executable and text files between the system and a host computer system in the network.

You access NCL from the Main Menu. Use the `[UPARROW]` (⇑) or `[DOWNARROW]` (⇓) to position the cursor at **Network Control Language Interpreter**, then press `[RETURN]` -- or, you may simply press `<2>`. After you press `[RETURN]` or `<2>`, the console displays the NCL Command Screen (Figure 2-1).

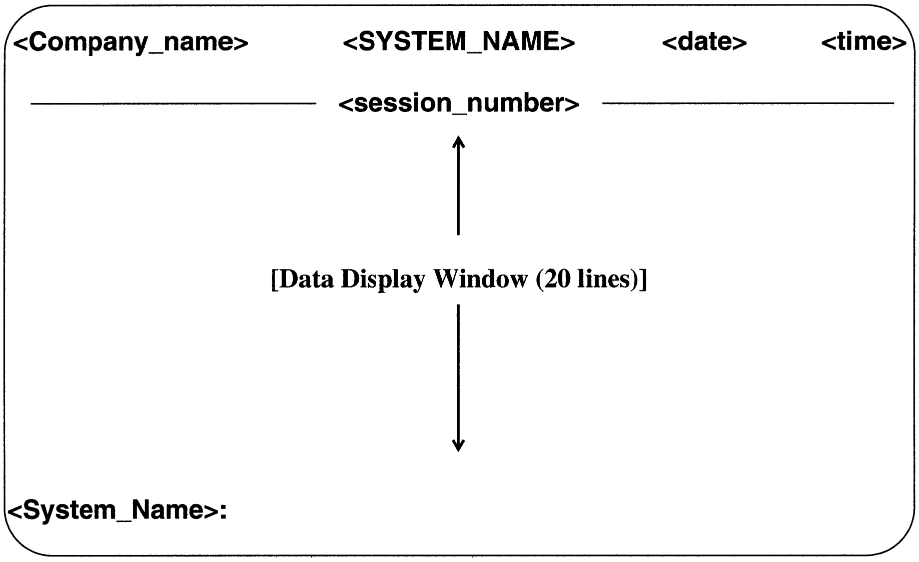


Figure 2-1 NCL Command Screen

A display banner at the top of the NCL Command Screen provides installation-specific data (company and system name) along with date, time, and session information. The NCL prompt, in the form of the system name followed by a colon, appears at the lower left of the screen. A 20-line area between the banner and the NCL prompt provides a window for data display.

2.1 Executing NCL Commands

You execute NCL commands from the NCL prompt. To execute a command:

1. **Enter the command name (or the legal abbreviation).**
2. **Supply required or optional argument values.**
3. **Press the `[RETURN]` key.**

In the NCL command examples that appear in following sections, two sets of delimiters distinguish between required and optional arguments, as follows:

```
ncl command <argument> {argument}
```

The left (<) and right (>) angle brackets delimit required arguments. The angle brackets are not entered as part of the command.

The left ({) and right (}) braces delimit optional arguments. Like angle brackets, braces are not entered as part of the command.

For example:

```
ping <ip_address> {count} {timeout}↵
```

where:

`ping`

Is the NCL **`PING`** command.

`ip_address`

Is the required ip_address of the target host or gateway.

`count`

Is the optional number of times to repeat the **`PING`** command.

`timeout`

Is the optional timeout (in seconds) for each ping.

↵

Is the carriage return required for command completion.

For purposes of clarity, all NCL command examples appear in bold face and typewriter font.

3 Managing Files

This chapter describes how to use NCL to manage files in the system. The system maintains three types of files: directory files, which contain lists of other files; executable (binary) files, which contain application and operational software; and text files, which contain data input to, or output by the system.

Table 3-1 lists file types. Table 3-2 lists all NCL file management commands:

Table 3-1 File Types

File Type	File Name	Function
directory	*HELP*	Contains a list of files whose contents provide the on-line help text.
	CRASH	Contains a list of "K" files.
	IPH.BIN	Contains firmware for FDDI VME driver.
executable	*ACE.OUT*	Contains executable application software.
	DMAP.OUT	Contains executable software.
	VRTX.OUT	Contains executable software.
	NETBOOT.OUT	Contains executable boot software.
text	*CONFIG*	A read/write file containing network-specific data.
	*K.**	A read-only file(s) containing the reason for system shutdown.
	*LOG.**	A read-only file(s) containing a sequential record of system-generated event messages.
	STAMP	A read-only file containing the software image stamp.
	README.TXT	A read-only file containing the software release notes.

Table 3-2 File Management Commands

Command	Function
`COPY`	Copy a text file to a backup diskette
`DEL`	Delete a file
`DIR`	Display contents of a directory file
`RENAME`	Rename a file
`TYPE`	Display contents of a specified text file

3.1 Displaying a Directory

You use the `DIR` (directory) command to display the contents of a directory file, as follows:

Syntax: **`dir {filename}↵`**

where:

`filename`

Is the optional name of a directory file. In the absence of an argument, **`dir`** defaults to the root directory.

Example: **`dir↵`**

Lists all files in the root directory.

`dir crash↵`

Lists all files in the directory file *`crash`*.

`DIR` provides a tabular listing of files in the root directory or in the directory file specified by **filename**. The listing provides file-specific information in the following format:

```
<filename> <ext> <size> <date> <time>
```

where:

`filename`

Is the name of a file in the root or specified directory.

`ext`

Is the file extension, when present.

`size`

If a numerical value, indicates the file size in bytes; if “**dir**”, indicates a directory file.

date `and` **time**

Are the data and time of file creation in month/date/year and hour/minute/second formats.

Figure 3-1 shows a sample root directory listing.

```
      ACE      OUT   689734   12-19-90   17:24:12
      VRTX     OUT   42       2-20-90    17:39:02
      DMAP     OUT   42       2-20-90    17:38:16
      IPH      BIN   131072   11-06-90   13:05:22
      NETBOOT  OUT   8332     12-19-90   15:16:14
      HELP           <DIR>    12-13-90   13:09:24
      CONFIG         1913     12-21-90   10:57:06
      CRASH          <DIR>    12-17-90   11.36.30
      LOG            1148     12-13-90   14:20:28
      NETBOOT OUT    8832     12-14-90   10:26:27

---- 'End of file' ----
```

Figure 3-1 Sample Root Directory Listing

The individual files listed in Figure 3-1 are as follows:

- ♦ *ACE.OUT, VRTX.OUT,* and *DMAP.OUT* are binary files that contain executable software.
- ♦ *IPH.BIN* contains firmware for the FDDI VME driver.
- ♦ *NETBOOT.OUT* is a file that performs some intermediate boot functions and must not be deleted from the diskette. If the file is lost or damaged, the router will still be able to boot from the *ACE.OUT* image on the diskette, but not from the network.

- *HELP* is a directory file. It contains of list of files whose contents provide the on-line help text.
- *CONFIG* is the *config* file.
- *CRASH* is a directory file containing a list of "**K**" files. Figure 3-2 shows a sample *CRASH* directory listing.

Whenever the system shuts down, it creates a file to record the cause of the event. Each of these files is assigned a sequential file name from *K0* to *K9*. As additional shutdowns occur, the contents of the oldest file are overwritten. The file *KPTR* contains a single numeric value that points to the oldest file. In Figure 3-2, *KPTR* contains the value 1, which points to file *K1* (the oldest *K* file, created at 16:08:12 on 7-26-89; it will be overwritten to record the next system shutdown).

```
.          <DIR>    8-00-80     0:00:00
..         <DIR>    8-00-80     0:00:00
KPTR           3    8-08-89    10:35:22
K0            21    8-08-89    10:35:22
K1            21    7-26-89    16:08:12
K2            21    7-26-89    16:13:22
K3            21    7-27-89    14:39:12
K4            21    8-01-89    12:58:18
K5            21    8-01-89    15:34:24
K6            21    8-02-89    15:35:24
K7            21    8-02-89    16:04:20
K8            21    8-04-89    10:18:16
K9            21    8-04-89    15:44:26


---- 'End of file' ----
```

Figure 3-2 Sample Crash Directory Listing

LOG.1 and *LOG.2* are log files that contain a sequential record of system event messages. The files *LOG.1* and *LOG.2* illustrate the log file numbering feature, which allows you to initiate a new event log with each system reboot. Refer to the *Configuration Guide* for information on enabling sequential event logs.

3.2 Displaying a Text File

You use the **TYPE** command to read a text file from the system diskette and display the file contents on the console screen, as follows:

Syntax: `type <filename>↵`

where:

`filename`

Is the required name of the text file to be displayed.

Example: `type config↵`

Displays the contents of *`config`* on the console screen.

`type readme.txt↵`

Displays the version-specific software release notes (contained in the file *README. TXT*).

`type crash/k9↵`

Displays a file named *K9* contained in a directory named *crash.*

The **TYPE** command displays all text files (log files, crash files, and those files created with the Configuration Editor) contained on the system diskette. Before displaying a file, you should verify that *page mode* is enabled, so that the file is displayed one screen at a time.

If a file exceeds 20 lines, the console displays **-- MORE --** below the last line of displayed data.

To view additional data:

1. **Press `[RETURN]` for one more line.**
2. **Type a number from 1 to 9 to display that number of additional lines.**
3. **Press any other key for an additional screen of data.**
4. **Press the `[LEFTARROW]` (⇐) to return to the NCL prompt.**

When the console screen reaches the last line of the file, it displays **- 'End of file'-** and positions the cursor at the NCL prompt.

3.3 Copying a Text File

You use the `COPY` command to copy a text file from the system diskette to a backup diskette, as follows:

Syntax: `copy <source> <destination>↲`

where:

`source`

Is the required name of the file to be copied.

`destination`

Is the required name of the file on the backup diskette.

Example: `copy config config.bu↲`

Copies the file *config* to a file named *config.bu* on the destination diskette.

Use the following procedure to make a copy of *config*, a log file, or any other non-executable file on the system diskette:

1. **At the NCL prompt, enter the `COPY` command, specify the source and destination files, and press `[RETURN]`.**
2. **When prompted to do so, remove the system diskette, insert the formatted target diskette, and press `[RETURN]`.**
3. **Wait for the system to complete the copy operation.**
4. **When prompted to do so, remove the target diskette, insert the system diskette, and press `[RETURN]`.**

The `COPY` command does not support copying binary files, nor does it support the use of wildcards in file names.

3.4 Deleting a File

You use the **DEL** (delete) command to delete a file from the system diskette, as follows:

Syntax: **del <filename>↵**

where:

filename

Is the required name of the file to be deleted.

Example: **del config.bu↵**

Deletes a file named *config.bu*.

del crash/k9↵

Deletes a file named *k9* in the directory *crash*.

The **DELETE** command does not prompt you to verify a deletion. Consequently, you must be certain that you do indeed wish to delete the specified file. Once you have deleted a file, you cannot recover it.

3.5 Renaming a File

You use the **RENAME** command to change the name of a file, as follows:

Syntax: **rename <orig_name> <new_name>↵**

where:

orig_name

Is the required name of an existing file. **<orig_name>** can contain a pathname to a file in a subdirectory.

new_name

Is the required name of the new file. **<new_name>** cannot include a directory path. Consequently, **RENAME** can change the name of a file, but it cannot alter the file's directory location.

Example: **rename ace.out oldace.out↵**

Renames the current software image *ace.out* to *oldace.out*.

rename config config.bu↵

Renames the current configuration file *config* to *config.bu*.

4 Using TFTP

This chapter describes how to use the Trivial File Transfer Protocol (TFTP) to transfer executable and text files between the system and a host computer system in the network. The system provides both a *server* and *client* application of the TFTP (as specified in Internet Request for Comments 783). The TFTP software resides within the IP Router. Consequently, you must load and enable the IP Router to use TFTP.

NOTE

The TFTP implementation does not support simultaneous file transfers. An attempt to initiate a second file transfer, while a file transfer is still in progress results in the rejection of the second transfer request.

4.1 Using the TFTP Client

The TFTP *client* capability allows transfer of files from one Wellfleet system to another Wellfleet system, or from a Wellfleet system to a host, as controlled by the Wellfleet system. This feature eliminates the need to use a host system to perform remote software distribution to other Wellfleet systems. The TFTP client capability is supplied by two NCL commands: **FPUT** and **FGET**.

The IP Router must be present on slot 2 and enabled, and TFTP must be enabled, to use these commands. Control is not returned to NCL until the file transfer is complete. Disk logging is suspended during a TFTP file transfer. Only one TFTP can be in progress at any given time on a single Wellfleet system.

4.1.1 TFTP FPUT

The **FPUT** command copies a file from the diskette of the local system to the file system of the remote target.

Syntax: **`FPUT <target_address><local_file_name>`**
`<remote_ file _name><mode>`

where:

`<target_address>`

Is the IP address of the remote target system.

`<local_ file_name>`

Is the name of the file to be transferred.

`<remote_file_name>`

Is the name of the transferred file on the target system; that is, the name under which the file is stored.

`<mode>`

Is a single character **`T`** for text (NETASCII) mode transfer or **`O`** for binary (OCTET) mode transfer.

4.1.2 TFTP FGET

The **FGET** command copies a file from the remote target's file system to the local diskette of the local system.

Syntax: **`FGET <target_address><local_file_name>`**
`<remote_file_name><mode>`

where:

`<target_address>`

Is the IP address of the remote target system.

`<local_ file_name>`

Is the name under which the retrieved file is stored.

`<remote_file_name>`

Is the name of the file which is retrieved.

`<mode>`

Is a single character **`T`** for text (NETASCII) mode transfer or **`O`** for binary (OCTET) mode transfer.

4.2 Using the TFTP Server to Initiate File Transfer

The Wellfleet system is the passive server during file transfer, which must be initiated by a client application residing on the network host. Executable files must be transferred in the OCTET (binary) mode; configuration files, log files, crash files, and other text files must be transferred in the NETASCII (text) mode.

The following is an example of a TFTP file transfer from a host to a system using a SUN-3 workstation running SUN OS 3.5 (refer to your host documentation for host-specific information on accessing and using the host *client* TFTP application).

```
$ tftp 192.32.1.62
tftp> binary
tftp> put ace.out  newace.out
Sent 521176 bytes in 55.0 seconds
tftp> ascii
tftp> put config.811  newcfg
Sent 10202 bytes in 10.8 seconds
tftp> quit
$
```

4.3 TFTP Security Features

The system provides some security measures to control access to and use of the TFTP facility.

Initial TFTP connection requests are made on the well known User Datagram Protocol (UDP) port 69. Access to TFTP on a specific network interface can be blocked, therefore, by constructing a TCP/UDP port filter to drop incoming datagrams destined for port 69.

In addition, TFTP does not auto-enable in the default state. While you can configure TFTP to auto enable, this option may not be desirous in environments where security is on concern. Use the following procedure to transfer files to or from a system on which TFTP is not auto-enabled.

1. **Telnet to the system.**
2. **Log in.**

3. **Access the Network Control Language Interpreter from the Main Menu.**
4. **Use the NCL `ENABLE` command to enable TFTP `(EN TFTP)`.**
5. **Transfer files as required.**
6. **Disable TFTP after completing file transfer `(DIS TFTP)`.**

Finally, to prevent the accidental or malicious destruction of key files, TFTP forbids the writing of files named *`ACE.OUT`* and *`CONFIG`* to the system diskette.

4.4 Performing Remote Software Distribution

The following steps describe how to perform remote software distribution using the Network Boot method.

NOTE

X.25 and FDDI circuits are not supported in remote software distribution using BOOTP.

4.4.1 Preparing the Source System as a BOOTP Server

Prior to actual software distribution, you must select one Wellfleet system to act as the source system. The source system acts as a BOOTP server and, upon request, supplies target systems (BOOTP clients) with software image files.

The source system must be running TCP/IP (with TFTP enabled) and be able to reach all target systems with TFTP enabled. Additionally, the source system must be configured as a BOOTP server with each target system identified as a BOOTP client (refer to the *Configuration Guide* for information on BOOTP server configuration).

After receiving a diskette that contains a new release of system software, verify the release version by checking the sticker on the diskette (or, in the absence of a sticker, by booting the system to see what version is on the diskette). Then, install the new software release on the source system by copying the source system's configuration file to the new diskette. After copying the file, boot the source system with the new diskette.

The size of the Wellfleet software image precludes a single stage file transfer between source and target systems. Consequently, the first stage file transfer passes a compressed software image from source to destination system. Because this compressed image does not contain X.25 and FDDI software, such circuits cannot support remote software distribution.

Remove the current diskette and insert the diskette that contains the compressed software image. Remote Software Distribution is a four stage process that consists of the following:

1. Transferring a compressed image file from source system to target system.
2. Transferring Netboot software from source system to target system.
3. Implementing a network boot using BOOTP client and server functionality.
4. Installing the new image on the target system.

4.4.2 Stage 1: Transferring the Compressed Image

During Stage 1, you use TFTP to transfer a compressed version of the software image file (*ACE.OUT*) from the source to the target system. This compressed image provides boot capability and enables the subsequent transfer of Netboot software. You then (depending on the type of circuits supported by the system) modify the target's configuration file and reboot the target system.

Prior to file transfer, use the NCL **dir** command to view the contents of the root directory file and note the size of *ACE.OUT.* Then use the following procedure to transfer *ACE.OUT* to the target system.

1. **Establish a TELNET session to the target system.**
2. **Log in to the target system and enable TFTP (if it is not already enabled).**
3. **Delete** *IPH.BIN* **from the target system, if** *IPH.BIN* **exists.**

   Deleting *IPH.BIN* frees up disk space to facilitate file transfer. *IPH.BIN* contains FDDI driver software. As the compressed image that will be transferred during Stage 1 does not support FDDI, the file is not needed. You will restore this file later in the distribution process.
4. **Use the NCL FGET command to transfer the load image file** *ACE.OUT* **from the source system to the target system in OCTET (binary) mode as follows:**

   FGET <source system IP address> *ACE.OUT NEWACE.OUT*

   Note that you save the image file under the name *NEWACE.OUT.*
5. **Use the NCL DIR command to ensure that the downloaded file (***NEWACE.OUT***) is the same size as** *ACE.OUT* **in the source system.**

   If the sizes are not the same, delete any unnecessary files from the diskette, then repeat Steps 4 and 5 (make certain to specify the OCTET (binary) mode).
6. **Delete any X.25 or FDDI circuits from the target's configuration file.**

If you do not delete such circuits, the system will not boot with the compressed *NEWACE.OUT* which contains no X.25 or FDDI software support.

7. **Save the configuration file as** *CONFIG.NEW.*
8. **Reboot the target system.**

   If your system controller in the target system is at revision level 12 or higher, you reboot by issuing the following command:

   BOOT NEWACE.OUT CONFIG.NEW

   If your system controller in the target system is at revision level 11 or lower, you reboot with the following series of commands:

   ```
   RENAME ACE.OUT OLDACE.OUT

   RENAME NEWACE.OUT ACE.OUT

   RENAME CONFIG CONFIG.OLD

   RENAME CONFIG.NEW CONFIG

   BOOT
   ```

If the download was successful, the target system should reboot and re-initialize properly. If the download was unsuccessful and the target system is unable to boot sufficiently to support TELNET access, repeat Steps 1 through 8 after having some on-site personnel perform the following recovery procedure:

Recovery procedure:

- ♦ If you rebooted the target system by issuing the **BOOT** *NEWACE.OUT* command, the person on-site only needs to press the red RESET button on the target system.
- ♦ If you rebooted the target system by renaming the *NEWACE.OUT* file, the person on-site needs to:

1. **Remove the diskette from the system**
2. **Rename the** *OLDACE.OUT* **file to** ACE.OUT **and rename** *CONFIG.OLD* **to** *CONFIG* **(using any MS-DOS compatible computer with a 1.2 MB drive).**
3. **Re-insert the diskette into the target system.**
4. **Press the red** RESET **button on the target system.**

4.4.3 Stage 2: Transferring the Netboot Software

During Stage 2, you use TFTP to transfer a network boot software (*NETBOOT.OUT*) from the source to the target system. The Netboot software enables the BOOTP services. You then use the Configuration Editor to auto enable TFTP and Network Boot. Depending upon your network topology, you may also configure the target system as a BOOTP server. After completing configuration changes, you reboot the target system.

Before starting this part of the procedure, remove the current diskette (which contains a compressed software image) and insert the system diskette that contains the full software image. Use the following procedure to transfer *NETBOOT.OUT* to the target system.

1. **Establish a TELNET session to the target system.**
2. **Use the NCL `FGET` command to transfer the load image file *NETBOOT.OUT* from the source system to the target system**

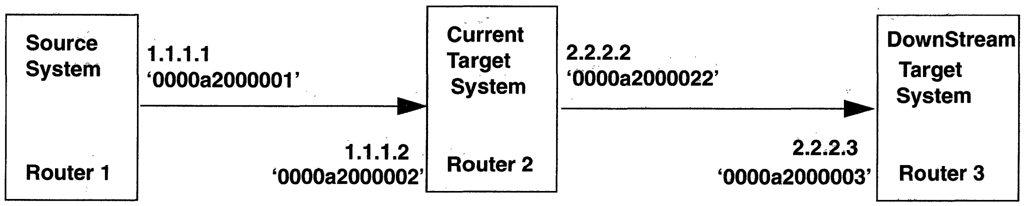
   Make certain that you specify the OCTET (binary) mode for this file.
3. **Use the Configuration Editor to modify *CONFIG.NEW* and auto-enable TFTP (refer to the *Configuration Guide*).**

NOTE

If the file *CONFIG.NEW* was renamed to *CONFIG* in step 8 of Stage 1, edit the file *CONFIG* instead of *CONFIG.NEW* in steps 3 through 6 of this section.

4. **Use the Configuration Editor to modify *CONFIG.NEW* and auto-enable Network Boot (refer to the *Configuration Guide*).**
5. **If the current target system is acting as a router between the source system and other target systems that will be upgraded in subsequent downloads, modify *CONFIG.NEW* and configure a BOOTP Server record on the current target system for each of those other target systems.**

   For example the BOOTP Server records for router 1 and 2 in the sample network (see Figure 4-1) must be configured as described in their respective tables.



	Router 1 BOOT Server Record		Router 2 BOOT Server Record
	Record 1	Record 2	Record 1
Client Hardware Address	0000a2000002	0000a2000003	0000a2000003
Client IP Address	1.1.1.2	2.2.2.3	2.2.2.3
Server IP Address	1.1.1.1	1.1.1.1	1.1.1.1
BOOTP Image File	ACE.OUT	ACE.OUT	ACE.OUT

Figure 4-1 Sample Network Configuration

6. **Save the modified configuration file as** *CONFIG.NEW.*

NOTE

If the file *CONFIG.NEW* was renamed to *CONFIG* in step 8 of Stage 1, save the file as *CONFIG.*

7. **Reboot the target system.**

   If your system controller in the target system is at revision level 12 or higher, you reboot by issuing the following command:

   ```
   BOOT NEWACE.OUT CONFIG.NEW
   ```

   If your system controller in the target system is at revision level 11 or lower, you reboot by issuing the following command:

   ```
   BOOT
   ```

4.4.4 Stage 3: Booting Across the Network

During Stage 3, which is automatic and essentially transparent to the operator, the system performs two boot operations. The first operator-initiated boot (refer to Step 7 in the previous section) causes the system's BOOTP client functionality to broadcast a request for an image file source across its attached non-FDDI and non-X.25 interfaces.

The BOOTP request is responded to by a neighboring BOOTP Server who replies with a BOOTP response designating an image source. After receiving the response, the system uses TFTP to obtain the designated image file. After receiving the image file, the system performs an automatic boot from the newly arrived software image.

If the manual boot proves unsuccessful, repeat the recovery procedure detailed in Stages 1 and 2.

4.4.5 Stage 4: Final Installation

During Stage 4, you write the new software image permanently to disk and retrieve a copy of a previously deleted file.

1. **Establish a TELNET session to the target system.**
2. **Rename** *NEWACE.OUT* **to** *ACE.OUT***, thus destroying the old** *ACE.OUT***.**

NOTE

You only need to delete files *OLDACE.OUT* and *CONFIG.OLD* if you renamed them from *ACE.OUT* and *CONFIG*, respectively, in step 8 of Stage 1.

3. **Rename** *CONFIG.NEW* **to** *CONFIG***, thus destroying the old** *CONFIG***.**
4. **Retrieve the** *IPH.BIN* **file from the source system and transfer it to the target system by issuing the `FGET` command.**

   Make certain that you specify the OCTET (binary) mode for this file.
5. **Use the Configuration Editor to restore FDDI and X.25 circuits as needed.**
6. **Reboot the target system to enable these circuits if they are restored.**

This completes the remote software distribution procedure.

5 Managing System Resources

This chapter describes how to use the Network Control Language (NCL) Interpreter to manage system resources. A set of NCL commands allows you to control various system resources. These commands are listed in Table 5-1.

Table 5-1 System Management Commands

Command	Function
`ATPING`	Send an AppleTalk Echo Protocol request message
`BOOT`	Boot the system
`DIS`	Disable a software entity or service
`E`	Enable a software entity or service
`EXIT`	Leave the NCL Interpreter
`HELP`	Obtain help text on NCL commands
`INSERT`	Mount the system diskette
`LOG`	Examine the RAM-based event log
`PAGE`	Enable *page mode*
`PASSWORD`	Assign/modify password protection
`PING`	Send a Internet Control Message Protocol echo request message
`REMOVE`	Dismount the system diskette
`STAMP`	Display software image data
`TELNET`	Establish an Internet virtual terminal connection
`TIME`	Get time/date information
`!`	Repeat last NCL command

The following sections describe each of these commands.

5.1 Booting the System

You use the `BOOT` command to boot the system.

Syntax: **`boot {image_name} {config_name}↵`**

where:

`image_name`

Is the optional name of the file that contains the software image. In the absence of this argument, the system defaults to *`ace.out`*.

`config_name`

Is the optional name of the file that contains the configuration data. In the absence of this argument, the system defaults to *`config`*.

Example: `boot↵`

Boot the system using the default files *`ace.out`* and *`config`*.

`boot testace.out` *`config.1`*↵

Boot the system using the software image contained in the file *`testace.out`* and the configuration data contained in the file *`config.1`*.

`boot testace.out↵`

Boot the system using the software image contained in the file *`testace.out`* and the default configuration file, *`config`*.

`boot - config.1↵`

Boot the system using the default software image (*`ace.out`*) and the configuration file *`config.1`*. Note that to boot the system with a specified configuration file (a file other than *`config`*) and the default software image (*`ace.out`*), you replace the first argument with a SPACE, hyphen SPACE sequence.

Use the following procedure to boot the system:

1. **Enter `BOOT` (and optional arguments if any), then press `[RETURN]` at the NCL prompt.**
2. **In response to the Enter current password prompt, enter the password followed by `[RETURN]`.**

   If you have not previously assigned a password, or if you have removed password protection, the system does not prompt for a password.
3. **Type `Y` at this prompt:**

   Do you want to reboot the system? [y/n]

   The console screen first displays **Sync'ing** . . . while it dismounts the system diskette, and then displays **REBOOTING THE SYSTEM**. When the system completes the reboot, it returns you to the Main Menu.
4. **If you do not wish to reboot the system, type `N` at the prompt:**

   Do you want to reboot the system? [y/n]

   Then the console screen displays **Boot aborted** and returns you to the NCL prompt.

The `BOOT` command dismounts the system diskette and flushes all disk I/O prior to booting the system, thereby ensuring that the system diskette remains uncorrupted. Consequently, it is recommended that you boot the system by means of the `BOOT` command, rather than by pressing the `RESET` button.

NOTE

The software image and configuration files revert to their respective defaults (*`ace.out`* and *`config`*) after every boot. Consequently, systems with Automatic Reboot enabled reboot from *`ace.out`* and *`config`* as they attempt to recover from failure.

5.2 Disabling Software Entities

You use the `DIS` (disable) command to remove an application software module, a circuit or another software object from service, as follows:

Syntax: `dis <identifier>↵`

where:

`identifier`

Is the managed object, the numeric equivalent, or a pathname.When disabling entities, you may use object identification codes to identify specific software objects. As a result, the following two command sequences are equivalent; both disable the DECnet Phase IV Router:

```
dis drs↵
dis 4↵
```

Example: `dis ip↵`

Disables the IP Router.

`dis cct.lab_net↵`

Disables the circuit named lab_net.

`dis cct.e21 cct.e22 cct.e51↵`

Disables the circuits named e.21, e.22, and e.51.

`dis echo↵`

Disables TCP echo service.

Refer to the section entitled *Displaying the Information Base* in Chapter 11 for a description of object identification codes. You cannot use `DIS` to remove lines or remove hardware entities from service.

NOTE

Disabling the XNS router (by using the `dis xrx` command results in the removal of all routing table entries (other than directly connected networks and static routes) and prevents access to the table through the `rgetxr` command. Re-enabling the router (by using the `en xrx` command) starts the router operating again as if you had rebooted the unit. Disabling and enabling the IPX router (by using the `dis ipx` and `en ipx` commands) has the same effect as for the XNS router except that the router's SAP table and its routing table are both cleared during the disable operation. When either router is disabled it will not forward any traffic or learn routes.

5.3 Enabling Software Entities

You use the `E (enable)` command to place an application software module, a circuit, an X.25 point-to-point dedicated virtual circuit, or another software entity into service, as follows:

Syntax: `e identifier>↵`

where:

`identifier`

Is the managed object, the numeric equivalent, or a pathname.When disabling entities, you may use object identification codes to identify specific software objects. As a result, the following two command sequences are equivalent; both disable the DECnet Phase IV Router:

```
e drs↵
e 4↵
```

Example: `e ip↵`

Enables the IP Router.

`e cct.lab_net↵`

Enables a circuit named lab_net.

`e cct.e21 cct.e22 cct.e51↵`

Enables the circuits named e.21, e.22, and e.51.

`e echo↵`

Enables TCP echo service.

You use `e` to restore to service software entities previously removed by `DIS` (disable), or those entities not auto-enabled at system start-up.

5.4 Leaving the NCL Interpreter

You use the `EXIT` command to leave NCL and return to the Main Menu, as follows:

Syntax: `exit↵`

5.5 Getting Help

You use the `HELP` command to display a summary listing of NCL commands and their syntax, as follows:

Syntax: `help↵`

5.6 Viewing the Event Log

You use the `LOG` command to display the RAM event log buffer, as follows:

Syntax: `log↵`

The `LOG` command provides access to the RAM event log. Refer to the chapter entitled *Using the Event Log* for information on interpreting the contents of the event log.

By default, the system allocates a 100-item circular (FIFO) RAM buffer to record event messages. If you enable logging during the configuration process, the system schedules resources to write the contents of the RAM event log buffer to a file (named *log*) on the system diskette.

When you establish a disk-logging session, you specify both the size and the name of the event file on the system diskette. You also may implement a log-file numbering feature that creates up to 10 log files (identified by a sequential numeric suffix, 0 through 9). With this feature enabled, the system creates a new log file each time it is rebooted. Prior to creating this new file, the system closes the previous log file, appends a numeric suffix to the file name, and saves the file on the system diskette.

`LOG` provides access to the RAM event log buffer; the NCL `TYPE` command provides access to the event log file on the system diskette.

Before displaying the contents of the RAM event log buffer, verify that *page mode* is enabled so that the buffer contents are displayed one screen at a time.

If the event log buffer exceeds 20 lines, the console displays **-- MORE --** below the last line of displayed data.

To view additional data:

1. **Press `[RETURN]` for one more line.**
2. **Type a number from `1` through `9` to display that number of additional lines.**
3. **Press any other key for an additional screen of data.**

4. **Press the `[LEFTARROW]` (⇐) to return to the NCL prompt.**
5. **When the system reaches the end of the buffer, it positions the cursor at the NCL prompt.**

5.7 Implementing Password Protection

The **`PASSWORD`** command assigns, changes, or removes password protection, as follows:

Syntax: `password↵`

NOTE

Only users with manager-level passwords, which are described in the next section, can access the **`PASSWORD`** command after it is initially set.

5.7.1 Assigning an Initial Password

When assigning an initial password, you must decide the level of security to place upon it.

- ♦ manager level, read/write

  With this type of password, users can access any menu (NCL, statistics screens, event log, and configuration editor).

- ♦ user level, read-only

  With this type of password, users can examine menus and the event log and can use the configuration editor to look at configuration files, but cannot **`SAVE`** a configuration file. Users at this level can execute *only* the following NCL commands:

DIR	RGETA	RGETI	RGETR	TYPE
HELP	RGETATA	RGETIR	RGETATR	
LIST	RGETATP	RGETIS	RGETS	
LOG	RGETB	RGETM	RGETW	
OSPF	RGETD	RGETMS	RGETX	
PAGE	RGETDA	RGETMW	RGETXS	
RGET	RGETDN	RGETNEXT	STAMP	

NOTE

You must be connected to the system at the manager level to change either of the passwords. Either password can be changed or deleted using the **Password** command. The session banner (located at the top of the console screen display) identifies the security level associated with the current login. The configuration editor will warn you at the beginning of an editor session if you are not authorized to save a file.

You use the following procedure to assign an initial manager- or user-level password:

1. **At the NCL prompt, type `password`, then press `[RETURN]`.**

   The console screen displays:

   Which password is changing? (Q) = Quit, (M) = Mgr, (U) = User

2. **Press `Q` to quit the password assigning procedure, `M` to assign a manager level password, or `U` to assign a user level password. If you select `M` or `U`, the system displays this message:**

   Enter new password

   Depending on your choice, the password assigned by the system will be either a manager- or user-level password.

3. **Now enter a string of one to 14 alphanumeric characters, followed by `[RETURN]`. Note that passwords are case-sensitive; WELLFLEET and wellfleet are not equivalent.**

   The system then redisplays the message:

   Enter new password.

4. **You confirm the password by entering the identical string, followed by `[RETURN]`.**

After you confirm the password, the system returns you to the NCL prompt.

5.7.2 Changing an Existing Password

You use the following procedure to change an existing password:

1. **Type `password`, then press `[RETURN]`, at the NCL prompt. The console screen displays Enter current password.**

2. **At Enter current password, enter the existing password, followed by `[RETURN]`.**

3. **At Enter new password, enter a string of one to 14 alphanumeric characters, followed by `[RETURN]`.**
4. **At Enter new password again, confirm the password by entering the identical string, followed by `[RETURN]`.**

   After you confirm the password, the system returns you to the NCL prompt.

5.7.3 Removing Password Protection

You use the following procedure to remove password protection:

1. **Type `password`, then press `[RETURN]`, at the NCL prompt. The console screen displays Enter current password.**
2. **At Enter current password, enter the existing password, followed by `[RETURN]`.**
3. **At Enter new password, press `[RETURN]`.**
4. **At Enter new password again, confirm the removal of password protection by pressing `[RETURN]`.**

   After you press `[RETURN]`, the system returns you to the NCL prompt.

5.8 Sending an Echo Request Message

`PING` is a program used within the Internet community to test the reachability of remote hosts. `PING` transmits an Internal Control Message Protocol (ICMP) echo request message to an IP address and waits for a response. If the system receives an echo response within the designated or default interval, the console displays a message indicating that the target address is “alive.” If the system does not receive a response within the specified interval, it displays a message indicating that the target did not respond, and prompts you to press any key to continue. (`PING` does not support loopback (pinging your own system) or broadcast addresses.)

You use the `PING` command to send an ICMP **echo request** message to a specified IP address, as follows:

Syntax: `ping <remote_host> {count} {timeout}↵`

where:

`remote_host`

Is the required remote host IP address, in dotted decimal notation.

`count`

Is the optional number of times to repeat the **`PING`** command. In the absence of an argument, the default is 1.

`timeout`

Is the optional timeout value (in seconds) for each ping. In the absence of an argument, the default is 5 seconds.

Example: **`ping 192.32.1.62↵`**

Pings IP address 192.32.1.62 one time (the default), and waits five seconds (the default) for a response.

`ping 192.32.1.62 20 1↵`

Pings IP address 192.32.1.62 twenty times, and waits one second for the response to each ping.

`ping 192.32.1.62 2↵`

Pings IP address 192.32.1.62 twice, and waits five seconds (the default) for the response to each ping.

5.9 Sending an AppleTalk Echo Request Message

You use the **`ATPING`** command to send an AppleTalk Echo Protocol (AEP) request message to a specified AppleTalk system address, as follows:

Syntax: **`atping <addr> {timeout}↵`**

where:

`addr`

Is the required AppleTalk system address (network number/ system identifier pair) of the target system.

`timeout`

Is the optional timeout value (in seconds) for the **`atping`**. In the absence of an argument, the default is 5 seconds.

Example: **`atping 178.46↵`**

Sends an AEP request message to the system whose AppleTalk address is 178.46 and wait 5 seconds (the default) for a reply.

`atping 178.46 1↵`

Sends an AEP request message to the system whose AppleTalk address is 178.46 and wait 1 second for a reply.

5.10 Working with the System Diskette

This section tells you how to mount, dismount, and replace the system disk.

5.10.1 Mounting the System Diskette

The `INSERT` command establishes the logical connection between the operating system and the system's drive. You use the `INSERT` command to mount the system diskette.

Syntax: `insert↵`

5.10.2 Dismounting the System Diskette

The **`REMOVE`** command disconnects the logical connection between the operating system and the system's drive. You use the REMOVE command to dismount the system diskette.

Syntax: `remove↵`

After you enter the command, the message **SYNC'ing** appears. Wait until this message disappears before removing the diskette.

Because the **`REMOVE`** command breaks the logical connection between the operating system and the system's drive, the drive is not accessible when the diskette has been removed, and any attempt to access the drive (such as with the **`DIR`** or **`TYPE`** commands) is unsuccessful.

5.10.3 Replacing the System Diskette

WARNING

Failure to follow this procedure may corrupt the system diskette.

If you want to replace the current system diskette with another, follow this procedure:

1. **Issue the `REMOVE` command to dismount the current system diskette.**
2. **Swap the diskettes.**
3. **Issue the `INSERT` command to mount the new diskette.**

5.11 Obtaining Software Image Data

You use the **STAMP** command to display the software-image stamp. Using the **STAMP** command allows you to identify the revision level of the system software.

Syntax: `stamp↵`

Example: `stamp↵`

```
image stamp: V5.60
    Fri May 31 13:00 14 EDT 1991
```

The revision level of this software is Version 5.60.

5.12 Establishing a Virtual Terminal Connection

You use the **TELNET** command to establish a virtual terminal connection to a remote host, as follows:

Syntax: `telnet <addr>↵`

where:

`addr`

Is the required IP address, in dotted decimal notation, of the remote host.

Example: `telnet 192.32.1.94↵`

Establishes a TCP virtual terminal connection to a remote host whose IP address is 192.32.1.94.

The **TELNET** command requires the installation of IP Router software in Slot 2 of the system. The telnet protocol allows you to establish a TCP connection between the system and a login system at a remote site. Once a connection is established, **TELNET** passes keystrokes from your system to the remote host.

When attempting to **TELNET** to a remote peer, keep in mind that the system supports a maximum of four simultaneous TCP connections.

5.13 Setting the Date and Time

You use the **TIME** command to set the system clock and/or calendar, as follows:

Syntax: `time <{mm/dd/yy} {hh:mm:ss}>↵`

where:

`mm/dd/yy`

Is the optional date in month/day/year format.

`hh:mm:ss`

Is the optional time in military (24 hour) format.

Example: `time 02/29/92 14:15:00↵`

Sets the time and date to 2:15 PM on February 29, 1992.

`time 12/23/89`

Sets the date to December 23, 1989.

`time 1:00:00↵`

Sets the time to 1:00 AM.

The **TIME** command requires a minimum of one argument (either date or time). You cannot use it without arguments to obtain the current date and time. The current date and time are always displayed in the upper left-hand corner of the console screen.

5.14 Enabling Page Mode

You use the **PAGE** command to enable or disable *page mode*, as follows:

Syntax: `page↵`

With *page mode* enabled (the default condition), system output is displayed on the console screen one page (twenty lines) at a time. With *page mode* disabled, system output is displayed as a continuous data stream.

5.15 Repeating the Previous NCL Command

You use the **!** command to repeat the previous NCL command.

Syntax: `! {n}↵`

where:

n

Is an optional numeric value that specifies the number of times to repeat the previous NCL command.

Example: `get lb.jrb.recv↵`

`!↵`

Repeats the **GET** command to obtain an updated packet count.

6 Accessing Application-Specific Tables

Some NCL commands work with the Simple Network Management Protocol (SNMP) management agent software and the IP router software to provide access to application specific routing and configuration tables maintained by local or remote peer systems. Table 6-1 lists these NCL commands.

Table 6-1 Application Table Access Commands

Command	Function
RGETAT	Obtain the Configuration Table for a specified network.
RGETATR	Obtain the AppleTalk Routing Table for a specified network system.
RGETATA	Obtain the AppleTalk Address Resolution Protocol (AARP) Table for a specified network system.
RGETB	Obtain the Bridge Forwarding/Filtering Table.
RGETD	Obtain the DECnet Phase IV Configuration Table.
RGETDN	Obtain the DECnet Phase IV Router Level 1 Routing Table.
RGETDA	Obtain the DECnet Phase IV Router Level 2 Routing Table.
RGETXR	Obtain the IPX Routing Table.
RGETIR	Obtain the XNS Routing Table.
RGETIS	Obtain the IPX Service Advertising Protocol Table.
RGETRIF	Obtain the RIF cache for a local or remote system.

6.1 Obtaining the AppleTalk Configuration Table

You use the **RGETAT** command to obtain a formatted version of the AppleTalk Router Configuration Table for a local or remote network, as follows:

Syntax: `rgetat <addr> {comm}↲`

where:

`addr`

Is the required IP address (in dotted decimal notation) of the target system.

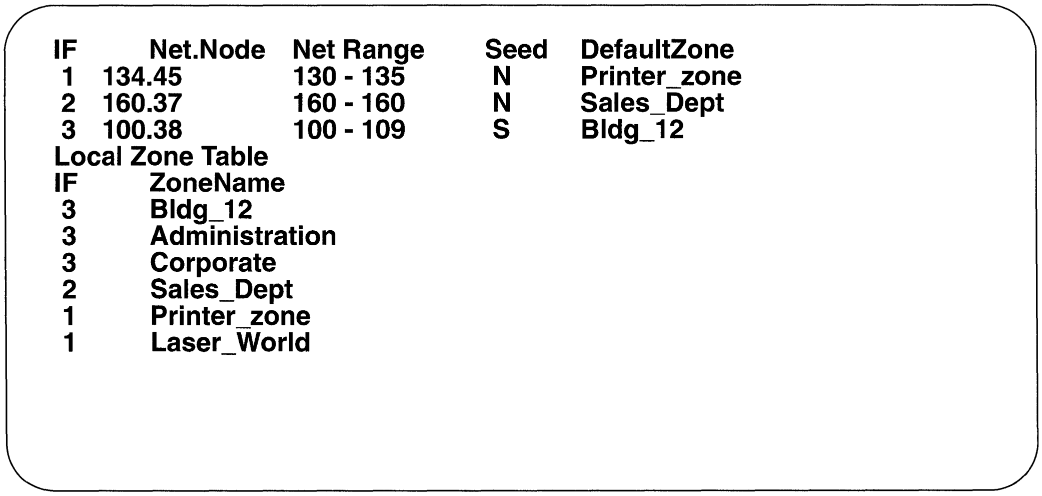
`comm`

Is the optional name of the SNMP community that enables access to the system specified by **<addr>**. In the absence of an argument, RGETAT defaults to **public**.

Example: `rgetat 192.32.1.163↲`

Displays the status of all AppleTalk circuit groups of the system whose IP address is 192.32.1.163.

In response, the console screen displays the AppleTalk Router Configuration Table. Figure 6-1 shows a sample table.



```
IF      Net.Node  Net Range     Seed  DefaultZone
 1  134.45        130 - 135      N    Printer_zone
 2  160.37        160 - 160      N    Sales_Dept
 3  100.38        100 - 109      S    Bldg_12
Local Zone Table
IF      ZoneName
 3      Bldg_12
 3      Administration
 3      Corporate
 2      Sales_Dept
 1      Printer_zone
 1      Laser_World
```

Figure 6-1 AppleTalk Router Configuration Table

Individual fields in Figure 6-1 are as follows:

- **IF** contains the system-assigned interface number that identifies the AppleTalk port.
- **Net.Node** contains the AppleTalk address (the network number or node (system) identifier pair) of each AppleTalk Router port.
- **Net Range** contains the range of network numbers available to systems on the directly-connected medium.
- **Seed** specifies whether the router is an AppleTalk seed router (**Y** = seed router; **N** = non-seed router).
- **DefaultZone** contains the default zone name for **Net Range**.
- **Local Zone Table** contains a list of zone names serviced by each of the Appletalk Router ports. This list includes the port's default zone name. The maximum number of zones supported is 36.

6.2 Obtaining the AppleTalk Routing Table

You use the `RGETATR` command to obtain a formatted version of the AppleTalk Routing Table for a local or remote network system, as follows:

Syntax: `rgetatr <addr> {comm}↵`

where:

`addr`

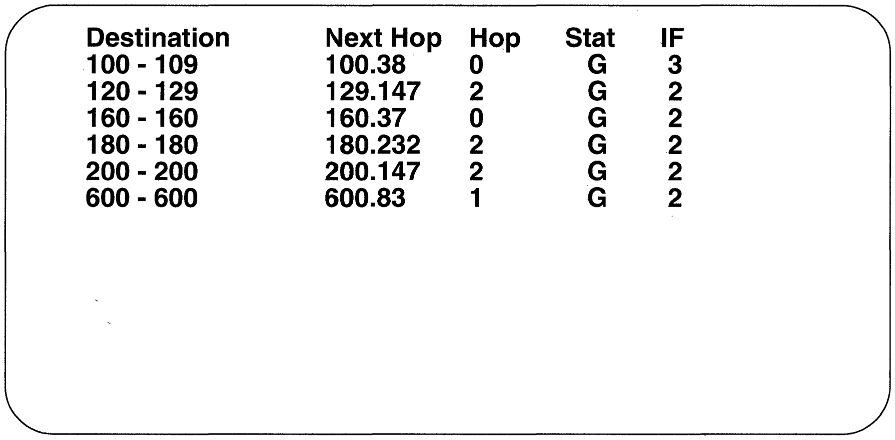
Is the required IP address (in dotted decimal notation) of the target system.

`comm`

Is the optional name of the SNMP community that enables access to the system specified by **<addr>**. In the absence of an argument, RGETATR defaults to **public**.

Example: `rgetatr 192.32.1.163↵`

Displays the AppleTalk Routing Table of the system whose IP address is 192.32.1.163. Figure 6-2 shows a sample AppleTalk Routing Table.



```
Destination      Next Hop  Hop    Stat   IF
100 - 109        100.38    0        G    3
120 - 129        129.147   2        G    2
160 - 160        160.37    0        G    2
180 - 180        180.232   2        G    2
200 - 200        200.147   2        G    2
600 - 600        600.83    1        G    2
```

Figure 6-2 AppleTalk Routing Table

Individual fields in Figure 6-2 are as follows:

- **Destination** contains the destination network range.
- **Next Hop** contains the AppleTalk address (the network number or node (system) identifier pair) of the next hop router.
- **Hop** contains the number of hops to **Destination.**
- **Stat** contains the state of the routing entry: (**G**)ood, (**S**)uspect, or (**B**)ad.
- **IF** contains the system-assigned interface number that accesses **Next Hop**.

6.3 Obtaining the AppleTalk Address Resolution Table

You use the `RGETATA` command to obtain a formatted version of the AppleTalk Address Resolution Protocol Table for a local or remote network system, as follows:

Syntax: `rgetata  <addr>  {comm}↵`

where:

`addr`

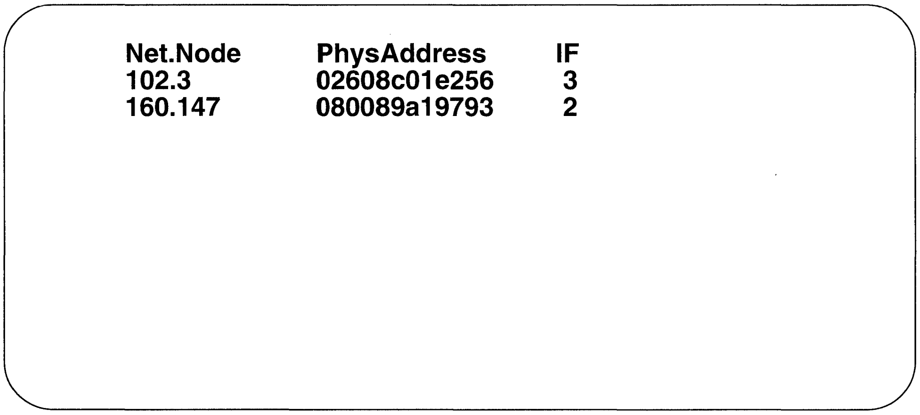
Is the required IP address (in dotted decimal notation) of the target system.

`comm`

Is the optional name of the SNMP community that enables access to the system specified by `<addr>`. In the absence of an argument, RGETATR defaults to `public`.

Example: `rgetata 192.32.1.163↵`

Displays the AppleTalk Address Resolution Table of the system whose IP address is `192.32.1.163`. Figure 6-3 shows a sample table.



```
Net.Node       PhysAddress      IF
102.3          02608c01e256      3
160.147        080089a19793      2
```

Figure 6-3 AppleTalk Address Resolution Table

Individual fields in Figure 6-3 are as follows:

- ♦ **Net.Node** contains the AppleTalk address (the network number or node (system) identifier pair).
- ♦ **PhysAddress** contains the physical address of **Net.Node**.
- ♦ **IF** contains the system-assigned value that identifies the interface to **Net.Node.**

6.4 Obtaining the Bridge Forwarding/Filtering Table

You use the **RGETB** command to obtain a formatted version of the Bridge Forwarding/ Filtering Table for a local or remote peer system, as follows:

Syntax: `rgetb <addr> {comm}↵`

where:

`addr`

Is the required IP address (in dotted decimal notation) of the local or remote peer system.

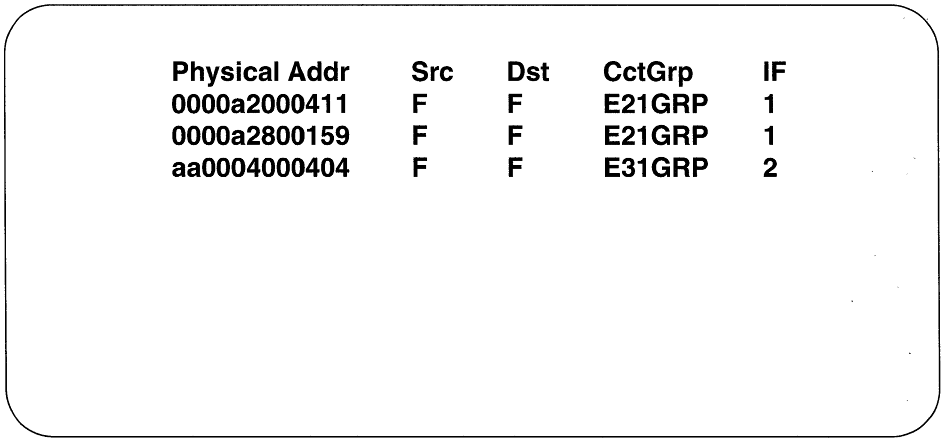
`comm`

Is the optional name of the SNMP community that enables access to `<addr>`. In the absence of an argument, RGETB defaults to `public`.

Example: `rgetb 192.32.1.94↵`

Displays the Bridge forwarding/filtering table for the system whose IP address is `192.32.1.94`.

In response, the console screen displays the Bridge Forwarding/ Filtering Table. Figure 6-4 shows a sample filtering/forwarding table.



```
Physical Addr    Src   Dst   CctGrp   IF
0000a2000411     F     F     E21GRP   1
0000a2800159     F     F     E21GRP   1
aa0004000404     F     F     E31GRP   2
```

Figure 6-4 Bridge Filtering/Forwarding Table

Individual fields in Figure 6-4 are as follows:

- **Physical Addr** lists the MAC-level addresses learned by the bridge.
- **Src** contains the disposition of frames that contain **Physical Addr** in the source address field of the Ethernet frame.
    - **F** indicates that frames are forwarded; **D** indicates that frames are dropped.
- **Dst** contains the disposition of frames addressed to **Physical Addr**.
    - **F** indicates that frames are forwarded; **D** indicates that frames are dropped.
- **CctGrp** contains the circuit group connected to **Physical Addr**.
- **IF** contains the system-assigned interface number that corresponds to **CCtGrp**.

6.5 Obtaining the DECnet Phase IV Router Configuration Table

You use the `RGETD` command to obtain a formatted version of the DECnet Router Configuration Table for a local or remote peer system, as follows:

Syntax: `rgetd <addr> {comm}↵`

where:

`addr`

Is the required IP address (in dotted decimal notation) of the local or remote peer system.

`comm`

Is the optional name of the SNMP community that enables access to `<addr>`. In the absence of this argument, `RGETD` defaults to `public`.

Example: `rgetd 192.32.1.94↵`

Displays the status of all DECnet circuit groups of the system whose IP address is `192.32.1.94`.

In response, the console screen displays the DECnet Router Configuration Table. Figure 6-5 shows a sample table.

```
DECNET configuration for <ip_address>.

Name     Stat  Cst   Hello  Prior   IF
E21GRP   up    4     15     127     1
E22GRP   up    10    15     127     2
E41GRP   up    3     15     64      4
E42GRP   up    3     15     127     5
```

Figure 6-5 DECnet Phase IV Router Configuration Table

Individual fields in Figure 6-5 are as follows:

- **ip_address** contains the IP address (in dotted decimal notation) of the local or remote peer system.
- **Name** lists the DECnet circuit groups.
- **Stat** contains the current status (up or down) of **Name**.
- **Cst** contains the cost associated with **Name**. Costs are set by the **Circuit Cost** parameter in `config`.
- **Hello** contains the interval (in seconds) between DECnet Hello messages. This interval is set by the **Hello Timer** parameter in `config`.
- **Prior** contains the relative priority of the router. Priority is set by the **Router Priority** parameter in `config`.
- **IF** contains the system-assigned interface number that identifies **Name**.

6.6 Obtaining the DECnet Phase IV Router Level 1 Routing Table

You use the **RGETDN** command to obtain a formatted version of the DECnet Router Level 1 (intra-area) Routing Table for a local or remote system, as follows:

Syntax: `rgetdn <addr> {comm}↵`

where:

`addr`

Is the required IP address (in dotted decimal notation) of the local or remote peer system.

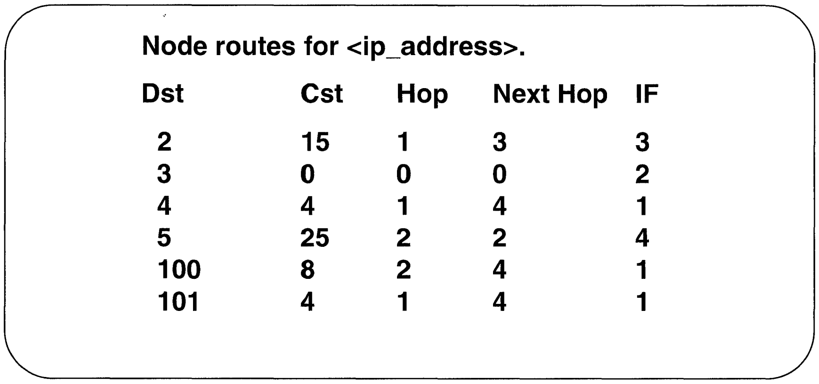
`comm`

Is the optional name of the SNMP community that enables access to **<addr>**. In the absence of this argument, **RGETDN** defaults to **public**.

Example: `rgetdn 192.32.1.94↵`

Displays the DECnet router intra-area routing table for the system whose IP address is **192.32.1.94**.

In response, the console screen displays DECnet Level 1 routing information. Figure 6-6 shows a sample DECnet Level 1 Routing Table.



```
Node routes for <ip_address>.
Dst        Cst    Hop    Next Hop   IF
 2         15     1      3          3
 3         0      0      0          2
 4         4      1      4          1
 5         25     2      2          4
 100       8      2      4          1
 101       4      1      4          1
```

Figure 6-6 DECnet Level 1 Routing Table

Individual fields in Figure 6-6 are as follows:

- ♦ **ip_address** contains the IP address (in dotted decimal notation) of the target system.
- ♦ **Dst** contains the destination system number.
- ♦ **Cst** contains the circuit cost to **Dst**.
- ♦ **Hop** contains the number of router hops to **Dst**.
- ♦ **Net Hop** contains the DECnet system number of the next hop router.
- ♦ **IF** contains the system-assigned interface number that accesses **Next Hop**.

6.7 Obtaining the DECnet Phase IV Router Level 2 Routing Table

You use the `RGETDA` command to obtain a formatted version of the DECnet Router Level 2 (intra-area) Routing Table for a local or remote system, as follows:

Syntax: `rgetda <addr> {comm}↵`

where:

`addr`

Is the required IP address (in dotted decimal notation) of the local or remote peer system.

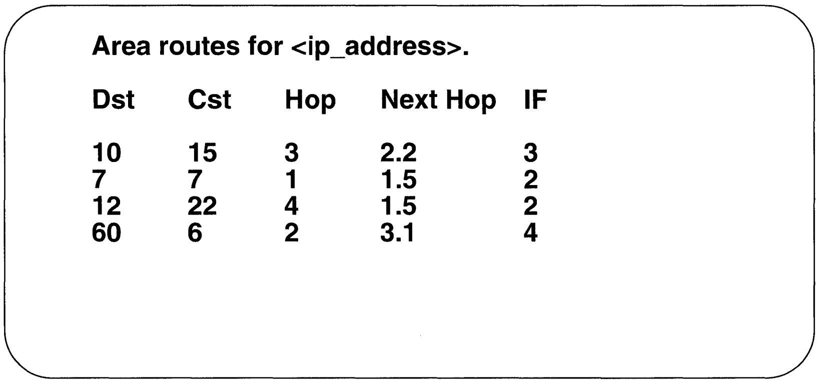
`comm`

Is the optional name of the SNMP community that enables access to `<addr>`. In the absence of this argument, `RGETDA` defaults to `public`.

Example: `rgetda 192.32.1.94↵`

Displays the DECnet router intra-area routing table for the system whose IP address is `192.32.1.94`.

In response, the console screen displays DECnet Level 2 routing information. Figure 6-6 shows a sample DECnet Level 2 Routing Table.



```
Area routes for <ip_address>.

Dst     Cst     Hop     Next Hop   IF

10      15      3       2.2        3
7       7       1       1.5        2
12      22      4       1.5        2
60      6       2       3.1        4
```

Figure 6-7 DECnet Level 2 Routing Table

Individual fields in Figure 6-6 are as follows:

- **ip_address** contains the IP address (in dotted decimal notation) of the target system.
- **Dst** contains the destination system number.
- **Cst** contains the circuit cost to **Dst**.
- **Hop** contains the number of router hops to **Dst**.
- **Net Hop** contains the DECnet system number of the next hop router.
- **IF** contains the system-assigned interface number that accesses **Next Hop**.

6.8 Obtaining the XNS Routing Table

You use the `RGETXR` command to obtain a formatted version of the XNS Routing Table for a local or remote peer system, as follows:

Syntax: `rgetxr <addr> {comm}`↵

where:

`addr`

Is the required IP address (in dotted decimal notation) of the local or remote peer system.

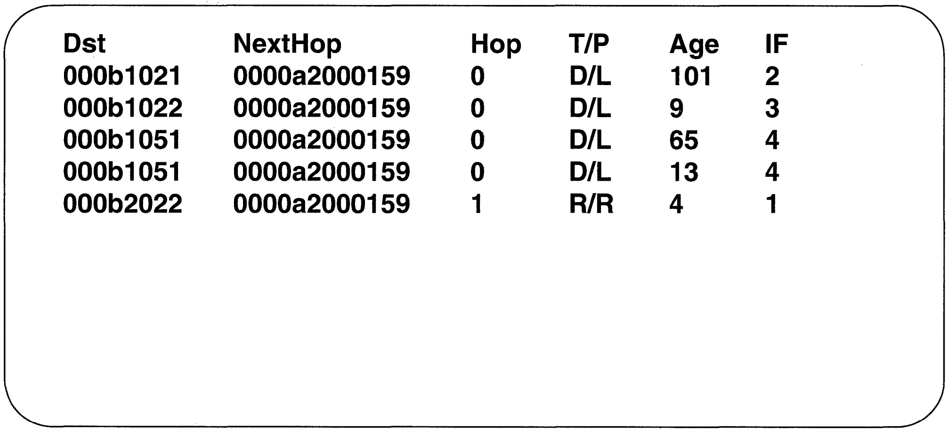
`comm`

Is the optional name of the SNMP community that enables access to the system specified by **`<addr>`**. In the absence of this argument, **`RGETIR`** defaults to **`public`**.

Example: **`rgetxr 192.32.1.94↵`**

Displays the XNS routing table for the system whose IP address is **`192.32.1.94`**.

In response, the console screen displays the XNS Routing Table. Figure 6-8 shows a sample table.



```
Dst         NextHop        Hop   T/P   Age   IF
000b1021    0000a2000159   0     D/L   101   2
000b1022    0000a2000159   0     D/L   9     3
000b1051    0000a2000159   0     D/L   65    4
000b1051    0000a2000159   0     D/L   13    4
000b2022    0000a2000159   1     R/R   4     1
```

Figure 6-8 Sample XNS Routing Table

Individual fields in Figure 6-8 are as follows:

- **Dst** lists the XNS network number of the destination network (expressed as an 8-digit hexadecimal value).
- **NextHop** contains the XNS host address of the next hop router.
- **Hop** contains the hop count to **Dst**.
- **T** contains the route type as follows:
  - **D**-- a direct (local) route
  - **I**-- an invalid route
  - **R**-- a remote route

- **P** contains how the route type was learned, as follows:
  - **L**-- statically-configured route
  - **R**-- Routing Information Protocol
- **Age** contains the number of seconds since the route was learned.
- **IF** contains the system-assigned interface number over which **NextHop** is reached.

6.9 Obtaining the IPX Routing Table

You also use the `RGETIR` command to display a formatted version of the IPX Routing Table for a local or remote peer node, as follows:

Syntax: `rgetir <addr> {comm}↵`

where:

`addr`

Is the required IP address (in dotted decimal notation) of the local or remote Wellfleet system.

`comm`

Is the optional name of the SNMP community that enables access to the system specified by `<addr>`. In the absence of this argument, the default is `public`.

Example: `rgetir 192.32.1.94↵`

Displays the IPX routing table for the system whose IP address is `192.32.1.94`. In response, the console screen displays the IPX Routing Table, as shown in Figure 6-8.

```
Dst         NextHop         Hop    T/P    Age    IF
000b1021    0000a2000159    0      D/L    101    2
000b1022    0000a2000159    0      D/L    9      3
000b1051    0000a2000159    0      D/L    65     4
000b1051    0000a2000159    0      D/L    13     4
000b2022    0000a2000159    1      R/R    4      1
```

Figure 6-9 Sample IPX Routing Table

Individual fields in Figure 6-8 are as follows:

- **Dst** lists the IPX network number of the destination network (expressed as an 8-digit hexadecimal value).
- **NextHop** contains the IPX host address of the next hop router.
- **Hop** contains the hop count to **Dst**.
- **T** contains the route type as follows:
  - **D**-- a direct (local) route
  - **I**-- an invalid route
  - **R**-- a remote route
- **P** contains how the route type was learned, as follows:
  - **L**-- statically-configured route
  - **R**-- Routing Information Protocol
- **Age** contains the number of seconds since the route was learned.
- **IF** contains the node-assigned interface number over which **NextHop** is reached.

6.10 Obtaining the IPX Servers Table or Obtaining the Service Advertising Protocol Table

You use the **RGETIS** command to display the contents of the Service Advertising Protocol (SAP) table.

Syntax: `rgetis <addr> {comm}↵`

where:

`addr`

Is the required IP address (in dotted decimal notation) of the local or remote Wellfleet system.

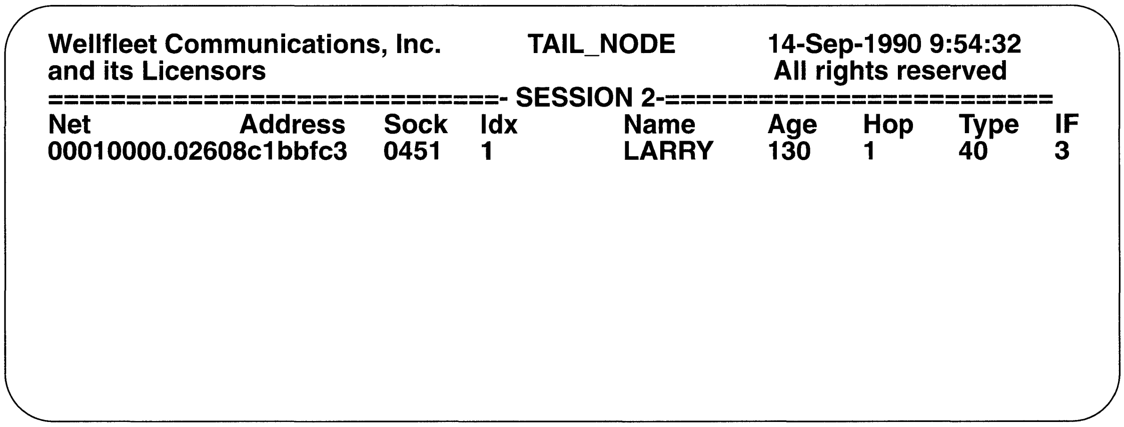
`comm`

Is the optional name of the SNMP community that enables access to the system specified by `<addr>`. In the absence of this argument, the default is `public`.

Example: `rgetis 192.32.1.84↵`

Displays the IPX systems that the Wellfleet system whose IP address is `192.32.1.84` has discovered by listening to the SAP protocol. The console screen displays the Servers Table, which is shown in Figure 6-10.

.



```
Wellfleet Communications, Inc.        TAIL_NODE        14-Sep-1990 9:54:32
and its Licensors                                       All rights reserved
=============================- SESSION 2-==========================
Net              Address     Sock   Idx          Name     Age    Hop    Type   IF
00010000.02608c1bbfc3        0451   1            LARRY    130    1      40     3
```

Figure 6-10 IPX Servers Table

The following fields are found in the SAP Table shown in Figure 6-10.

- **Net Address** -- IPX network and station address of the system described in a row of the table (value is in hexadecimal).
- **Sock** -- IPX Socket through which the system is addressed (value is in hexadecimal).
- **Idx** -- Small index integer, of no IPX protocol significance, which is used to distinguish multiple systems sharing the same Net, Address, and Sock.
- **Name** -- IPX system's name. If the name is more than 16 characters long, only the first 15 characters will be displayed, and the 16th character will appear as an asterisk '*' to indicate the name is incomplete.
- **Age** -- Number of seconds since a SAP advertisement of this system was last received.
- **Hop** -- Number of hops to the destination specified in Net Address.
- **Type** -- Hexadecimal number that expresses the type of service supplied by the named system, as follows:

Type Number	Description
0	Unknown
3	Print Server
4	File Server
5	Job Server
9	Archive Server
24	Remote Bridge Server
47	Advertising Print Server

- **IF** -- Interface number of the circuit group through which the system can be reached.

6.11 Obtaining the Source Routing RIF Cache

You use the NCL command `rgetrif` to obtain a formatted version of the Routing Information Field (RIF) cache for a local or remote network system as follows:

Syntax: `rgetrif <addr>↵`

where:

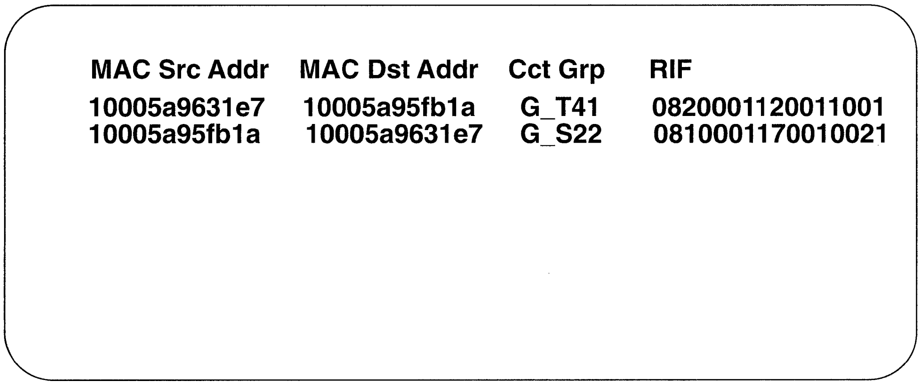
`addr`

Is the required IP address (in dotted decimal notation) of the local or remote target system.

Example: `rgetrif 192.32.1.94`

Displays the source routing RIF cache for the system whose IP address is `192.32.1.94`.

In response, the console screen displays the source routing RIF Cache. Figure 6-11 shows a sample RIF Cache.



```
MAC Src Addr    MAC Dst Addr    Cct Grp    RIF
10005a9631e7    10005a95fb1a     G_T41     0820001120011001
10005a95fb1a    10005a9631e7     G_S22     0810001170010021
```

Figure 6-11 Sample RIF Cache

Individual fields in Figure 6-11 are as follows:

- **MAC Src Addr** lists the MAC-level (physical) address of the source interface.
- **MAC Dst Addr** lists the MAC-level (physical) address of the destination interface.
- **Cct Grp** lists the circuit group that connects the source interface to the first physical interface used in the RIF route.

- **RIF** describes the path (in hexadecimal format) used to source route packets between the source system and the destination. The first two bytes contain the routing control (RC) field that describes routing type, field length, direction bit and largest frame size. Next are a series of two-byte route descriptors; each describing the LAN ID and bridge ID number of the next hop required to reach the destination. (See Figure 6-12 and Figure 6-13).

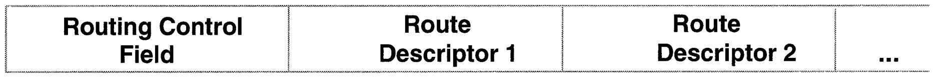
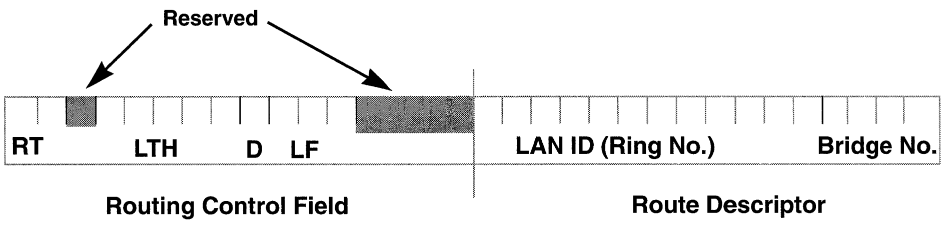


Figure 6-12 Basic RIF Format



Bits	Description	Specifications
RT	RIF type	00: Specifically Routed Frame (SRF) 11: Spanning Tree Explorer (STE) 10: All Routes Explorer (ARE)
LTH	Total length in bits of the RIF field	Maximum = 18 bytes
D	Direction that the frame traverses the LANs	0: frame moves forward 1: frame moves in reverse
LF	Largest frame size that can be handled by this route	Maximum = 4472 bytes

Figure 6-13 Detailed RIF format

7 Managing OSPF

This chapter describes how to use NCL to monitor OSPF (Open Shortest Path First). OSPF is an internal gateway routing protocol (IGP) that functions as an autonomous system (AS), a collection of routers all running a single IGP. OSPF is an open routing protocol for which the protocol specification is openly available and is not proprietary to any single vendor. OSPF uses an SPF routing protocol that uses SPF technology.

7.1 Displaying List of OSPF Neighbors

The `OSPF NBRS` command displays the status of all the neighbors currently maintained by OSPF. Figure 7-1 shows a sample screen of output from this command.

```
Wellfleet Communications, Inc    LN_1            4-Jan-91    13:29:03
======================- SESSION 1-==============================
Interface          Router Id       Nbr IP Addr       State   mode   Priority
-----------------------------------------------------------------------------
Area:   0.0.0.0
190.190.190.10     13.13.13.13     190.190.190.13    Full    Slave  5
190.190.190.10     12.12.12.12     190.190.190.12    Full    Slave  5



LN_1:
```

Figure 7-1 Sample Output from `OSPF NBRS` Command

Syntax: `OSPF NBRS↵`

where:

`Area`

Is the area ID to which the neighbor belongs.

`Interface`

Is the interface number of the network to which the neighbor is attached.

`Interface`

Is the interface number of the network to which the neighbor is attached.

`Router Id`

Is a 32-bit number that uniquely identifies the router in the Autonomous System. One algorithm for Router ID assignment is to choose the largest or smallest IP address assigned to the router.

`Nbr Ip Addr`

Is the IP address of the neighbor.

`State`

Is the state of the connection with the neighbor.

Down:
Is the initial state of a neighbor conversation. It indicates that there has been no recent information received from the neighbor.

Attempt:
Is the state that is only valid for neighbors attached to non-broadcast networks. This state indicates that no recent information has been received from the neighbor, but that a more concerted effort should be made to contact the neighbor.

Init:
Is the state where a Hello packet has recently been seen from the neighbor. However, bi-directional communication has not yet been established with the neighbor.

2-Way:
Is the state of bidirectional communication with the neighbor.

ExStart:

Is the first step in creating an adjacency with the neighbor. The goal of this step is to decide which router is the master.

Exchange:

Is the state where two neighboring routers begin synchronizing their databases by sending Database Description packets.

Loading:

Is the state where Link State Request packets are sent to the neighbor asking for the more recent advertisements that have been discovered but net yet received in the Exchange state.

Full:

Is the state where full adjacency has been established with the neighbor, and the databases are synchronized.)

`Mode`

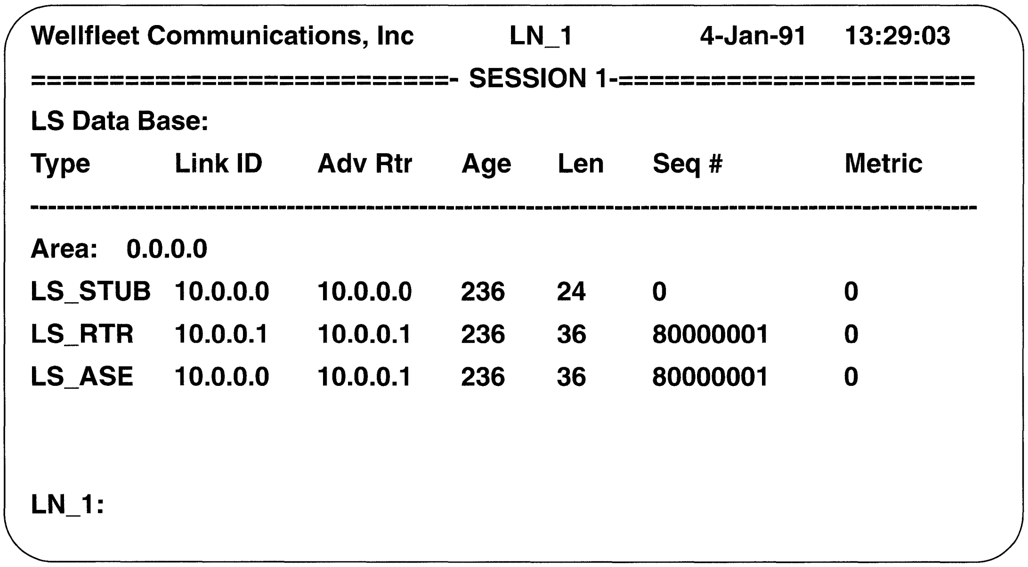
Is Master or Slave mode of the neighbor.(The master sends the first Database Description packet, and is the only part that is allowed to retransmit. The slave can only respond to master's Database Description packet.)

`Priority`

The Router Priority of the neighbor. This is used when selecting the Designated Router for the attached network.

7.2 Displaying the OSPF Link State Database

The **`OSPF LSDB`** command displays the contents of OSPF's Link State Database. Figure 7-2 shows a sample screen of output from this command.



```
Wellfleet Communications, Inc          LN_1          4-Jan-91    13:29:03
==========================- SESSION 1-======================
LS Data Base:
Type        Link ID     Adv Rtr     Age    Len    Seq #           Metric
-----------------------------------------------------------------------------------------
Area:   0.0.0.0
LS_STUB   10.0.0.0     10.0.0.0     236    24     0               0
LS_RTR    10.0.0.1     10.0.0.1     236    36     80000001        0
LS_ASE    10.0.0.0     10.0.0.1     236    36     80000001        0



LN_1:
```

Figure 7-2 Sample Output from `OSPF LSDB` Command

Syntax: `OSPF LSDB`↵

where:

`Area`

Is the area ID of the link state database.

`Type`

Is one of the various types of link state database entries:

LS_RTR:
: This type describes the state and cost of the router's links (or interface) to the area. Each router in an area originates a router links advertisement.

LS_NET:
: This type describes all routers attached to the network, including the Designated Router itself. These advertisements are originated by the network's Designated Router.

LS_SUM_NET:
: This type describes a route to a destination network which

belongs to the AS, yet is outside the area. These advertisements are originated by area border routers.

LS_SUM_ASB:
: This type describes a route to an AS boundary router. These advertisements are also originated by the area border routers.

LS_STUB:
: This type describes a default route for a stub area. In a stub area, instead of importing external routes each area border router originates a "default summary link" (Link State ID = DefaultDestination) into the area.

LS_ASE:
: This type describes a route to a destination network which is external to the AS. These advertisements are originated by AS boundary routers.

`Link ID`

Identifies the object that this router link connects to. Value depends on the link's type:

LS_RTR:
: Is the neighbor's OSPF Router ID.

LS_NET:
: Is the Designated Router's IP interface address.

LS_SUM_NET:
: Is the destination IP network number.

LS_SUM_ASB:
: Is the AS boundary router's OSPF Router ID.

LS_STUB:
: Is set to DefaultDestination(0.0.0.0).

LS_ASE:
: Is the destination IP network number.

`Advrtr`

Is the field that specifies the OSPF Router ID of the advertisement's originator. (See **`Type`** field.)

`Age`

Is the time in seconds since the link state advertisement was originated.

`Len`

Is the length in bytes of the link state advertisement.

`Seq`

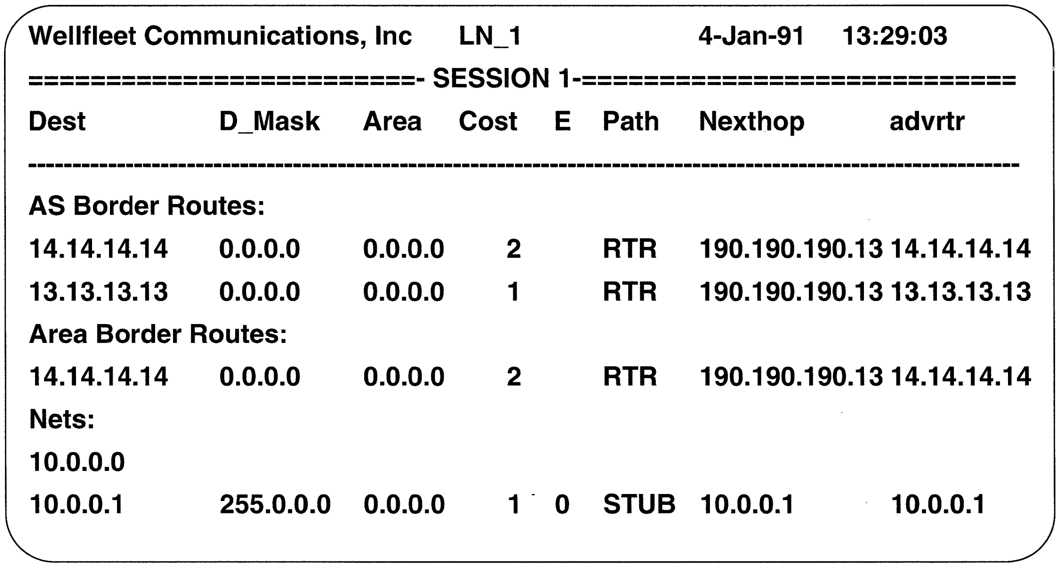
Is used to detect old and duplicate link state advertisements. Successive instances of a link state advertisement are given successive LS sequence numbers.

`Metric`

Is the cost of this route.

7.3 Displaying the OSPF Routing Table

The `OSPF RTAB` command displays the contents of OSPF's routing table. Figure 7-3 shows a sample screen of output from this command.



```
Wellfleet Communications, Inc     LN_1                4-Jan-91    13:29:03
=======================- SESSION 1-===========================
Dest            D_Mask     Area      Cost   E   Path    Nexthop          advrtr
-------------------------------------------------------------------------------------------
AS Border Routes:
14.14.14.14     0.0.0.0    0.0.0.0      2       RTR     190.190.190.13 14.14.14.14
13.13.13.13     0.0.0.0    0.0.0.0      1       RTR     190.190.190.13 13.13.13.13
Area Border Routes:
14.14.14.14     0.0.0.0    0.0.0.0      2       RTR     190.190.190.13 14.14.14.14
Nets:
10.0.0.0
10.0.0.1        255.0.0.0  0.0.0.0      1   0   STUB    10.0.0.1         10.0.0.1
```

Figure 7-3 Sample Output from `OSPF RTAB` Command

Syntax: `OSPF RTAB↵`

where:

`Dest`

Is the IP network number of the destination.

`D_mask`

Is the IP network mask of the destination.

`Area`

Is the entry's associated area. This field indicates the area whose link state information has led to the routing table entry's collection of paths.

`Cost`

Is the link state cost of the path to the destination. For all paths except type 2 external paths this describes the entire path's cost. For type 2 external paths, this field describes the cost of the portion of the path internal to the AS.

`E`

Is valid only for type 2 external paths. This field indicates the cost of the path's external portion. This cost has been advertised by an AS boundary router, and is the most significant part of the total path cost.

`Path`

Is the types of the path:

INT:
: Intra-area paths indicate destinations belonging to one of the router's attached areas.

EXT:
: AS external paths are paths to destinations external to the AS.

SUM:
: Inter-area paths are paths to destinations in other OSPF areas.

`Nexthop`

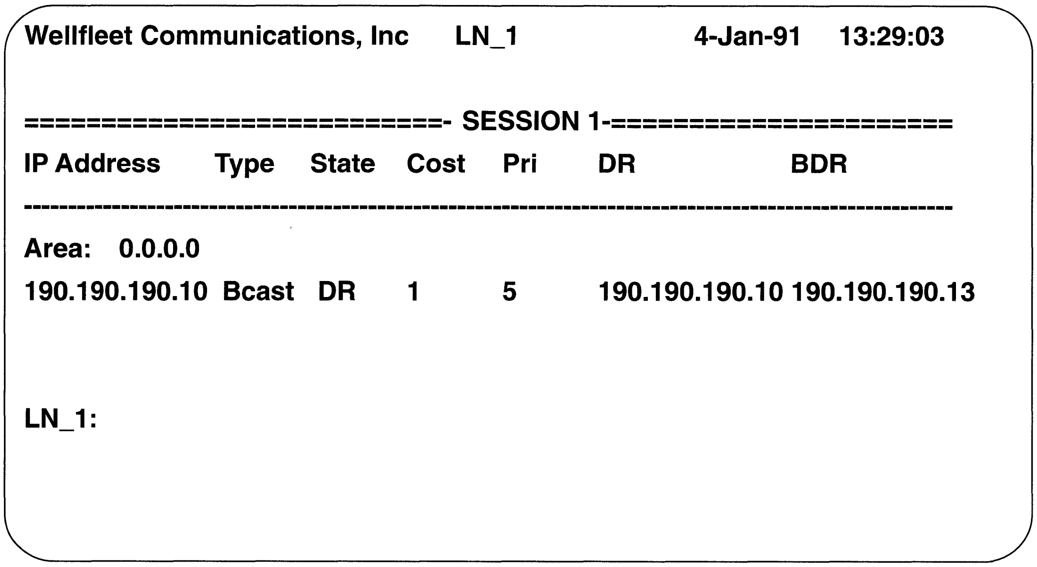
Is the next hop to the destination.

`Advrtr`

Is valid only for inter-area and AS external paths. This field indicates the Router ID of the router advertising the summary link or AS external link that led to this path.

7.4 Displaying Status of OSPF Interfaces

The `OSPF INTF` command displays the status of the interfaces over which OSPF is running. Figure 7-4 shows a sample screen of output from this command.



```
Wellfleet Communications, Inc     LN_1              4-Jan-91     13:29:03


==========================- SESSION 1-=====================
IP Address      Type    State    Cost    Pri      DR                BDR
--------------------------------------------------------------------------
Area:    0.0.0.0
190.190.190.10  Bcast   DR       1       5        190.190.190.10 190.190.190.13



LN_1:
```

Figure 7-4 Sample Output form `OSPF INTF` Command

Syntax: `OSPF INTF`↵

where:

`Area`

Is the area ID to which the attached network interface belongs.

`IP Address`

Is the IP address associated with the interface.

`Type`

Is the kind of network to which the interface attaches:

Bcast:
Broadcast.

PtoP:
Point-to-point.

Virt:
: Virtual link.

`State`

Is the functional level of an interface. This Status field refers to the status of the interface in a neighbor adjacency relationship.

Down:
: Is the in-operable interface.

Loopback:
: Is a self-referential interface. This does not represent a connection to a network.

Waiting:
: Specifies that an initial packet has be sent. The interface is waiting to hear from other routers on the network in order to perform the designated router selection.

Point-to-point:
: Is the interface is point to point and therefore has not designated router election.

DR:
: Is the Designated router on this interface.

Dr Other:
: Is neither DR or BDR on this interface

Backup:
: Is the BDR on this network interface.

`Cost`

Is the cost of sending a packet on the interface, expressed in the link state metric.

`Pri`

Is the router priority. When two routers attached to a network both attempt to become Designated Router, the one with the highest Router Priority takes precedence.

`DR`

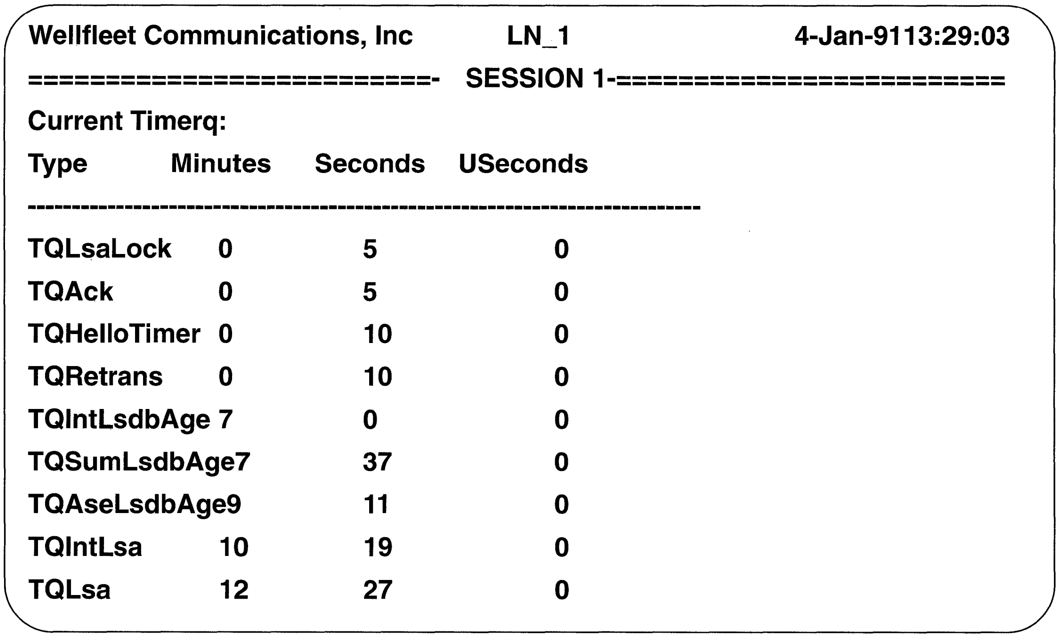
Is the Designated Router selected for the attached network.

`BDR`

Is the Backup Designated Router selected for the attached network.

7.5 Displaying the OSPF Timer

The `OSPF TQ` command displays the top ten timers on OSPF's timer queue. Figure 7-5 shows a sample screen of output from this command



```
Wellfleet Communications, Inc        LN_1                    4-Jan-9113:29:03
==========================-  SESSION 1-========================
Current Timerq:
Type         Minutes    Seconds    USeconds
--------------------------------------------------------------------
TQLsaLock      0          5            0
TQAck          0          5            0
TQHelloTimer   0          10           0
TQRetrans      0          10           0
TQIntLsdbAge 7            0            0
TQSumLsdbAge7             37           0
TQAseLsdbAge9             11           0
TQIntLsa       10         19           0
TQLsa          12         27           0
```

Figure 7-5 Sample Output from `OSPF TQ` Command

Syntax: `OSPF TQ↵`

where:

`Type Minutes Seconds USeconds`

Is the type of timer and the time it was issued.

OSPF issues the following types of timers:

`TQHelloTimer`

Is the next time a Hello packet will be sent out.

`TQIntLsa`

Indicates the Dijkstra will be run on Internal Lsas. Running the Dijkstra indicates a recalculation of the shortest path first tree with respect to the Internal areas and nets within the OSPF area.

`TQSumLsa`

Indicates the Dijkstra will be run on Summary Lsas. Running the Dijkstra indicates a recalculation of the shortest path first tree with respect to the Summary link information received from Area Border Routers.

`TQAseLsa`

Indicates the Dijkstra will be run on External Lsas. Running the Dijkstra indicates a recalculation of the shortest path first tree with respect to the external link information received from outside the OSPF domain.

`TQAck`

Is used for sending delayed acknowledge messages to Link State Update messages.

`TQRetrans`

Is used to send pending retransmissions of unanswered OSPF packets.

`TQIntLsdbAge`

Indicates next time the Internal Link State Database entries will be aged. The Database checksum is recalculated and the age of the entries is checked.

`TQSumLsdbAge`

Indicates next time the Summary Link State Database entries will be aged. The Database checksum is recalculated and the age of the entries is checked.

`TqAseLsdbAge:`

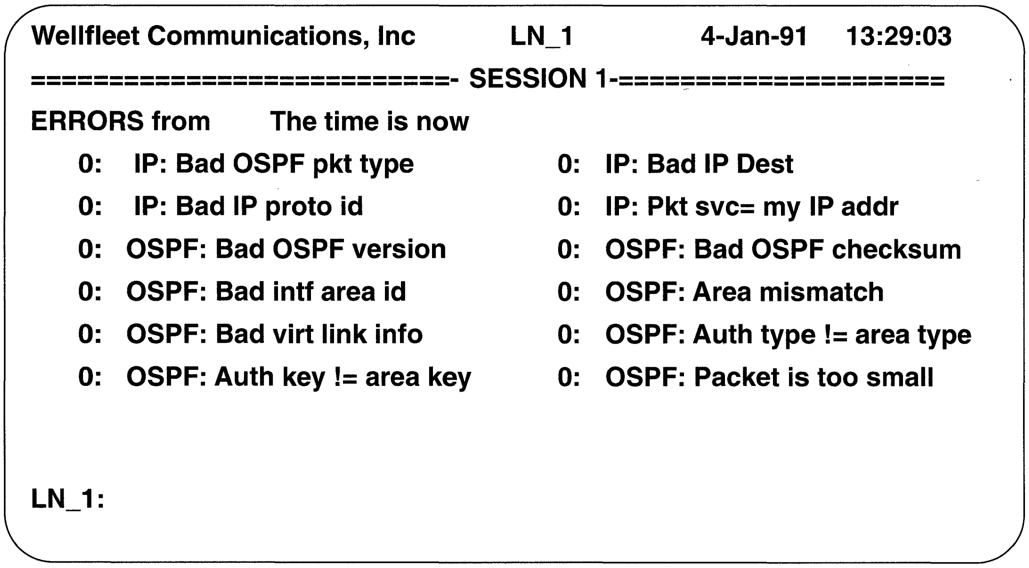
Indicates the next time the External Link State Database entries will be aged. The Database checksum is recalculated and the age of the entries is checked.

`TqLsaLock`

Indicates when a Link State Advertisement could be sent. This timer is set to the minimum allowable value between successive Link State Advertisement messages.

7.6 Displaying the OSPF Error Counts

The **OSPF ERRS** command displays the number of errors accrued by OSPF. Figure 7-6 shows a sample screen of output from this command, and Table 7-1 lists the possible OPSF errors.



```
Wellfleet Communications, Inc          LN_1          4-Jan-91    13:29:03
===========================- SESSION 1-=====================
ERRORS from       The time is now
    0:  IP: Bad OSPF pkt type              0:  IP: Bad IP Dest
    0:  IP: Bad IP proto id                0:  IP: Pkt svc= my IP addr
    0:  OSPF: Bad OSPF version             0:  OSPF: Bad OSPF checksum
    0:  OSPF: Bad intf area id             0:  OSPF: Area mismatch
    0:  OSPF: Bad virt link info           0:  OSPF: Auth type != area type
    0:  OSPF: Auth key != area key         0:  OSPF: Packet is too small



LN_1:
```

Figure 7-6 Sample Output from `OSPF ERRS` Command

Syntax: `OSPF ERRS`↵

where:

`Errors from`

Is the number of errors accrued by OSPF, followed by a colon, and then the type and name of the error.

A subset of the errors listed in Table 7-1 will appear as event messages in the event log. For detailed descriptions of their meanings and any associated actions, refer to the chapter entitled *Using the Event Log* in the second volume of this manual.

Table 7-1 OSPF Errors Returned by `ERRS` Command

OSPF Errors	OSPF Errors (Contd.)
Monitor request	Hello
DB Description	Link-State Req
Link-State Update	Link-State Ack
Monitor response	Hello
DB Description	Link-State Req
Link-State Update	Link-State Ack
IP: Bad OSPF pkt type	IP: Bad IP Dest
IP: Bad IP proto id	IP: Pkt src = my IP addr
OSPF: Bad OSPF version	OSPF: Bad OSPF checksum
OSPF: Bad intf area id	OSPF: Area mismatch
OSPF: Bad virt link info	OSPF: Auth type != area type
OSPF: Auth key != area key	OSPF: Packet is too small
OSPF: Packet size > IP length	OSPF: Transmit bad
OSPF: Received on down IF	Hello: IF mask mismatch
Hello: IF hello timer mismatch	Hello: IF dead timer mismatch
Hello: Extern option mismatch	Hello: Unknown Virt nbr
Hello: Unknown NBMA nbr	DD: Unknown nbr
DD: Nbr state low	DD: Nbr's rtr = my rtrid
DD: Extern option mismatch	Ack: Unknown nbr
Ack: Nbr state low	Ls Req: Nbr state low
Ls Req: Unknown nbr	Ls Req: Empty request
LS Req: Bad pkt	LS Update: Nbr state low
Update: Unknown nbr	Ls Update: Newer self-gen LSA
Ls Update: Bad LS chksum	Ls Update: less recent rx
Ls Update: Unknown type	

8 Using the Event Log

The system generates event messages and stores them in the event log. This chapter describes the event log's structure, tells you how to access the log, and explains how to interpret log entries.

8.1 Event Log Structure

The event log is a circular (FIFO) buffer containing up to 100 entries. Each entry is composed of five fields as shown below.

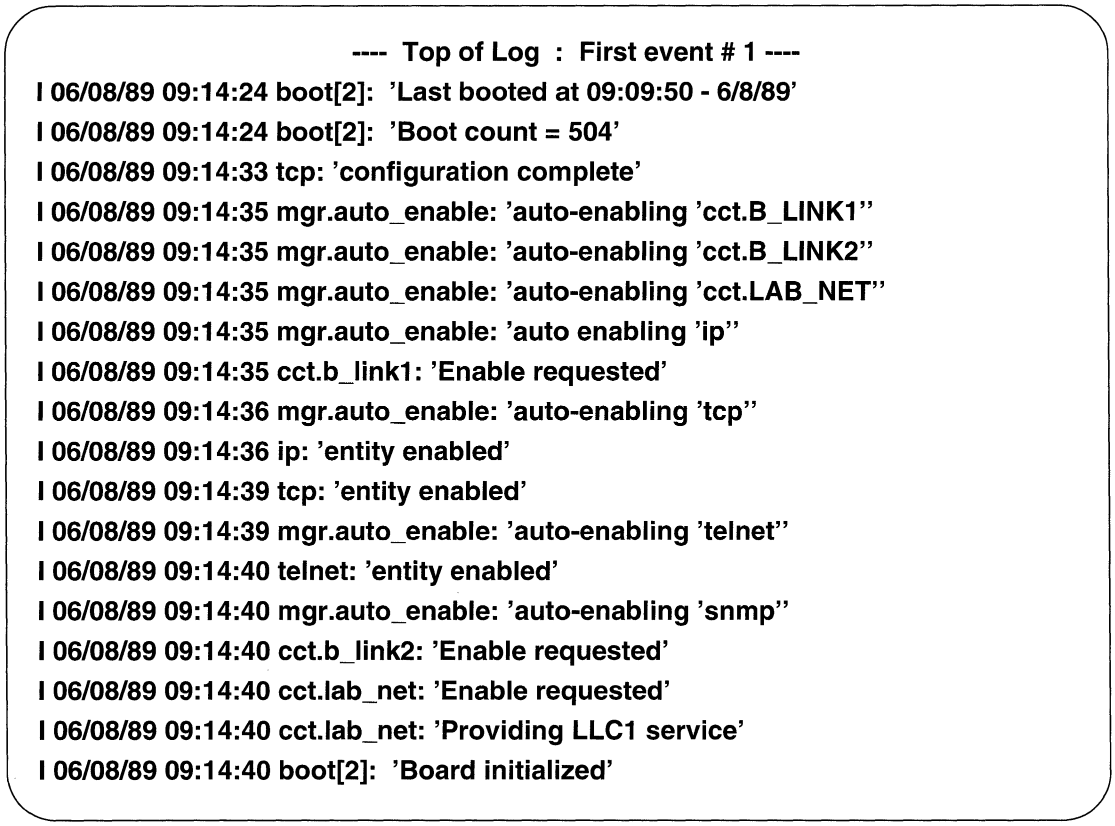
<severity> <date> <time> <object> <event message>

where:

<severity>	contains the level of severity of the event:
F	(fatal) indicates a major system disruption.
M	(major) indicates service appearance/disappearance.
P	(performance) indicates that a service, although still present, has degraded.
W	(warning) indicates that a service has behaved in an unexpected fashion.
I (information)	indicates routine events.
D (debug)	indicates link state information.
<date>	contains the date, in mm/dd/yy format, that the entry was placed in the log.
<time>	contains the time, in hh:mm:ss format, that the entry was placed in the log.
<object>	contains the name and specific instance of the managed object that generated the log entry.
<event message>	contains the managed-object-generated event message.

8.2 Accessing the Event Log

You access the log from the Main Menu. Use the `[UPARROW]` (⇑) or `[DOWNARROW]` (⇓) to position the cursor at **Event Log**, then press `[RETURN]` -- or, you may simply press the `<4>` key. After you press `[RETURN]` or type `<4>`, the console screen displays the log. Figure 8-1 shows a sample event log display.



```
                ---- Top of Log  :  First event # 1 ----
I 06/08/89 09:14:24 boot[2]:  'Last booted at 09:09:50 - 6/8/89'
I 06/08/89 09:14:24 boot[2]:  'Boot count = 504'
I 06/08/89 09:14:33 tcp: 'configuration complete'
I 06/08/89 09:14:35 mgr.auto_enable: 'auto-enabling 'cct.B_LINK1''
I 06/08/89 09:14:35 mgr.auto_enable: 'auto-enabling 'cct.B_LINK2''
I 06/08/89 09:14:35 mgr.auto_enable: 'auto-enabling 'cct.LAB_NET''
I 06/08/89 09:14:35 mgr.auto_enable: 'auto enabling 'ip''
I 06/08/89 09:14:35 cct.b_link1: 'Enable requested'
I 06/08/89 09:14:36 mgr.auto_enable: 'auto-enabling 'tcp''
I 06/08/89 09:14:36 ip: 'entity enabled'
I 06/08/89 09:14:39 tcp: 'entity enabled'
I 06/08/89 09:14:39 mgr.auto_enable: 'auto-enabling 'telnet''
I 06/08/89 09:14:40 telnet: 'entity enabled'
I 06/08/89 09:14:40 mgr.auto_enable: 'auto-enabling 'snmp''
I 06/08/89 09:14:40 cct.b_link2: 'Enable requested'
I 06/08/89 09:14:40 cct.lab_net: 'Enable requested'
I 06/08/89 09:14:40 cct.lab_net: 'Providing LLC1 service'
I 06/08/89 09:14:40 boot[2]:  'Board initialized'
```

Figure 8-1 Event Log Display

The event log header (**Top of Log : First event # n**) precedes the contents of the log. **n** specifies the sequence number of the first displayed log entry. The system assigns a sequence number to each entry as it is placed in the log. When **n** = 1, the event log has not as yet exceeded its capacity. When **n** > 1, the log capacity has been exceeded and the system has begun to overwrite the earliest entries.

Following the header, the console displays the first 19 log entries, and the following prompt:

PRESS: UP, DOWN, LEFT, RIGHT, RETURN

You move through the log as follows:

1. Press the `[UPARROW]` (⇑) or `[DOWNARROW]` (⇓) to scroll through the log one entry (line) at a time.
2. Press `[RETURN]` to scroll to the next screen of entries.
3. Press the `[RIGHTARROW]` (⇒) to move to the end of the log.
4. Press the `[LEFTARROW]` (⇐) to return to the Main Menu.

The event log trailer (**Bottom of Log : Last event # n**) follows the last log entry. In the trailer, **n** designates the sequence number of the last displayed entry.

8.3 Source Route Summary Table

Several event messages refer to the Source Route Summary Table, shown in Table 8-1. This table lists the maximum number of source routing entries that the 2 Mb and 5 Mb ACE boards can support. Refer to this table to determine if you should upgrade the systems on your network with 5 Mb ACE Boards.

Table 8-1 Source Route Summary Table

Table Name	Max. No. Entries (2M ACE)	Max. No. Entries (5M ACE)	Explanation
Rif_table	1K	4K	Contains the RIFs that the system uses to route source routed packets over the network.
Madr_table	4K	16K	Contains the MAC addresses of all of the systems that communicate through the bridge/router.
Sr_is_table	7000	16K	Contains indexes into the Madr_table for source destination pairs of systems. It also contains pointers to the RIF table.
Sr_es_table	1K	4K	Contains indexes into the Madr_table for destination systems that communicate through token ring interfaces (those that are directly connected to the router). It also contains pointers to the RIF table.

9 Event Messages, A to H

This chapter describes the event messages generated by the system and stored by the system in the event log. This chapter contains the messages, arranged alphabetically by managed object, beginning with the letters **a** to **h**. The messages are arranged alphabetically within each section.

9.1 AppleTalk Event Messages

This section provides an alphabetical listing of the event messages generated by the AppleTalk Router. Each message is accompanied by an explanation of the message contents and a recommended action (if any is required).

AARP mapping table is full

Meaning: The AppleTalk Router cannot add an address resolution pair (AppleTalk address and an associated data-link address) to its Address Mapping Table; the table contains its maximum number of entries.

Action: Increase the value of the **AARP Mapping Table** parameter, to expand the size of the table.

AARP PROBE/RSP : node address X.Y

Meaning: The AppleTalk Router (after having issued a *Probe* packet to test the uniqueness of the network number/system identifier pair specified by **X.Y**) has received a *Response* packet indicating that the network number/ system identifier pair is in use by another network system. Consequently, the AppleTalk Router will generate a new address pair, and issue another *Probe*.

AARP probing terminated for <cg_name>

Meaning: The AppleTalk Router cannot obtain a unique network number/system identifier pair for <**cg_name**>. The directly connected medium is supporting the maximum number of systems.

Action: Reconfigure the AppleTalk network with a larger network range.

AARP REQ/RSP : node address X.Y

Meaning: The AppleTalk Router resolved the hardware (data-link) address for the AppleTalk address **X.Y** (where **X** is the network number, and **Y** is the system identifier).

at_amt_alloc: out of memory

Meaning: The AppleTalk Router cannot obtain sufficient memory to allocate the AARP Address Mapping Table.

Action: Adjust the **AARP Mapping Table** parameter downward to place a lesser demand on memory.

at_cg_cb_alloc: out of memory

Meaning: The AppleTalk Router cannot obtain sufficient memory to allocate a circuit group control block.

Action: Reconfigure with smaller memory requirements.

at_routing_tbl_alloc: out of memory

Meaning: The AppleTalk Router cannot obtain sufficient memory to allocate the Routing Table.

Action: Adjust the **Routing Table Size** parameter downward to place a lesser demand on memory.

at_zone_name_tbl_alloc: out of memory

Meaning: The AppleTalk Router cannot obtain sufficient memory to allocate the Zone Name Table.

Action: Adjust the **AARP Mapping Table** and **Routing Table Size** parameters downward to place a lesser demand on memory.

Cfg: cannot get dflt zone name for <cg_name>

Meaning: This message is generated only if **<cg_name>** is a non-seed port. After a non-seed port establishes contact with a seed router, it issues a query to obtain the network's default zone name. This message is generated if the port fails to receive a response to its query.

Action: No action is required. The non-seed port, **<cg_name>**, repeats its query until it obtains the default zone name.

Cfg: cannot get zone name list for <cg_name>

Meaning: This message is generated only if **<cg_name>** is a non-seed port. After a non-seed port obtains the default zone name from a seed router, it issues a query to obtain a list of zone names associated with the network. This message is generated if the port fails to receive a response to its query.

Action: No action is required. The non-seed port, **<cg_name>**, repeats its query until it obtains the list of associated zone names.

Cfg: dflt zone name must be cfg'd for <cg_name>

Meaning: The circuit group specified by **<cg_name>** has been configured as a seed port. No response was entered, however, to the **Dflt Zone Name** parameter.

Action: Configure a default zone name for **<cg_name>**.

Cfg: illegal zone name <zone-name>

Meaning: The **zone-name** that you entered uses an incorrect format of the escaped character syntax. A non-hexadecimal, non-back-slash (\) character was entered as one of the two characters immediately following a back-slash (\) character.

Action: Configure a valid zone name to replace **zone-name**. Refer to the *Configuration Guide.*

Cfg: invld dflt zone name for <cg_name>

Meaning: The circuit group specified by **<cg_name>** has been configured as a seed port. The directly-connected network is serviced not only by this seed, but also by another seed router already in service. The default zone names conveyed by these seed routers are inconsistent.

Action: One of the seed routers must be reconfigured to ensure the consistency of the default zone names. If you reconfigure the AppleTalk Router to match the default zone name of the in-service router, no further changes are necessary. If you reconfigure the in-service router to match the default zone name of the AppleTalk Router, however, you must restart all network systems and routers.

Cfg: invld zone name cfg'd for <cg_name>

Meaning: The circuit group specified by **<cg_name>** has been configured as a seed port. The directly connected network is serviced not only by this seed, but also by another seed router already in service. The zone-name lists conveyed by these seed routers are inconsistent in that while each list contains the same number of zone entries, the lists are not identical.

Action: One of the seed routers must be reconfigured to ensure the consistency of the zone-name lists. If you reconfigure the AppleTalk Router to match the zone-name list of the in-service router, no further changes are necessary. If you reconfigure the in-service router to match the zone-name list of the AppleTalk Router, however, you must restart all network systems and routers.

Cfg: network range incorrect for <cg_name>

Meaning: The circuit group specified by **<cg_name>** has been configured as a seed port. The directly connected network is serviced not only by this seed, but also by another seed router already in service. The network ranges conveyed by these seed routers are inconsistent.

Action: One of the seed routers must be reconfigured to ensure the consistency of the network ranges. If you reconfigure the AppleTalk Router to match the network range of the in-service router, no further changes are necessary. If you reconfigure the in-service router to match the network range of the AppleTalk Router, however, you must restart all network systems and routers.

Cfg: number zone names incorrect for <cg_name>

Meaning: The circuit group specified by **<cg_name>** has been configured as a seed port. The directly connected network is serviced not only by this seed, but also by another seed router already in service. The zone-name lists conveyed by these seed routers are inconsistent in that each list contains a different number of zone entries.

Action: One of the seed routers must be reconfigured to ensure the consistency of the zone name lists. If you reconfigure the AppleTalk Router to match the zone-name list of the in-service router, no further changes are necessary. If you reconfigure the in-service router to match the zone name list of the AppleTalk Router, however, you must restart all network systems and routers.

Cfg: NULL zone name cfg'd for <cg_name>

Meaning: The AppleTalk Router has configured a NULL zone name for **<cg_name>**; the value entered at **Zone Name** was invalid.

Action: Configure a valid zone name for **<cg_name>**.

Cfg: too many zone names cfg'd for <cg_name>

Meaning: More than ten zone names have been added to Zone Name List associated with **<cg_name>**.

Action: No action is required. The AppleTalk Router ignores any zone names beyond the maximum value of ten.

<cg_name> enabled with network range X - Y

Meaning: The circuit group, AppleTalk port, specified by **<cg_name>** is enabled and connected to the attached medium whose range of network numbers is **X** to **Y**.

<cg_name> enabled with node address X.Y

Meaning: The circuit group, AppleTalk port, specified by **<cg_name>** is enabled with the system address **X.Y**, where **X** is the network number, and **Y** is the system identifier.

Circuit Group record not configured for cg# X

Meaning: `config` does not contain a circuit group record for the circuit group identified by **X**, where **X** is the system-assigned interface number that identifies the circuit group.

Action: First, use the NCL `GET` command (in the form `g cct.X`) to obtain the circuit-group name, along with associated statistics. Then verify and, if necessary, reconfigure the circuit group.

Forcing AARP probing for cg <cg_name>

Meaning: An AppleTalk circuit group (specified by **<cg_name>**) was configured with **AARP Probe** disabled. In completing the configuration of this circuit group, however, you did not specify a system identifier and/or you did not select a network number (for seed routers only). Consequently, the AppleTalk Router has generated a quasi-random system identifier and/or network number and is using the Probe facility to ensure the uniqueness of the network number, system identifier pair.

Illegal network number for <cg_name>

Meaning: The network number configured for **<cg_name>** is outside of the valid range specified by the **Network Min** and **Network Max** parameters.

Action: Modify *config*. Reconfigure the network number to within the valid network range.

Illegal network range for <cg_name>

Meaning: Either the **Network Min** or the **Network Max** parameter of **<cg_name>** is outside of the valid range of legal AppleTalk network numbers (1 through 65279).

Action: Modify *config*. Reconfigure the network range parameters to within the valid AppleTalk range.

Incoming invalid zone len <len> - zone <aa bb cc dd>

Meaning: The Wellfleet system has detected an incorrectly formatted zone information packet. The zone length **<len>**, and the first four characters of the zone name (in ASCII hexadecimal format) are **<aa bb cc dd>**.

Action: None. The Wellfleet router will repeat its request for zone names.

Invalid AARP event X, AARP state Y for cg <cg_name>

Meaning: The circuit group specified by **<cg_name>** was in AARP State **Y**, and thus unable to process AARP event **X**. Possible values for **X** and **Y** are shown below:

Event Codes	**(X values)**
0	xmit AARP PROBE
1	xmit AARP REQUEST
2	rcv AARP PROBE
3	rcv AARP REQUEST
4	rcv AARP RESPONSE
5	timer expired/cancelled

AARP State Codes	(Y values)
0	circuit group disabled
1	circuit group xmit (AARP PROBE)
2	circuit group xmit (AARP REQUEST)
3	circuit group enabled

No AppleTalk circuit group configured, slot X

Meaning: You have not configured an AppleTalk circuit group on Slot **X**.

Action: Configure an AppleTalk circuit group on Slot **X**.

No AppleTalk Record configured

Meaning: `config` does not include an AppleTalk Record.

Action: Configure the AppleTalk Router.

No AppleTalk software configured, slot X

Meaning: The portion of `config` that contains AppleTalk configuration data is faulty. One or more of the following required records may be missing: AppleTalk record, configuration summary record, entity records, or boot-load record.

Action: Verify, and if necessary, reconfigure the AppleTalk Router on Slot **X**.

No circuit group recorded for cg number X

Meaning: `config` does not contain a valid circuit-group record for the circuit group identified by **X**, where **X** is the system-assigned interface number that identifies the circuit group.

Action: First, use the NCL **`GET`** command (in the form `g cct.X`) to obtain the circuit-group name, along with associated statistics. Then verify and, if necessary, reconfigure the circuit group.

Rcv'd zone name(s) on slot, X: cannot distribute

Meaning: A list of zone names received by Slot **X** could not be sent to the other slots running AppleTalk.

Zone name table full

Meaning: The AppleTalk Router cannot add a zone name to its Zone Name Table; the table contains its maximum of 256 entries.

Action: Check your network topology. Wellfleet's AppleTalk Router implementation allows a maximum of 256 zones on an internet.

9.2 Boot Event Messages

This section contains an alphabetical list of event messages generated by the *boot* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

Board initialized

Meaning: The ACE board contained in Slot **[#]** has successfully initialized.

Boot count = nnn

Meaning: The system controller board (contained in Slot **1**) has been booted **nnn** times. Note that this count is specific to the system controller board, not to the system.

Last booted at hh:mm:ss - mm/dd/yy

Meaning: The system was last booted on **mm/dd/yy** at **hh:mm:ss**.

WARNING: ACE REV nn

Meaning: **nn** is the revision level of the ACE board in Slot **[#]**.

Action: Certain software applications require specific (i. e., more recent) ACE revisions. Check with Customer Support.

9.3 Bridge Event Messages

This section contains an alphabetical list of event messages generated by the *lb* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

Circuit Group xxxxxxx Blocking

Meaning: The spanning tree algorithm has placed circuit group **xxxxxxx** in the **Blocking** State. A circuit group in this state does not participate in frame relay. The spanning tree algorithm, however, does include blocked ports in its calculation of the active topology.

Circuit Group xxxxxxx Disabled

Meaning: The spanning tree algorithm has placed circuit group **xxxxxxx** in the **Disabled** State. A circuit group in this state does not participate in frame relay. The spanning tree algorithm does not include disabled ports in its calculation of the active topology.

Circuit Group xxxxxxx Forwarding

Meaning: The spanning tree algorithm has placed circuit group **xxxxxxx** in the **Forwarding** State. A circuit group in this state is participating in frame relay.

Circuit Group xxxxxxx Learning

Meaning: The spanning tree algorithm has placed circuit group **xxxxxxx** in the **Learning** State. A circuit group in this state is participating in frame relay, and has enabled the learning function.

Circuit Group xxxxxxx Listening

Meaning: The spanning tree algorithm has placed circuit group **xxxxxxx** in the **Listening** State. A circuit group in this state is preparing to participate in frame relay.

entity disabled

Meaning: The Bridge has been disabled in response to an NCL `DIS[able]` command.

entity enabled

Meaning: The Bridge successfully initialized, or has been enabled with the NCL `E[nable]` command.

LB fwd dlay should be >= xxx

Meaning: The **fwd dlay** configuration parameter was misconfigured.

Action: Reconfigure the **fwd dlay** parameter to the value in **xxx**, given the correct, configured value of the **max age** parameter.

LB max age should be >= xxx

Meaning: The **max age** configuration parameter was misconfigured.

Action: Reconfigure the **max age** parameter to the value in **xxx**, given the configured value of the **Hello Time** parameter.

No Bridge Circuit Group configured, slot

Meaning: The Bridge record in the configuration file contains an improperly configured circuit group.

Action: Check Bridge circuit groups. Refer to the *Configuration Guide.*

No Bridge Record configured

Meaning: The Bridge software has not been configured.

Action: Configure the Bridge software. Refer to the *Configuration Guide.*

No Bridge Software configured, slot

Meaning: The Bridge software has not been loaded into **slot #**.

Action: Modify `config` to include a Bridge record for **slot #**. Refer to the *Configuration Guide.*

SR internal LAN ID not in RIF route

Meaning: A Specifically Routed Frame (SRF) was received that did not include the Internal LAN ID of the Wellfleet bridge. The frame cannot be forwarded and must be dropped.

SR is_srf_rif_insert: no rif entry

Meaning: A Specifically Routed Frame (SRF) was received, however, the appropriate entry in the Source Routing Intermediate Station Table (**Sr_is_table**) does not contain a Routing Information Field (RIF) for the destination Station. The packet is dropped.

SR max hops exceeded in explorer frame

Meaning: The maximum number of hops was exceeded in an All Routes Explore (ARE) frame or a Spanning Tree Explorer (STE) frame. The maximum number of hops for a source routed packet is 7. The packet cannot be forwarded and must be dropped.

SR max hops exceeded in Specifically Routed Frames

Meaning: The maximum number of hops was exceeded in a Specifically Routed Frame (SRF). The maximum number of hops for a Source Routed packet is 7. The packet cannot be forwarded and must be dropped.

SR out cg = cg for SRF

Meaning: The Specifically Routed Frame (SRF) is being forwarded out the same interface on which it was received. This can occur if the Loop Detection Time is set too low, or if the network is reconfiguring (due to a link failure), or if the Wellfleet system is rebooted.

SR out of buffers

Meaning: An attempt was made to allocate a packet buffer to flood an All Routes Explorer (ARE) packet out of a particular interface, however, no packet buffer was available. The ARE cannot be flooded out of the interface.

SR possible ARE loop

Meaning: A possible All Routes Explorer (ARE) loop has been detected. This can occur if the Loop Detection Time is set too low, or if the network is reconfiguring (due to a link failure), or if the Wellfleet system is rebooted.

Action: Check the Loop Detection Time parameter in `config.` The Loop Detection Time may be set too low to detect an ARE loop and may need to be increased. Refer to the *Configuration Guide*.

SR Rif_table out of space

Meaning: The Routing Information Field (RIF) table (**Rif_table**) is out of space. The RIF Table contains RIFs that are used to route source routed packets between Wellfleet systems and remote hosts over token ring networks. The RIF table contains RIFs used for both End Station (ES) Source Routing over IP and Intermediate Station (IS) source routing over the bridge.

Action: A memory upgrade may be required. For a description of the size of the **Rif_table**, refer to Table 8-1.

SR sr_is_find: Madr_table out of space

Meaning: The MAC Address Table (**Madr_table**) is out of space. The **Madr_table** is a table that contains MAC addresses that are used for both End Station (ES) source routing for IP as well as Intermediate Station (IS) source routing for the bridge.

Action: A memory upgrade may be required. For a description of the size of the **Madr_table**, refer to Table 8-1.

SR sr_is_table: out cct's cg is 0, sending ARE

Meaning: The Specifically Routed Frame (SRF) was received and the appropriate entry in the Source Route Intermediate Station Table (**Sr_is_table**) for the source destination pair does not yet include the route to the destination station. The Wellfleet bridge will now send an All Routes Explorer (ARE) packet in order to discover the route to the destination station. (The route to the destination station is discovered when a response packet is received from the destination station.) This situation can occur if the network is re-configuring (due to a link failure), or if the Wellfleet system was rebooted.

SR sr_is_table out of space

Meaning: The source routing Intermediate Station table is out of space. The **Sr_is_table** is a table that contains source-destination pairs of MAC addresses. It contains pointers to the RIF table.

Action: A memory upgrade may be required. For a description of the size of the **Sr_is_table**, refer to Table 8-1.

SR sr_ring full

Meaning: The **sr_ring** data structure used to contain outgoing source routed packets is full. No other source routed packets will be sent out until the ring has been emptied.

SR TF forward_ring full

Meaning: The **forward _ring** data structure is used to contain outgoing source routed packets that have been filtered through traffic filters. This data structure is now full. No other source routed packets that need to be filtered through traffic filters will be sent out until the ring has been emptied.

SRT out of buffers

Meaning: The Source Routing Transparent bridge cannot allocate a packet buffer when processing an explorer packet(s).

9.4 Circuit Event Messages

This section contains a list of event messages generated by the *cct* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required). The messages are divided into the following categories:

- FDDI VME event messages
- Frame Relay event messages
- General circuit event messages
- LAN event messages
- LLC1 event messages
- LLC2 event messages
- Point-to-Point Protocol event messages
- Token Ring event messages

Within each section, the messages are arranged alphabetically.

9.4.1 FDDI VME Event Messages

This section contains an alphabetical list of event messages generated by the *cct* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

Driver[n]: FDDI download complete

Meaning: Software that controls the FDDI board has been successfully loaded onto the FDDI board.

Driver[n]: Slot 2 downloading FDDI board[x (short nnnn)]

Meaning: The software detects the presence of FDDI hardware and has started to download software to the FDDI board (**x** is the FDDI board number (1, 2, or 3); the VME short address is in 4-digit form **nnnn**).

Driver[n]: Slot n hosting FDDI board[x (short nnnn)]

Meaning: The driver software managing the FDDI board detects the presence of the FDDI board (**x** is the FDDI board number (1, 2, or 3); the VME short address is in four digit form **nnnn**).

Driver[n]:V4211-FDDI[FDDI Firmware Rev, Date, Time]

Meaning: This is the identity and release stamp of the software loaded onto the FDDI board.

9.4.2 Frame Relay Circuit Event Messages

This section contains an alphabetical list of event messages generated by the *frame relay* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

<cct_name> - Attempt to init FR on inactive cct

Meaning: The frame relay circuit is not active, yet there was an attempt to initialize the circuit. This is an internal problem and cannot be corrected by the user. **<cct_name>** identifies the frame relay circuit.

<cct_name> - Bad address option combination

Meaning: The selected combination of addressing encoding and address length is not compatible. Allowable combinations are:

Q921 -	TWO BYTE only
Q922 March90 -	TWO BYTE only
Q922 November 90 -	TWO BYTE
	THREE BYTE
	FOUR BYTE
Q922 -	TWO BYTE
	TWO + CONTROL
	THREE BYTE
	THREE + CONTROL
	FOUR BYTE

<cct_name> identifies the frame relay circuit for which this error occurred.

Action: Modify the configuration so that the address encoding and address length are compatible. Refer to the *Configuration Guide* .

<cct_name> - Bad full polling; reset to 6

Meaning: The chosen **interval between full polling** parameter is not within the proper range. The proper range is between 1 and 255 polls. The default value is to send a full status enquiry every 6 polling cycles. **<cct_name>** identifies the frame relay circuit for which the change was made.

Action: Modify the configuration so that the polling interval is within the proper range. Refer to the *Configuration Guide.*

<cct_name> - Bad interface discriminator found <xxx>

Meaning: A packet was received on the management interface Data Link Connection Identifier (DLCI), but the protocol discriminator field did not contain the proper value for the selected management interface method. Proper values are 9 for LMI and 8 for ANSI Annex D. **<cct_name>** identifies the frame relay circuit which received the erroneous frame and **<xxx>** identifies the protocol discriminator that was found.

<cct_name> - Bad poll interval; reset to 10

Meaning: The polling interval chosen during configuration is not within the valid range of 5 to 30 seconds. **<cct_name>** identifies the erroneous frame relay circuit.

Action: Modify the configuration so that the polling interval is within the proper range. Refer to the *Configuration Guide* .

<cct_name> - Bridge mapping table is full

Meaning: The bridge table mapping destination MAC addresses to frame relay DLCIs is full and can not accommodate a new entry. This table is tied to the internal bridging tables and therefore, my be indicative of a bridging problem. **<cct_name>** identifies the frame relay circuit.

<cct_name> - DLCI <xxx> not within range; not added

Meaning: A dlci value was added that was not within the range of valid values for the specified type and length. **<cct_name>** identifies the frame relay circuit, and **<xxx>** identifies the dlci in error.

<cct_name> - DLCI too large for interface; <xxx>

Meaning: DLCI values exceeding that which could be fit into the length specified during configuration. Primarily this will happen if there is no management interface selected and PVCs are hand configured with large DLCIs. **<cct_name>** identifies the frame relay circuit and **<xxx>** identifies the DLCI which is too large.

<cct_name> - Excessive errors; interface disabled

Meaning: When the DTE receives a designated number of errors within a designated number of events, the interface is considered unstable and is taken down. This message indicates that this situation has occurred. **<cct_name>** identifies the frame relay circuit.

<cct_name> - Invalid message type <xxx> found

Meaning: A management message was received with an unknown or invalid message type. Valid values are:

0x7D -	Status
0x75 -	Status Enquiry
0x7B -	Status Update

<cct_name> identifies the frame relay circuit and **<xxx>** identifies the message type that was found.

<cct_name> - Invalid shift parameter <xxx>

Meaning: A management packet was received with an unrecognizable shifting parameter. This applies only to those interfaces running ANSI Annex D. All information elements must be encoded using locking shift 5. Any other shifting parameters will be flagged as an error and data following will be discarded.

<cct_name> - Invalid STATUS message received

Meaning: The status message returned from the frame relay switch is not properly formatted. That is, the report type and keep-alive sequence exchange messages were not found in the proper order. **<cct_name>** identifies the frame relay circuit.

<cct_name> - Keep-alive recovery (from net)

Meaning: The Wellfleet router is now properly receiving the network keep-alive sequence number for circuit name **<cct_name>**.

<cct_name> - Management Interface connection established

Meaning: The network and the DTE have established the connection necessary for periodic polling and status message exchange on circuit **<cct_name>**.

<cct_name> - Net not receiving keep-alive seq

Meaning: The frame relay switch is returning last received sequence numbers that do not correspond to the numbers that the Wellfleet router is actually sending. **<cct_name>** identifies the frame relay circuit.

<cct_name> - Net sequence num receive recovery

Meaning: The network is now properly sending keep-alive sequence numbers for the circuit **<cct_name>**.

<cct_name> - New DLCI <xxx> added over existing one

Meaning: A PVC was marked as new in the PVC status message before it was removed via the management interface. The old PVC is deleted and this new one is put in its place. **<cct_name>** identifies the frame relay circuit.

<cct_name> - Not receiving seq num on MI enquiry

Meaning: The other side of the frame relay interface is issuing status enquiry messages with last received keep-alive sequence numbers that are not as expected. That is, this sequence number is not the last sequence number we sent. This indicates that the other side is not receiving our status message response. **<cct_name>** identifies the frame relay circuit.

<cct_name> - Out of sequence keep alive (net)

Meaning: The Wellfleet router is not properly receiving the network keep-alive sequence number for circuit name **<cct_name>**. Usually this means the switch has reset its sequence counter indicating it is experiencing difficulties.

<cct_name> - PVC <xxx> added - Active

Meaning: A PVC has been added in the active state. This is due to a full status message or an update status message from the management interface. **<xxx>** is the DLCI associated with the newly added PVC. **<cct_name>** identifies the frame relay circuit.

<cct_name> - PVC <xxx> added - Inactive

Meaning: A PVC has been added in the inactive state. This is due to a full status message or an update status message from the management interface. Usually this indicates that the station at the other side of this connection is not active. **<xxx>** is the DLCI associated with the newly added PVC. **<cct_name>** identifies the frame relay circuit.

<cct_name> - PVC <xxx> deleted

Meaning: The PVC indicated by the given DLCI **<xxx>** has been deleted. Deletion may occur because a PVC status IE was not present in the full status message or because it has been explicitly deleted. **<cct_name>** identifies the frame relay circuit.

<cct_name> - PVC <xxx> status change to Active

Meaning: A PVC has changed state to active. This is due to a full status message or update status message from the management interface. **<xxx>** is the DLCI associated with the PVC for which the change has occurred. **<cct_name>** identifies the frame relay circuit.

<cct_name> - PVC <xxx> status change to Inactive

Meaning: A PVC has changed state to inactive. This is due to a full status message or update status message from the management interface. **<xxx>** is the DLCI associated with the PVC for which the change has occurred. **<cct_name>** identifies the frame relay circuit.

<cct_name> - PVC IE out of order for dlci <xxx>

Meaning: A full status message was received for which the PVC status information elements were not in ascending order by DLCI. **<cct_name>** identifies the frame relay circuit and **<xxx>** identifies the first out of order PVC information element.

<cct_name> - Status msg not received within timeout

Meaning: The status message expected in response to a status enquiry message was not received within the timeout set within the configuration. **<cct_name>** identifies the frame relay circuit.

Action: No action is required, yet some may be desirable. If the interface is very busy it may be desirable to increase the timeout to allow the switch to respond within the period specified. It may also be desirable to configure the polling interval to request updates less frequently. Refer to the *Configuration Guide* .

<cct_name> - Status msg polling recovery

Meaning: The network is now responding to the status enquiry messages within the given timeout period. After three consecutive status/enquiry-status message exchanges, this message is displayed to indicate the system is in sync.

<cct_name> - Unable to perform update for dlci <xxx>

Meaning: An operator tried to modify a dlci but the system was unable to perform the requested modification. Possible problems are adding a dlci that has already been added, or deleting a dlci that isn't present. **<xxx>** identifies the circuit for which the request was made.

<cct_name> - Unsupported IE value <xxx> found

Meaning: A valid status update or full status message was received with an unrecognized or unsupported Information Element (IE). **<cct_name>** identifies the frame relay circuit and **<xxx>** identifies Information Element identifier code in question.

<cct_name> - Unsupported NLPID found <xxx>

Meaning: A received data packet that included a NLPID value that was not recognized. The NLPID identifies the encapsulated data type. Packets received with unknown NLPIDs are discarded. **<cct_name>** identifies the frame relay circuit and **<xxx>** identifies the NLPID value received.

<cct_name> - Wrong sequence number on MI enquiry

Meaning: The other side of the frame relay interface is issuing status enquiry messages with keep-alive sequence numbers that are not as expected. That is, this sequence number is more than one greater than the last one we received. **<cct_name>** identifies the frame relay circuit.

Error in Configuration record

Meaning: The established configuration for the frame relay circuit is in error.

Action: Modify the configuration to establish a valid frame relay circuit. Refer to the *Configuration Guide*.

Too many circuits configured for slot <xxx>

Meaning: The configuration contains more than four frame relay interfaces for a single slot. The slot number of the error is included as **<xxx>**.

Action: Modify the configuration so that there are fewer frame relay interfaces for slot **<xxx>**. Refer to the *Configuration Guide*.

<xx> - Invalid QOS. Setting to LLC1

Meaning: The Quality of Service parameter for cct **<xx>** was not configured as LLC1 (unreliable datagram). Presently LLC1 is the only quality of service supported for frame relay. Therefore, the Quality of Service parameter has been automatically modified to be LLC1.

Action: Modify config so that the Quality of Service parameter is LLC1. Refer to the *Configuration Guide*.

9.4.3 General Circuit Event Messages

This section contains an alphabetical list of general event messages generated by the *cct* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

Bad QofS

Meaning: `config` contains a mismatch between the **Quality of Service** parameter and the **Circuit Type** parameter.

Action: Modify `config` to ensure that the quality of service is appropriate for the circuit type. Refer to the *Configuration Guide.*

Enable requested

Meaning: The circuit named in the **<object>** field is requesting to be enabled.

entity already disabled

Meaning: An already disabled circuit, identified in the **<object>** field, has received an `NCL DIS[able]` command.

entity already enabled

Meaning: An already enabled circuit, identified in the **<object>** field, has received an `NCL E[nable]` command.

entity disabled

Meaning: The circuit named in the **<object>** field has been disabled in response to an NCL `DIS[able]` command.

entity enabled

Meaning: The circuit named in the **<object>** field successfully initialized, or has been enabled with the NCL `E[nable]` command.

Invalid MFS, dflt=2

Meaning: The configuration record for **DEV CCT: <cct_name>** contains a faulty value in the **Minimum Frame Spacing** field. The system has defaulted to a value of 2.

Action: Modify **Minimum Frame Spacing** in `config`. Refer to the *Configuration Guide.*

Memory error (MERR) detected

Meaning: The Local Area Network Controller or Link Level Controller cannot access the system memory. The circuit is effectively disabled.

Action: Verify hardware integrity.

No buffers to reconnect to remote

Meaning: The circuit identified in the **<object>** field cannot establish a connection because of insufficient buffer space.

Physical level error

Meaning: The Local Area Network Controller or Link Level Controller has detected a Level 1 (physical media) error.

Action: Verify hardware integrity.

Receiver overflow detected

Meaning: The Local Area Network Controller, or the Link Level Controller for the circuit identified in the **<object>** field has dropped a received packet because of lack of space in the Receiver FIFO buffer.

Too many T1 circuits configured for module

Meaning: `config` contains an excessive number (greater than two) of T1 circuit records for this slot.

Action: Modify `config` to ensure that no more than two T1 circuit records are established for any slot. Refer to the *Configuration Guide.*

Too many V35 circuits configured for slot

Meaning: `config` contains an excessive number of V35 line records for this slot.

Action: Modify `config.` Refer to the *Configuration Guide.*

Transmit underflow detected

Meaning: The Link Level Controller has truncated packet transmission because of an interruption in the flow of data from memory.

V35 circuit record missing

Meaning: A circuit (identified by **DEV CCT: <cct_name>**) has not been configured.

Action: Modify `config` to ensure that it includes circuit records for **<cct_name>**. Refer to the *Configuration Guide.*

9.4.4 LAN Event Messages

This section contains an alphabetical list of event messages generated by the LAN *cct* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

Excessive collisions

Meaning: The LAN circuit named in the **<object>** field has dropped a frame after it has detected collisions on 16 successive transmission attempts.

Action: If message occurs frequently, investigate the causes of LAN congestion.

Physical level error

Meaning: The Local Area Network Controller has detected a Level 1 (physical media) error.

Action: Verify hardware integrity.

Receiver overflow detected

Meaning: The Local Area Network Controller for the circuit identified in the **<object>** field has dropped a received packet because of lack of space in the Receiver FIFO buffer.

SQE absent (non 802.3 XCVR)

Meaning: The circuit named in the **<object>** field has detected a loss of the Signal Quality Error (SQE) signal.

Action: Check transceiver hardware. Note that SQE is not supported by Ethernet (non-802.3) transceivers.

Transceiver signal loss

Meaning: The Local Area Network Controller cannot communicate with the Ethernet transceiver.

Action: Verify the integrity of the transceiver.

9.4.5 LLC1 Event Messages

This section contains an alphabetical list of event messages generated by the LLC1 *cct* managed object. LLC1 is connectionless service. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

Providing LLC1 service

Meaning:	The circuit named in the **<object>** field is enabled and providing LLC1 service.

9.4.6 LLC2 Event Messages

This section contains an alphabetical list of event messages generated by the LLC2 *cct* managed object. LLC2 is connection-mode service. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

Circuit reset complete

Meaning:	The circuit identified in the **<object>** field has completed resetting.

Disabled, LLC2 retries exhausted

Meaning:	The circuit identified in the **<object>** field has been disabled because the system was unable to obtain positive acknowledgment of an outstanding frame after a sequence of retransmission attempts. The number of such attempts is governed by the **Retry Counter (N2)**, **Retry Timer (T1)**, and **Connect Retries** LLC2 parameters.
Action:	Verify point-to-point connectivity.

Invalid N2, dflt=16

Meaning:	The configuration record for **DEV CCT: <cct_name>** contains a faulty value in the **Retry Counter (N2**) field. The system has defaulted to a value of 16.
Action:	Modify **Retry Counter (N2)** in `config`. Refer to the *Configuration Guide.*

Local disconnected remote

Meaning: The system has disconnected the point-to-point link after first sending a **DISCONNECT** request to the remote end and receiving an unnumbered acknowledgment.

Local LLC reset remote

Meaning: The system has sent a **RESET** request to the remote end of the point-to-point link.

Local not hearing from remote

Meaning: A disparity exists between the ends of a point-to-point circuit. The circuit's remote signal and sense testing require that both ends of the circuit be designated as active. This message is generated by the active side of a mismatched pair.

Action: Modify *`config`* to ensure that both **Remote signal & sense** parameters are set to **Active**.

Local reset of remote succeeded

Meaning: The system has received a **RESET** indication in response to an outstanding **RESET** request.

Providing LLC2 service to remote

Meaning: The circuit named in the **<object>** field is enabled and providing LLC2 service.

Remote clearing

Meaning: The remote end of a point-to-point circuit is in the process of resetting.

Remote connection refused by local

Meaning: The system has rejected a **CALL** request from the remote end.

Remote disconnect after local FRMR

Meaning: The remote end of a point-to-point circuit has disconnected after receiving a **FRAME REJECT** packet from the local end of the circuit.

Remote disconnect confirmed

Meaning: A local LLC2 circuit has received a positive confirmation of a previously issued disconnect.

Remote disconnected local

Meaning: The system has received (and processed) a **DISCONNECT** request from the remote end.

Remote disconnect received

Meaning: A local LLC2 circuit has issued a disconnect to a remote circuit, and this disconnect has been received. (Formerly this message was **Remote clearing**.)

Remote disconnect retries exhausted

Meaning: A local LLC2 circuit has issued a disconnect to a remote circuit, and the number of retries has been exhausted after n2 times of trying.

Remote disconnect timeout

Meaning: A local LLC2 circuit has issued a disconnect to a remote circuit, and this disconnect has timed out and become idle.

Remote reset to local

Meaning: The system has received a **RESET** request from the remote end.

Responded to reset, service continued

Meaning: The system has received (and processed) a **RESET** request from the remote end. Refer also to **Unexpected remote reset to local** in this section.

Retrying LLC2 connection

Meaning: The circuit identified in the **<object>** field has been unable to obtain positive acknowledgment of an outstanding frame. It will continue to retry the connection as specified by the **Retry Counter (N2)**, **Retry Timer (T1)**, and **Connect Retries** LLC2 parameters.

Unexpected remote reset to local

Meaning: This message occurs at one end of a point-to-point link when the connection is first being established. One end of the link usually comes up before the other end. The first end subsequently receives a **RESET** request from the other end when it comes up. The side that comes up first displays the above message, which always appears in tandem with the **Responded to reset, service continued** message.

9.4.7 Point-to-Point Protocol Event Messages

This section contains an alphabetical list of event messages generated by the *point-to-point protocol* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

<cct_name>: BNCP is down

Meaning: Bridge Network Control Protocol (BNCP) has gone down. **<cct_name>** identifies the PPP circuit.

<cct_name>: BNCP is up

Meaning: Bridge Network Control Protocol (BNCP) is up. Bridge packets may now be sent and received on the link. **<cct_name>** identifies the PPP circuit.

<cct_name>: illegal packet received

Meaning: A packet was received which did not conform to standard PPP format. The packet was dropped. No further action is necessary. **<cct_name>** identifies the PPP circuit.

<cct_name>: IPCP is down

Meaning: IP Control Protocol (IPCP) has gone down. **<cct_name>** identifies the PPP circuit.

<cct_name>: IPCP is up

Meaning: IP Control Protocol (IPCP) is up. IP packets may now be sent and received on the link. **<cct_name>** identifies the PPP circuit.

<cct_name>: LCP is being restarted

Meaning: Link Control Protocol (LCP) went down and the system is restarting LCP because the LCP Auto-Restart option was configured as **yes**. **<cct_name>** identifies the PPP circuit.

<cct_name>: LCP is down

Meaning: Link Control Protocol (LCP) has gone down. If the LCP Auto-Restart option was configured as **yes**, then the system attempts to restart LCP. **<cct_name>** identifies the PPP circuit.

<cct_name>: LCP is up

Meaning: Link Control Protocol (LCP) is up. The system now attempts to bring up the appropriate network layer control protocols. **<cct_name>** identifies the PPP circuit.

<cct_name>: LQM negotiation rejected by remote station

Meaning: The remote station refuses to accept negotiation of the Link Quality Monitor (LQM) parameter. The system cannot bring up the Link Control Protocol (LCP) until the remote station is willing to accept negotiation of the LQM parameter. **<cct_name>** identifies the PPP circuit.

Action: If it is not considered necessary to receive Link Quality Report (LQR) packets in order to monitor link quality, then modify config so that the LQM Time parameter is set to 0. If the LQM Time parameter is set to 0, then the system will not require the peer (remote) station to send LQR packets. Refer to the *Configuration Guide.*

<cct_name>: Max Pkt Size adjusted down to 1500

Meaning: The Max Pkt Size parameter exceeded the maximum value allowed of 1500 bytes. TheMax Pkt Size parameter has been automatically adjusted down to 1500 bytes. **<cct_name>** identifies the PPP circuit.

Action: Modify config so that the Max Pkt Size parameter is within the legal range. Refer to the *Configuration Guide.*

<cct_name>: missed receiving <xx> LQRs: link is down

Meaning: **<xx>** Link Quality Monitor (LQM) time periods elapsed without a Link Quality Report (LQR) packet being received from the peer (remote) station. The link is declared down, the Link Control Protocol (LCP) is brought down, and all Network Control Protocols are also brought down. If the **LCP Auto-Restart** option is configured as **yes**, then the system attempts to bring LCP up again. **<cct_name>** identifies the PPP circuit.

<cct_name>: possible loop-back has been detected

Meaning: PPP detected a possible loop-back condition. A loop-back may occur during the initialization of the 802.1 Spanning Tree Protocol. **<cct_name>** identifies the PPP circuit.

<cct_name>: protocol 0x<yy> not supported

Meaning: The peer (remote) station sent a packet with a PPP protocol value of **<yy>** (hex), but the system does not support PPP protocol **<yy>**. **<cct_name>** identifies the PPP circuit.

<cct_name>: remote station has logged in to Server

Meaning: The remote station has successfully logged in to the system. **<cct_name>** identifies the PPP circuit

<cct_name>: remote station rejected BNCP

Meaning: The remote station has rejected the Bridge Network Control Protocol (BNCP). No bridge traffic may occur over the link until the remote station is ready to accept the Bridge Network Control Protocol. **<cct_name>** identifies the PPP circuit.

<cct_name>: remote station rejected IPCP

Meaning: The remote station has rejected the IP Control Protocol (IPCP). No IP traffic may occur over the link until the remote station is ready to accept the IP Control Protocol. **<cct_name>** identifies the PPP circuit.

<cct_name>: remote station rejected LCP

Meaning: The remote station has rejected the Link Control Protocol (LCP). Link initialization cannot continue until the remote station is ready to accept the Link Control Protocol. **<cct_name>** identifies the PPP circuit.

<cct_name>: remote station rejected UPAP

Meaning: The remote station has rejected the User/Password Authentication Protocol (UPAP). Link initialization cannot continue until the remote station is ready to accept the User/Password Authentication Protocol. **<cct_name>** identifies the PPP circuit.

Action: If UPAP is not considered necessary, then modify config to disable UPAP. Refer to the *Configuration Guide* .

<cct_name>: remote station's login attempt failed

Meaning: The remote station failed in its attempt to login to the system. The remote station's User ID or Password (or both) were incorrect.**<cct_name>** identifies the PPP circuit.

Action: The remote station's User ID and/or Password in config may be modified (if desired) so that the remote station may successfully login to the system. Refer to the *Configuration Guide* .

<cct_name>: Server has logged in to remote station

Meaning: The system successfully logged in to the remote station. **<cct_name>** identifies the PPP circuit.

<cct_name> Server's login attempt failed

Meaning: The system failed in its attempt to login to the remote station. The system User ID or the System Password (or both) were incorrect. **<cct_name>** identifies the PPP circuit.

Action: Modify system User ID and/or System Password in config. Refer to the *Configuration Guide* .

<cct_name>: remote station's LQM time > configured time

Meaning: The Link Quality Monitor (LQM) time which the remote station is willing to negotiate for is greater than the LQM time configured for the PPP circuit. The higher LQM time is accepted, but it means that the remote station will be sending Link Quality Report packets less often than the system originally requested. **<cct_name>** identifies the PPP circuit.

<cct_name>: too many packets lost: link unreliable

Meaning: The number of packets lost (either by the system or by the remote station) exceeded the allowed number of lost packets as determined via the Desired Link Quality parameter. All Network Control Protocols will be brought down by the system. The system will continue to send and receive Link Quality Report (LQR) packets. When desired link quality is re-established, the Network Control Protocols will be brought up again. **<cct_name>** identifies the PPP circuit.

<cct_name>: too many bytes lost: link unreliable

Meaning: The number of bytes lost (either by the system or by the remote station) exceeded the allowed number of lost bytes as determined via the Desired Link Quality parameter. All Network Control Protocols will be brought down by the system. The system will continue to send and receive Link Quality Report (LQR) packets. When desired link quality is re-established, the Network Control Protocols will be brought up again. **<cct_name>** identifies the PPP circuit.

ppp: bad configuration file

Meaning: PPP detected an inconsistency in the configuration.

Action: Modify `config`. Refer to the *Configuration Guide.*

9.4.8 Token Ring Event Messages

This section provides an alphabetical listing of the event messages generated by the *token ring* managed object. Each messages is accompanied by an explanation of the message contents and a recommended action (if any is required).

Can't configure 4 Mbps token interface

Meaning: You have configured 4Mbps service on a 16Mbps medium.

Action: Reconfigure the **Ring Interface** parameter setting the service properly.

Can't configure 16 Mbps token interface

Meaning: You have configured 16Mbps service on a 4Mbps medium.

Action: Reconfigure the **Ring Interface** parameter setting the service properly.

Token cable connection fault

Meaning: The Token Ring interface has detected a faulty cable connection.

Action: Check for a loose or disconnected cable; verify hardware integrity.

9.5 DECnet Phase IV Event Messages

This section contains an alphabetical list of event messages generated by the *drs* (DECnet routing service) managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

Adj Down CG <xxxxxxx>, Bad Pkt, Adj=aa.nnnn

Meaning: An adjacent system (accessible through circuit group **xxxxxxx**), whose Area and Node Address is **aa.nnnn**, has been declared DOWN because the system transmitted an erroneous packet.

Adj Down CG <xxxxxxx>, Chksum error, Adj=aa.nnnn

Meaning: An adjacent system (accessible through circuit group **xxxxxxx**), whose Area and Node Address is **aa.nnnn**, has been declared **DOWN** because the system transmitted a routing topology packet which contained an invalid checksum.

Adj Down CG <xxxxxxx>, Dropped, Adj=aa.nnnn

Meaning: An adjacent system (accessible through circuit group **xxxxxxx**), whose Area and Node Address is **aa.nnnn**, has been declared **DOWN** because the system transmitted a faulty (mis-addressed) hello packet.

Adj Down CG <xxxxxxx>, Out of range, Adj=aa.nnnn

Meaning: An adjacent system (accessible through circuit group **xxxxxxx**), whose Area and Node Address is **aa.nnnn**, has been declared **DOWN** because the system's Area and/or Node Address exceeds the values set by the **Max. Area** and/or **Max Nodes** parameters.

Action: Modify `config` to increase the values of **Max. Area** and/or **Max Nodes**. Refer to the *Configuration Guide.*

Adj Down CG <xxxxxxx>, Router Table Full, Adj=aa.nnnn

Meaning: An adjacent system (accessible through circuit group **xxxxxxx**), whose Area and Node Address is **aa.nnnn**, has been declared **DOWN**. The system information has been deleted from the current Adjacent Router Table.

Adj Down CG <xxxxxxx>, Sync lost, Adj=aa.nnnn

Meaning: The circuit group manager has declared circuit group **xxxxxxx** (which accesses Node **aa.nnnn**) to be disabled. Consequently, the DECnet Router declares **aa.nnnn** DOWN.

Adj Down CG <xxxxxxx>, Timeout, Adj=aa.nnnn

Meaning: An adjacent system (accessible through circuit group **xxxxxxx**), whose Area and Node Address is **aa.nnnn**, has been declared **DOWN** because the DECnet router has failed to receive three consecutive **hello** packets from this system.

Adj Down CG <xxxxxxx>, Version Skew, Adj=aa.nnnn

Meaning: An adjacent system (accessible through circuit group **xxxxxxx**), whose Area and Node Address is **aa.nnnn**, has been declared **DOWN** because the system's DECnet routing software predates Version 2.0.0.

Adj Rej CG <xxxxxxx>, node=aa.nnnn, Endnode Table Full

Meaning: An adjacent system (accessible through circuit group **xxxxxxx**), whose Area and Node Address is **aa.nnnn**, has been declared **DOWN**. The system information could not fit in the current Adjacent Endnode Table.

Adj Rej CG <xxxxxxx>, node=aa.nnnn, Router Table Full

Meaning: An adjacent system (accessible through circuit group **xxxxxxx**), whose Area and Node Address is **aa.nnnn**, has been declared **DOWN**. The system information could not fit in the current Adjacent Router Table.

Adj Up CG <xxxxxxx>, Adj=aa.nnnn

Meaning: The adjacent system whose Area and Node Address is **aa.nnnn** has been declared **UP**.

Area Reach Chg Area aa, Reachable

Meaning: The previously unreachable area, whose Area Address is **aa**, has become reachable.

Area Reach Chg Area ##, Unreachable

Meaning: The previously reachable area, whose Area Address is **aa**, has become unreachable.

CG Down CG <xxxxxxx>, Sync lost, node=aa.nnnn

Meaning: The circuit group manager has declared circuit group **xxxxxxx**, which accesses Node **aa.nnnn**, to be unavailable.

CG Up CG <xxxxxxx>, Adj=aa.nnnn

Meaning: The circuit group manager has declared circuit group **xxxxxxx**, which accesses Node **aa.nnnn**, to be **UP**.

Circuit Group misconfigured

Meaning: The configuration file contains an improperly configured circuit group.

Action: Check circuit and circuit group records in *`config`*. Refer to the *Configuration Guide.*

Circuit misconfigured

Meaning: The configuration file contains an improperly configured circuit.

Action: Check circuit records in *config*. Refer to the *Configuration Guide.*

DECnet Circuit Group misconfigured

Meaning: The DECnet record in the configuration file contains an improperly configured circuit group.

Action: Check DECnet circuit group parameters. Refer to the *Configuration Guide.*

entity disabled

Meaning: DECnet has been disabled in response to an NCL **`DISABLE`** command.

entity enabled

Meaning: DECnet successfully initialized, or has been enabled with the NCL **`ENABLE`** command.

Init Fail CG <xxxxxxx>, Block size small, Ver=nn.nn.nn

Meaning: An adjacent host (accessible over circuit group **xxxxxxx**) failed to complete initialization because of an insufficient configured block size.

No DECnet Record configured

Meaning: The configuration file does not contain a DECnet record.

Action: Modify `config` to include a DECnet configuration record. Refer to the *Configuration Guide.*

Node Reach Chg Node aa.nnnn, Reachable

Meaning: The previously unreachable system, whose Area and Node Address are **aa.nnnn**, has become reachable.

Node Reach Chg Node aa.nnnn, Unreachable

Meaning: The previously-reachable system, whose Area and Node Address are **aa.nnnn**, has become unreachable.

Pkt Fmt Err CG <xxxxxxx>, ffffffffffff, node=aa.nnnn

Meaning: The DECnet Router received a partially correct packet from system **aa.nnnn** over circuit group **xxxxxxx**. **ffffffffffff** is the hexadecimal representation of the first six bytes of the packet.

remote SMDS address invalid for DRS Area <xx>, Node <yy>

Meaning: The SMDS address in the entry in the DECnet Routing Service (DRS) Remote Address Map for DECnet Area <xx>, system <yy> is invalid. An SMDS address is 10 digits in length, and each digit must be in the range 0 to 9.

Action: Modify `config` to correct the SMDS address in the appropriate DECnet Address Map entry. Refer to the *Configuration Guide.*

Routing Pkt CG <xxxxxxx>, Highest=aa.nnnn, Adj=aa.nnnn

Meaning: An adjacent router is configured with an area and/or system number greater than the values for which the router is configured. **Adj=** contains the source address of the packet. **Highest=** contains the faulty address data contained in the packet.

9.6 Driver Event Messages

This section contains an alphabetical list of event messages generated by the *driver* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

ACE REV nn with an ENET II

Meaning:	An Ethernet II link module is connected to an early revision (before revision 14) ACE.
Action:	Install proper ACE board in backplane slot.

Bad module ID

Meaning:	The link module id is unknown to the driver. Link module (**mod_id**) values are listed in Table 9-1.
Action:	Verify proper operational match of hardware and software. Confirm hardware integrity of link module.

Connected module is non-link: nn

Meaning:	The link module cannot be identified by the driver software. Link module (**mod_id**) values are listed in Table 9-1.
Action:	Verify proper operational match of hardware and software. Confirm hardware integrity of link module.

Connector out of range

Meaning:	`config` contains an invalid connector number.
Action:	Modify `config`. Refer to the *Configuration Guide.*

DEV CCT: <cct_id> - Ethernet circuit record missing

Meaning:	Circuit **<cct_id>** is undefined.
Action:	Modify `config` to include a circuit record for **<cct_id>.** Refer to the *Configuration Guide.*

DEV CCT: <cct_id> - Ethernet line record missing

Meaning: The driver software cannot associate circuit **<cct_id>** with an existing line record.

Action: Modify `config` to include proper line and circuit records. Refer to the *Configuration Guide.*

DEV CCT: nn - Too many lines assigned to Ethernet connector

Meaning: The same Ethernet line has been assigned to multiple physical connectors.

Action: Modify `config` to ensure that only a single line is assigned to each connector. Refer to the *Configuration Guide.*

DEV CCT: nn - XCVR n out of range in line record

Meaning: `config` contains an invalid transceiver number.

Action: Modify `config`. Refer to the *Configuration Guide.*

DEV CCT: <xx> - Invalid QOS for PPP, QOS = LLC1

Meaning: The Quality of Service parameter for **cct <xx>** was not configured as LLC1 (unreliable datagram). Since LLC1 is the only appropriate Quality of Service for the driver to use for Point to Point protocol, the Quality of Service parameter has been automatically modified to be LLC1. Note that the PPP subsystem, not the driver, provides additional services beyond LLC1.

Action: Modify config so that the Quality of Service parameter is LLC1. Refer to the *Configuration Guide.*

DEV CCT: <xx> - Invalid QOS for SMDS, QOS = LLC1

Meaning: The Quality of Service parameter for **cct <xx>** was not configured as LLC1 (unreliable datagram). Since LLC1 is the only appropriate Quality of Service for the driver to use for SMDS, the Quality of Service parameter has been automatically modified to be LLC1. Note that additional services beyond LLC1 are provided by the SMDS sub-system and not by the synchronous/T1 driver.

Action: Modify config so that the Quality of Service parameter is LLC1. Refer to the *Configuration Guid*e.

driver[s]: Pass-thru protocol enabled on cct n' driver[s]: (LLAN llll RLAN rrrr)

where:

s is the slot number

n is the cct number

l is the 48-bit local LAN address

r is the 48-bit remote LAN address

Meaning: Notifies the user that the Pass-Thru protocol is enabled on **<cct_name>**. Message is associated with the interface for the Pass-Thru protocol.

Ethernet circuit assigned to multiple lines

Meaning: The same Ethernet circuit has been assigned to multiple lines.

Action: Modify `config` to ensure that circuits are assigned to only one line. Refer to the *Configuration Guide.*

Frame Relay enabled on cct <xxx>

Meaning: The synchronous driver has enabled the support for frame relay on **cct <xxx>**. At this point, the management interface is also initialized for **cct <xxx>** if it has been configured.

I/O module PROM Error

Meaning: The module serial number PROM is corrupted.

Action: Verify hardware integrity.

Module DISABLED on slot: nn

Meaning: The link module contained in slot **nn** has failed diagnostics.

Action: Verify hardware integrity of the link module in backplane slot **nn**.

Module not detected on slot: nn

Meaning: A link module has not been installed in slot **nn**.

Action: Install link module in backplane slot.

No configuration summary record

Meaning: `config` is missing a circuit record.

Action: Modify `config` to ensure that all lines and circuits are defined. Refer to the *Configuration Guide.*

No circuits configured on slot: nn

Meaning: No circuits are configured on slot **nn**.

Action: Modify `config` to include required circuit records. Refer to the *Configuration Guide.*

No Ethernet circuits configured for slot

Meaning: No Ethernet circuits are configured for the slot.

Action: Modify `config` to include required circuit records. Refer to the *Configuration Guide.*

No lines configured on slot: nn

Meaning: No lines are configured on slot **nn**.

Action: Modify `config` to include required line records. Refer to the *Configuration Guide.*

Point-to-Point Protocol (PPP) enabled on cct <xx>

Meaning: The synchronous T1 or E1 driver has enabled the support for PPP on cct <xx>. If the LCP Auto-Restart option was configured as **yes** in the **ppp** circuit record for this circuit, the system now initiates the Link Control Protocol (LCP) for **cct <xx>**. Refer to the *Configuration Guide.*

Rx FRMR on circuit <cct_id> Frame: hh hh hh hh hh

Meaning: Circuit **<cct_id>** (running LLC) has received a Frame Reject Frame. **hh hh hh hh hh** are the first five bytes of the frame in hexadecimal format.

SMDS enabled on cct <xx>

Meaning: The synchronous T1 or E1 driver has enabled the support for SMDS on **cct <xx>**.

Too many Ethernet circuits configured for slot

Meaning: `config` contains more Ethernet circuit records than can be accommodated by the board in the slot.

Action: Modify `config` to ensure that no more than two Ethernet circuit records are assigned to a single board. Refer to the *Configuration Guide.*

Unknown dev_type - nn

Meaning: The link module cannot be identified by the driver software. Link module (**mod_id**) values are listed in Table 9-1.

Action: Verify proper operational match of hardware and software. Confirm hardware integrity of link module.

Table 9-1 Module IDs

ID	Opt	HWStat	Name	Enet	Port Configurations			
(hex)	(hex)	(dec)			Sync	Async	Token	Framer
00	01	1	ENET-1	2	0	0	0	0
08	00	8	ENET-2	2	0	0	0	0
10	00	16	SYNC-1	0	4	1	0	0
	01	17	SYNC-1 [1]	0	4	1	0	0
18	00	24	T1-1	0	2	1	0	2
20	00	32	DSE-1	1	2	1	0	0
	01	33	DSE-1 [1]	1	2	1	0	0
28	00	40	DST-4/16	0	2	1	1	0
	01	41	SST-4/16 [1]	0	1	1	1	0
	02	42	DST-4	0	2	1	1	0
	03	43	SST-4 [1]	0	1	1	1	0
	04	44	SST-4/16	0	1	1	1	0
	05	45	STOK-4/16	0	0	0	1	0
	06	46	SST-4	0	1	1	1	0
	07	47	STOK-4 [3]	0	0	0	1	0
38	00	56	T1-2	0	2	1	0	2
	02	58	ST1	0	1	1	0	1
	04	60	T1-56K	0	1	1	0	1
	05	61	E1 [3]	0	2	0	0	2
	06	62	ST1-56K	0	1	1	0	1

Table 9-1 (Continued) Module IDs

ID	Opt	HWStat	Name	Enet	Port Configurations			
(hex)	(hex)	(dec)			Sync	Async	Token	Framer
	07	63	SE1 [3]	0	1	0	0	1
50	00	80	SYNC-2	0	4	1	0	0
	01	81	SYNC-2 [1]	0	4	1	0	0
58	00	88	CMC-FDDI [6]	0	0	0	0	0
	01	89	IPHASE-FDDI [6]	0	0	0	0	0
70	00	112	DSDE-1	2	2	1	0	0
	01	113	DSDE-1 [1]	2	2	1	0	0
	02	114	ENET	2	0	0	0	0
	04	116	DSE-2	1	2	1	0	0
	05	117	DSE-2 [1]	1	2	1	0	0
	06	118	SSE	1	1	1	0	0
	07	119	SSE [1]	1	1	1	0	0
80	04	128	ENET-3 [2]	2	0	0	0	0
98	04	156	DSDE-2 [2]	2	2	1	0	0
A0	00	160	QE/NF	4	0	0	0	0
	04	164	QE/F [4]	4	0	0	0	0
	01	161	DE/NF [5]	2	0	0	0	0
	05	165	DE/F [4, 5]	2	0	0	0	0
B0	00	176	DTOK	0	0	0	2	0
C0	00	192	WF_FDDI	0	0	0	0	0

NOTES:

[1]	COM Port 1 configured for ASYNC; port B on AM8530 is configured for COM1.
[2]	DEFA - Hardware Filtering present; CAMS must be programmed (can contain 2 to 6 CAMS onboard).
[3]	No AM8530; should not be programmed.
[4]	Dual defa hardware; can be depopulated.
[5]	Two ports of hardware depopulated to make dual ethernet module.
[6]	FDDI modules; ID cannot be read from link module I/O space.

9.7 EGP Event Messages

This section contains an alphabetical list of event messages generated by the *egp* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

Already enabled

Meaning: An already enabled EGP has received an NCL `E[nable]` command.

Already disabled

Meaning: An already disabled EGP has received an NCL `DIS[able]` command.

Bad AS, local <ip_address> remote <ip_address>

Meaning: EGP has detected a faulty autonomous system address within `config`.

Action: Verify the integrity of **Local ASN** and **Remote ASN** entries in `config`. A valid entry pair consists of two unique, non-zero addresses. Refer to the *Configuration Guide.*

Bad IP address, <ip_address>

Meaning: EGP has detected a faulty IP address within `config`.

Action: Verify the integrity of **Local Address** and **Remote Address** entries in `config.` Ensure that these parameters do not include broadcast addresses. A valid entry pair consists of two unique, non-zero addresses. Refer to the *Configuration Guide.*

Configuration complete

Meaning: EGP has successfully initialized.

Configuration failed

Meaning: EGP could not initialize because of errors in `config`.

Action: Modify `config`. Refer to the *Configuration Guide.*

Configuration inconsistency repaired

Meaning: EGP has noted a minor discrepancy in `config`. It has initialized using default values.

Configuration record not found

Meaning: `config` does not include an EGP record.

Action: Modify `config`. Refer to the *Configuration Guide.*

Enable failed

Meaning: EGP failed to enable in response to an NCL **`E[nable]`** command.

Entity disabled

Meaning: EGP has been disabled in response to an NCL **`DIS[able]`** command.

Entity disabled, shutdown failed

Meaning: EGP has been disabled in response to an NCL **`DIS[able]`** command. Currently existing neighbor connections were terminated abruptly, and not in accordance with EGP protocol.

Entity enabled

Meaning: EGP has been enabled in response to an NCL **`E[nable]`** command.

Entity enabled, startup failed

Meaning: EGP has been enabled in response to an NCL **`E[nable]`** command. EGP, however, has failed to establish connections with neighboring routers.

Entity is still closing connections

Meaning: A busy EGP has received an NCL **`D[isable]`** command.

Action: Wait, then re-issue NCL command.

Entity not initialized

Meaning: EGP has received an NCL command before it has completed initialization.

Error nnnn attaching to <ip_address>

Meaning: EGP has encountered a specific error attempting to attach to the neighbor designated by **<ip_address>**. **nnnn** designates the error condition as listed in Table 9-2.

Error nnnn detaching from <ip_address>

Meaning: EGP has encountered a specific error attempting to detach from to the neighbor designated by **<ip_address>**. **nnnn** designates the error condition as listed in Table 9-2.

Table 9-2 Error Codes

Code	Explanation
320	Interface not found
321	Requested resource unavailable
322	Node out of memory
32	Node out of buffers
324	Necessary parameter unspecified
325	Option or command not supported
326	Invalid output parameter
327	Connection not established
328	Connection already exists
329	Connection closing
32a	Invalid operation for state
32b	Connection timed out
32c	Unknown network requested
32d	Connection refused
32e	Connection reset locally
32f	Connection reset by peer

Table 9-2 (Continued)Error Codes

Code	Explanation
320	Interface not found
330	Bad packet checksum
331	Packet too big
332	Unsupported options encountered
333	Unsupported flag encountered
334	Received ill-formed reset
335	Received ill-formed segment
336	Received ill-formed acknowledgment
337	Received duplicate acknowledgment
338	Received duplicate segment
339	Received segment out of sequence
33a	Send window closed
33b	Send (retransmit) ring buffer overflow
33c	Receive window closed
33d	Receive (resequencing) queue overflow
33e	Unknown (network) message code
33f	Unknown (network) message type
340	Internal (fatal) error

Insufficient buffers for operation

Meaning:	The system cannot provide sufficient buffers for EGP operations.
Action:	Consider reducing the number of neighbors to reduce buffer requirements. A memory upgrade may be required.

Insufficient memory for operation

Meaning: The system cannot provide sufficient memory for EGP operations.

Action: Consider reducing the number of neighbors to save memory space. A memory upgrade may be required.

Internal error

Meaning: EGP has encountered an unspecified error attaching to or detaching from a neighbor.

Invalid action

Meaning: EGP received a command, that while otherwise valid, was inappropriate to its current state.

Invalid number of neighbors

Meaning: While checking its neighbor table, EGP found too few or too many entries.

Action: Modify `config` to ensure that the number of neighbors is greater than 1, but less than 20. Refer to the *Configuration Guide*.

IP entity not available

Meaning: The IP entity is not available. EGP cannot function in the absence of IP.

Action: Check if IP has been disabled. Check `config` to verify proper IP Router configuration parameters. Refer to the *Configuration Guide*.

IP entity not ready

Meaning: The IP entity is not currently available.

Action: Wait for IP to initialize.

Neighbor <ip_address> acquired

Meaning: EGP has acquired a new neighbor (identified by **<ip_address>**).

Neighbor <ip_address> down

Meaning: The EGP neighbor-reachability algorithm has declared **<ip_address>** in the **DOWN** state. In the **DOWN** state, EGP allocates resources to the neighbor, and responds to `Request, Cease,` and `Hello` commands.

Neighbor <ip_address> unacquired

Meaning: The EGP neighbor reachability algorithm has declared **<ip_address>** in the **IDLE** state. In the **IDLE** state, EGP allocates no resources to the neighbor, and responds only to a Request Command or a system or operator-generated start event.

Neighbor <ip_address> up, polling disabled

Meaning: The EGP neighbor reachability algorithm has declared **<ip_address>** in the **UP** state. In the **UP** state, EGP allocates resources to the neighbor, and responds to all commands and request. EGP is in the passive state; it does not issue poll commands.

Neighbor <ip_address> up, polling enabled

Meaning: The EGP neighbor reachability algorithm has declared **<ip_address>** in the **UP** state. In the **UP** state, EGP allocates resources to the neighbor, and responds to all commands and request. EGP is in the active state; it issues poll commands.

Source address equal to destination

Meaning: *`config`* contains identical values for the **Local Address** and **Remote Address** parameters.

Action: Modify *`config`* to ensure the accuracy of local and remote addresses. Refer to the *Configuration Guide.*

10 Event Messages, I to Z

This chapter describes the event messages generated by the system and stored by the system in the event log. This chapter contains the messages, arranged alphabetically by managed object, beginning with the letters **i** to **z**.The messages are arranged alphabetically within each section.

10.1 IP Event Messages

This section contains an alphabetical list of event messages generated by the *ip* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

bad cg <cg_id> on <ip_address>

Meaning: IP has detected a discrepancy in the circuit group record **<cg_id>**.

Action: Modify `config`. Refer to the *Configuration Guide.*

bad ip address <ip_address>

Meaning: The IP address **<ip_address>** (probably a broadcast address) is invalid.

Action: Modify `config` to repair **<ip_address>**. Refer to the *Configuration Guide.*

bad mask <mask>/<ip_address>

Meaning: IP has detected a discrepancy between an IP address, **<ip_address>**, and its associated mask, **<mask>**.

Action: Modify `config` to repair **<ip_address>** and/or **<mask>**. Refer to the *Configuration Guide*.

bad mtu (nn) on circuit <cct_id>

Meaning: Circuit **<cct_id>** will not support the minimum IP Maximum Transmission Unit (mtu).

bad rx bcast <bcast> on <ip_address>

Meaning: *`config`* contains an invalid receive broadcast address on the interface designated by **<ip_address>**.

Action: No action is required as IP will use a default broadcast address. Note that the Configuration Editor guards against this error. This message should only be seen if a user has attempted to modify *`config`* without using the Configuration Editor.

bad tx bcast <bcast> on <ip_address>

Meaning: *`config`* contains an invalid transmit broadcast address on the interface designated by **<ip_address>**.

Action: No action is required as IP will use a default broadcast address. Note that the Configuration Editor guards against this error. This message should only be seen if a user has attempted to modify *`config`* without using the Configuration Editor.

duplicate ip address

Meaning: Multiple network interfaces have been configured with the same IP address.

Action: Modify *`config`* to ensure the uniqueness of interface addresses. Refer to the *Configuration Guide.*

entity already disabled

Meaning: An already disabled IP has received an NCL **`DIS[able]`** command.

entity already enabled

Meaning: An already enabled IP has received an NCL **`E[nable]`** command.

entity disabled

Meaning: IP has been disabled in response to an NCL **`DIS[able]`** command.

entity enabled

Meaning: IP has been enabled in response to an NCL `E[nable]` command.

entity reset

Meaning: IP has reinitialized.

filters configured

Meaning: IP has configured source-address, destination- address, and/or TCP/UDP port filters as specified in *`config`*.

global broadcasts will not be received

Meaning: The **Global Broadcast** parameter has been set to **No**.

Action: If you want to receive global broadcasts, modify *`config`* to set the value of **Global Broadcast** to **Yes**. Refer to the *Configuration Guide.* If you do not want to receive global broadcasts, no action is required.

icmp: <ip_address> unreachable (host)

Meaning: IP has received an Internet Control Message Protocol (ICMP) **Destination Unreachable** message from **<ip_address>**. The message contains a Code Field value of **1**, indicating that the IP host designated by **<ip_address>** is unreachable.

icmp: <ip_address> unreachable (net)

Meaning: IP has received an Internet Control Message Protocol (ICMP) **Destination Unreachable** message from **<ip_address>**. The message contains a Code Field value of 0, indicating that the network designated by **<ip_address>** is unreachable.

icmp: quench from <ip_address>

Meaning: IP has received an Internet Control Message Protocol (ICMP) **Source Quench** message from **<ip_address>**.

icmp: redirect from <ip_address>

Meaning: IP has received an Internet Control Message Protocol (ICMP) **Redirect** message from **<ip_address>**.

insufficient memory

Meaning: The system cannot provide sufficient memory for IP operations.

Action: A memory upgrade may be required.

invalid operation for state

Meaning: IP received a command, that while otherwise valid, was inappropriate to its current state.

ip: Source Routing not enabled (Token Ring only)

Meaning: The **Source Route** parameter for an IP network interface record is set to **Yes**, but the interface does not use token ring media. IP does not enable source routing for the interface since end system source routing applies to token ring media only.

Action: Modify `config`. Change **Source Route** to **No.**

network disabled on <ip_address>

Meaning: IP has disabled the network interface **<ip_address>**.

network enabled on <ip_address>

Meaning: IP has initialized the network interface **<ip_address>**.

no network interfaces configured

Meaning: `config` contains no network interface records.

Action: Modify `config` to include network definitions for all network interfaces. Refer to the *Configuration Guide.*

resolved: <ip_address1>/<ip_address2>

Meaning: IP has added a new entry, learned through the Address Resolution Protocol, to its Address Translation Table. **<ip_address1>** is the host address; **<ip_address2>** is the network interface address.

SR max RDs exceeded in explorer packet

Meaning: The maximum number of Route Descriptors (RDs) was exceeded in an All Routes Explorer packet or a Spanning Tree Explorer (STE) packet.

The maximum number of Route Descriptors for a source routed packet is 8. The packet cannot be accepted and is dropped.

SR max RDs exceeded in SRF packet

Meaning: The maximum number of Route Descriptors (RDs) was exceeded in a Specifically Routed Frame (SRF). The maximum number of Route Descriptors for a source routed packet is 8. The packet cannot be accepted and is dropped.

SR Rif_table out of space

Meaning: The Routing Information Field (RIF) table is out of space. The **Rif_table** contains routing information fields that are used to route source routed packets between the Wellfleet bridge/router and remote hosts over token ring networks.

Action: A memory upgrade may be required. For a description of the size of the **Rif_table**, refer to Table 8-1.

SR sr_es_find: Madr_table out of space

Meaning: The MAC Address Table (**Madr_table**) is out of space. The MAC Address Table contains MAC Addresses that are used for both End Station (ES) Source Routing over IP as well as Intermediate Station (IS) Source Routing over the bridge.

Action: A memory upgrade may be required. For a description of the size of the **Madr_table**, refer to Table 8-1.

SR Sr_es_table out of space

Meaning: The source routing End Station table is out of space. The **Sr_es_table** is a table that contains destination MAC addresses. It contains pointers to the RIF table.

Action: A memory upgrade may be required. For a description of the size of the **Sr_es_table**, refer to Table 8-1.

too many networks configured for this circuit group

Meaning: A single network interface has been configured with more than 16 networks.

Action: Modify *config*. Do not put more than 16 networks on a single circuit group.

10.2 Line Event Messages

This section contains alphabetical listings of event messages generated by the E1, synchronous, and T1 instances of the *line* managed object.

Each message is followed by an explanation of the message contents and a recommended action (if any is required).

10.2.1 E1 Line Event Messages

This section contains an alphabetical listing of the event messages generated by the E1 instance of the *line* managed object.

Bad connector (DS1 number) in line record

Meaning: `config` contains a faulty response in the **Connector** field of the E1 line record.

Action: Modify `config` to correct **Connector**. Refer to the *Configuration Guide.*

Circuit specified on line is not on slot

Meaning: The software cannot associate a line and circuit record.

Action: Examine `config` to ensure that you have properly configured this line and its circuits. Refer to the *Configuration Guide.* If problems persist, contact Customer Support.

Clocking mismatch, defaulting to manual

Meaning: A configuration mismatch has occurred in clocking modes between circuits.

Action: Reconfigure clocking on the DS-1 and DS1-2 circuits.

E1 line record missing

Meaning: `config` does not contain a record for this E1 line.

Action: Modify `config` to include a E1 line configuration record. Refer to the *Configuration Guide.*

Module slot inconsistent with line records

Meaning: There is a mismatch between the slot number that houses the E1 board and the **Slot Number** parameter in the E1 line record.

Action: Modify `config` to reflect the slot number which houses the E1 board. Refer to the *Configuration Guide.*

Sync lost

Meaning: **slot#_E1_n** has lost synchronization.

Sync recovered

Meaning: **slot#_E1_n** has recovered synchronization.

Too many lines for dual E1 module

Meaning: `config` contains an excessive number (greater than two) of E1 line records for this slot.

Action: Modify `config` to ensure that no more than two E1 line records are established for any slot.

10.2.2 Synchronous Line Event Messages

This section contains an alphabetical listing of the event messages generated by the synchronous instance of the *line* managed object.

Connector nn not on this module

Meaning: The configuration record reflects a nonexistent physical connector.

Action: Modify **Connector** in the line record in `config`. Refer to the *Configuration Guide.*

No buffers available for deadlock processing

Meaning: Indicates a degenerative line condition resulting in the lack of receive buffers at both the line source and termination.

No V35 circuits configured for slot

Meaning: A circuit(s) designated by the **Circuit Name** parameter in the line record has not been configured.

Action: Modify `config` to ensure that it includes circuit records for all circuits. Refer to the *Configuration Guide.*

Sync circuit assigned to multiple lines

Meaning: One or more line records contains references to the same point-to-point circuit.

Action: Modify `config` to ensure that all line and circuit records are consistent. Refer to the *Configuration Guide.*

T1 circuit record missing

Meaning: `config` does not contain a record for a T1.

Action: Modify `config` to ensure that a circuit record exists for each T1 circuit named by the **Circuit1 Name** and **Circuit2 Name** T1 line parameters. Refer to the *Configuration Guide.*

Too many lines assigned to V.35 connector

Meaning: `config` contains an excessive number (greater than two) of line records for a single connector.

Action: Modify `config` to ensure that no more than one line record references a specific physical connector. Refer to the *Configuration Guide.*

V35 line record missing

Meaning: A line record has not been associated with the circuit identified by **DEV CCT: <cct_name>**.

Action: Modify `config` to ensure that it includes properly associates line and circuit records. Refer to the *Configuration Guide.*

10.2.3 T1 Line Event Messages

This section contains an alphabetical listing of the event messages generated by the T1 instance of the *line* managed object.

Bad connector (DS1 number) in line record

Meaning: `config` contains a faulty response in the **Connector** field of the T1 line record.

Action: Modify `config` to correct **Connector**. Refer to the *Configuration Guide.*

Carrier lost

Meaning: **slot#_DS1_n** has lost the carrier signal.

Carrier recovered

Meaning: **slot#_DS1_n** has recovered the carrier signal.

Circuit specified on line is not on slot

Meaning: The software cannot associate a line and circuit record.

Action: Examine `config` to ensure that you have properly configured this line and its circuits. Refer to the *Configuration Guide.* If problems persist, contact Customer Support.

LBO out of range, setting to 655

Meaning: The **Line Buildout** field in `config` contains a faulty value (greater than 655).

Action: Modify `config` to reflect a **Line Buildout** value of less than 655. Refer to the *Configuration Guide.*

Module slot inconsistent with line records

Meaning: There is a mismatch between the slot number which houses the T1 board and the **Slot Number** parameter in the T1 line record.

Action: Modify `config` to reflect the slot number which houses the T1 board. Refer to the *Configuration Guide.*

Red alarm cleared

Meaning: **slot#_DS1_n** has cleared a Red alarm.

Red alarm received

Meaning: A Red alarm has been received on **slot#_DS1_n**.

T1 line record missing

Meaning: `config` does not contain a record for this T1 line.

Action: Modify `config` to include a T1 line configuration record. Refer to the *Configuration Guide.*

Too many lines for dual T1 module

Meaning: `config` contains an excessive number (greater than two) of T1 line records for this slot.

Action: Modify `config` to ensure that no more than two T1 line records are established for any slot.

Yellow alarm cleared

Meaning: **slot#_DS1_n** has cleared a Yellow alarm.

Yellow alarm received

Meaning: A Yellow alarm has been received on **slot#_DS1_n**.

10.3 Manager Event Messages

This section contains an alphabetical list of event messages generated by the *mgr* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

auto enabling <entity>

Meaning: The manager is auto-enabling the specified device or service. **<entity>** can be any of the following: a circuit of any type, an X.25 virtual circuit, the DECnet Router, the IP Router, the Bridge, the XNS Router, the IPX Router, EGP, SNMP, TCP, or telnet.

enable <entity> failed

Meaning: The manager could not enable **<entity>**.

Action: Contact Customer Support.

10.4 OSPF Event Messages

This section contains an alphabetical list of event messages generated by the *ospf* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

DD: Extern option mismatch

Meaning: The Hello external/stub option specified doesn't match the configured option.

DD: Nbr's rtr = my rtrid

Meaning: The OSPF entity has detected another OSPF router with the same router identification in a Database Description packet.

DD: Nbr state low

Meaning: OSPF has received a Database Description packet from a neighbor whose states is too low to honor, meaning the router will drop the **Ls Req** and **Ls Update** packets from the neighbor whose state is below Exchange in the following list:

Full
Loading
Exchange
Exch Start
2 Way
Init
Attempt
Down.

DD: Unknown nbr

Meaning: A Database Description packet has been received from an unknown neighbor.

Hello: Extern option mismatch

Meaning: The Hello external/stub option specified doesn't match the configured option.

Hello: IF dead timer mismatch

Meaning: The dead timer value specified in an incoming Hello packet does not match the configured value.

Hello: IF hello timer mismatch

Meaning: The hello timer value specified in an incoming Hello packet does not match the configured value.

Hello: IF mask mismatch

Meaning: The mask value specified in an incoming Hello packet does not match the configured value.

Hello: Unknown Virt nbr

Meaning: An unknown virtual link neighbor has tried to establish an adjacency with the resident OSPF entity.

IP: Bad IP Dest

Meaning: The OSPF entity received an IP packet that was not destined for it.

IP: Bad IP proto id

Meaning: The protocol ID in this IP packet was incorrect or unknown.

IP: Bad OSPF pkt type

Meaning: An incoming OSPF packet contains an unknown or incorrect OSPF packet type.

IP: Pkt src = my IP addr

Meaning: The source address in the IP packet was found to be the same as the address configured for the OSPF entity.

LS Req: Bad pkt

Meaning: OSPF has a received a Bad Link State Request.

Ls Req: Empty request

Meaning: OSPF has received an empty Link State Update Request.

Ls Req: Nbr state low

Meaning: The state of a neighbor sending a Link State Request is too low to honor, meaning the router will drop the **Ls Req** and **Ls Update** packets from the neighbor whose state is below Exchange in the following list:

Full
Loading
Exchange

Exch Start
2 Way
Init
Attempt
Down.

Ls Req: Unknown nbr

Meaning: An Unknown neighbor has sent a Link State Request to the OSPF entity.

Ls Update: Bad LS chksum

Meaning: The checksum calculated over the contents of this incoming Link State Update packet does not agree with the specified value.

Ls Update: less recent rx

Meaning: The OSPF entity has received a Link State Advertisement that is less recent than the current internal copy.

LS Update: Nbr state low

Meaning: The OSPF entity has received a Link State Update from a neighbor in a state too low to be processed, meaning the router will drop the **Ls Req** and **Ls Update** packets from the neighbor whose state is below Exchange in the following list:

Full
Loading
Exchange
Exch Start
2 Way
Init
Attempt
Down.

Ls Update: Newer self-gen LSA

Meaning: The OSPF entity has received a self-generated Link State Advertisement that appears to be newer than the internal copy.

Ls Update: Unknown nbr

Meaning: A Link State Update has been received from an unknown neighbor.

Ls Update: Unknown type

Meaning: The OSPF entity received an unknown Link State Update type.

OSPF: Area mismatch

Meaning: The OSPF entity discerns that one of its interfaces has been mismatched with a configured area.

OSPF: Auth key != area key

Meaning: There is a mismatch between the authentication key specified in the packet and the configured value.

OSPF: Bad intf area id

Meaning: The interface area identification specified in this packet does not match the one configured for this OSPF interface. Or, the packet was received on an interface belonging to another area.

OSPF: Bad OSPF checksum

Meaning: The checksum computed on this packet does not agree with the value specified in the packet.

OSPF: Bad OSPF version

Meaning: The version of OSPF as specified in this packet is incompatible with the version supported by the OSPF entity. Versions 1 and 2 are compatible; Wellfleet complies with Version 2.

OSPF: Bad virt link info

Meaning: The OSPF entity has interpreted information from a non-backbone source as incorrectly appearing on the backbone.

OSPF: Choosing DR INTF X.X.X.X
OSPF: DR = X.X.X.X
OSPF: BDR = X.X.X.X

Meaning: OSPF is performing the Designated Router algorithm with respect to interface **X.X.X.X.** Designated Router **(DR)** and Backup Designated Router **(BDR)** indicate the results of the algorithm.

OSPF: Entity enabled

Meaning: The OSPF entity has initialized correctly and has reached the enabled state.

OSPF: Packet is too small

Meaning: OSPF has received a packet that is too small.

OSPF: Packet size > IP length

Meaning: OSPF has received a packet exceeding the allowable IP datagram length.

OSPF: Received on down IF

Meaning: OSPF received a packet on an interface that was considered to be down.

OSPF: TQ_IFCHECK: Interface if_name(X.X.X.X) is down

Meaning: When OSPF entity was enabling, it found a specified interface in the down state. OSPF periodically checks the status of its interfaces and therefore is capable of recognizing state changes.

OSPF: Transmit bad

Meaning: OSPF was unable to transmit a packet.

OSPF: TRANS [IF/NBR] ID = X.X.X.X Event: X States: Y -> Z

Meaning: The transit interface or neighbor (**X.X.X.X**) has received an event (**X**) that caused it to pass through a state change from state **Y** to state **Z**. The following events (**X**) can cause state machine changes for interfaces or neighbors:

Events Received by Neighbors		Events Received by Interfaces
Hello Received	Reset Adjacency	Interface UP
Start	Kill Neighbor	Wait Timer
Two Way Received	Inactivity Timer	Backup Seen
Adjacency OK	One way	Neighbor Change
Negotiation Done	Lower Level Down	Interface Down
Bad LS Request	Seq # Mismatch	Loop Indication
Exchange Done	Loading Done	Unloop Indication

The associated states (**Y** to **Z**) that affected neighbors or interfaces can pass through are:

2 States Associated w/Neighbors		States Associated w/Interfaces
Down	Full	Down
Attempt	SCVirtual	Loopback
Init		Waiting
2 Way		P to P
Exch Start		DR
Exchange		BackupDR
Loading		DR Other

10.5 SMDS Event Messages

This section contains an alphabetical list of event messages generated by the *smds* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

bad configuration file

Meaning: SMDS detected an inconsistency in the configuration.

Action: Modify `config`. Refer to the *Configuration Guide*.

<cct_name>: illegal packet received

Meaning: A packet was received which did not conform to standard SMDS format. The packet was dropped. No further action is necessary. **<cct_name>** identifies the SMDS circuit.

invalid SMDS ARP group address for cct <xx>

Meaning: The SMDS Address Resolution Protocol (ARP) group address in the SMDS circuit record for **cct <xx>** is invalid. An SMDS address is 10 digits in length, and each digit must be in the range 0 to 9.

Action: Modify `config`. Refer to the *Configuration Guide.*

invalid SMDS group address for cct <xx>

Meaning: A group address in the SMDS circuit record for **cct<xx>** is invalid. An SMDS address is 10 digits in length, and each digit must be in the range 0 to 9.

Action: Modify `config`. Refer to the *Configuration Guide.*

invalid SMDS individual address for cct <xx>

Meaning: The individual address in the SMDS circuit record for **cct <xx>** is invalid. An SMDS address is 10 digits in length, and each digit must be in the range 0 to 9.

Action: Modify `config`. Refer to the *Configuration Guide.*

Madr_table is full

Meaning: The MAC Address Table (Madr_table) is out of space. The MAC Address Table is a hash table which contains MAC Addresses used by the Wellfleet Bridge/Router.

Action: A memory upgrade may be required.

no SMDS address for DRS area <xx>, node <yy>

Meaning: An attempt was made to transmit a DECnet Routing Service (DRS) packet to DECnet area **<xx>**, system **<yy>**, but no DECnet Address Map record exists which specifies which remote SMDS address should be used for the packet.

Action: Modify `config` to add the appropriate DECnet Address Map entry. Refer to the *Configuration Guide.*

Smds_bridge_table is full

Meaning: The SMDS Bridge Table (Smds_bridge_table) is out of space. The SMDS Bridge Table contains the mapping between remote MAC addresses and remote SMDS addresses.

10.6 Telnet Event Messages

This section contains an alphabetical list of event messages generated by the *telnet* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

---

port 23 connected to <ip_address>

Meaning: A telnet virtual terminal connection between the system and **<ip_address>** has been established through the *well known* telnet port.

---

port 23 disconnected from <ip_address>

Meaning: A telnet virtual terminal connection between the system and **<ip_address>** has been disconnected.

10.7 TFTP Event Messages

This section contains an alphabetical list of event messages generated by the *tftp* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

TFTP RRQ from <ip_addr>, file <file_name>

Meaning: A request was received from the IP address to read the specified file.

Cannot open file: file open limit exceeded

Meaning: The TFTP session limit has been exceeded so the specified file cannot be opened.

---

transfer <file_name> aborted after <#> retransmissions

Meaning: TFTP could not transfer the file and aborted the attempt.

Action: Reset the **max_retransmissions** parameter to a higher value. Refer to the *Configuration Guide*.

10.8 Transmission Control Protocol Event Messages

This section contains an alphabetical list of event messages generated by the *tcp* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

bad configuration, using defaults

Meaning:	TCP has rejected user-supplied protocol parameters; TCP will initialize using default parameters.
Action:	Modify `config` to accept default parameters. Refer to the *Configuration Guide.*

configuration complete

Meaning:	TCP has completed configuration using valid user-supplied parameters.

no configuration, using defaults

Meaning:	TCP has completed configuration using default parameters in the absence of user-supplied parameters.

10.9 XNS Routing Event Messages

The XNS routing modules generate event messages as it operates, and it stores these messages in the event log. This section provides an alphabetical listing of these messages; each message is accompanied by an explanation of the message contents and explains the recommended action (if any is required).

xrx: <cg_name> : New Rt to <xrx_net> via <router_addr>

Meaning:	The Xerox routing module generates a new event message whenever it learns a new route or update an existing route. The new route is specified by **xrx_net**, which is the destination network the new or updated route refers to, and by **router_addr,** which is the full Xerox address of the next hop router. The **router_addr** is represented in hexadecimal notation: the first 8 digits are the network address, the last 12 digits are the host address.

NOTE

If **<xrx_net>** is zero and the second part of the **<router_addr>** is the MAC address of the box, then the first part of the **<router_addr>** is a directly connected network on a slot other than slot 2.

xrx <cg_name>: Rcvd Err Pkt - Num = <err_num>, Param = <err_param>

Meaning: Whenever an error packet is received by the Xerox Router, this message is generated. The circuit group the error packet was received on is **<cg_name>**, the error number of the packet (in decimal) is **<err_num>**, and the error parameter of the packet (**516** in decimal) is **<err_param>**.

Valid error numbers are:

0	Unspecified error at destination
1	Bad checksum or other packet inconsistency at destination
2	Unknown socket at destination
3	Destination resource limitations
512	Unspecified error before reaching destination
513	Bad checksum or inconsistency before reaching destination
514	Destination host cannot be reached
515	Packet exceeded hop count of 15
516	Packet too large to forward; the **<err_param>** is the maximum acceptable length.

<xrx>:CG <cg_name>:Del Rtto<dest_net> via <next_hop_net> : <next_hop_gw>

where:

cg_name	is the name of the circuit group the route was learned on.
dest_net	is the destination network the route referred to.
next_hop_net	is the directly connected network that was to be used to get to the next hop router.
next_hop_gw	is the XNS or IPX address of the router that was the next hop for traffic destined for **<dest_net>**.

10.10 IPX Routing Event Messages

The IPX routing module generates event messages as it operates, and it stores these messages in the event log. This section provides an alphabetical listing of these messages; each message is accompanied by an explanation of the message contents and explains the recommended action (if any is required).

<ipx>:CG <cg_name>:Del Rtto <dest_net> via <next_hop_net> : <next_hop_gw>

where:

cg_name	is the name of the circuit group the route was learned on.
dest_net	is the destination network the route referred to.
next_hop_net	is the directly connected network that was to be used to get to the next hop router.
next_hop_gw	is the XNS or IPX address of the router that was the next hop for traffic destined for **<dest_net>**.

ipx: <cg_name> : New Rt to <ipx_net> via <router_addr>

Meaning: The IPX routing module generates a new event message whenever it learns a new route or updates an existing route. The new route is specified by **ipx_net**, which is the destination network the new or updated route refers to, and by **router_addr,** which is the full IPX address of the next hop router. The **router_addr** is represented in hexadecimal notation: the first 8 digits are the network address, the last 12 digits are the host address.

NOTE

If **<ipx_net>** is zero and the second part of the **<router_addr>** is the MAC address of the box, then the first part of the **<router_addr>** is a directly connected network on a slot other than slot 2.

ipx: cg_name : New Srv <srvr_name> at <ipx_addr)

Meaning: This event message will be generated whenever the Service Advertising Protocol learns a new system, **srvr_name**, which is a character string that is truncated after 11 digits, and by the full ipx address, **ipx_addr**, which is shown in hexadecimal notation (the first 8 digits are the network address; the last 12 digits are the host address).

ipx: cg_name: Del Srv <srvr_addr> at <ipx_addr>

Meaning: This event message will be generated whenever the Service Advertising Protocol learns that a system has been deleted. The system is specified by **srvr_name**, which is a character string that truncated after 11 digits, and by the full ipx address, **ipx_addr**, which is shown in hexadecimal notation (the first 8 digits are the network address, the last 12 digits are the host address).

CG <cg_name>: NetBIOS Bcast Rt ee ignored - bad net

Meaning: If a NetBIOS Broadcast Static Route configured on an IPX interface has a Destination Network of 00000000 or FFFFFFFF, the route will be ignored and this message will be generated. The circuit group associated with the IPX network interface is **<cg_name>,** and **ee** is the number of the invalid NetBIOS Static Route entry, as shown on the configuration screen.

CG <cg_name>: NetBIOS Bcast Rt ee ignored - bad name

Meaning: If a NetBIOS Broadcast Static Route configured on an IPX interface has a **NetBIOS Resource Name** that is invalid (it may be too long, it may contain non-hexadecimal characters following a \, etc.) the route will be ignored and this message will be generated. The circuit group associated with the IPX network interface is **<cg_name>,** and **ee** is the number of the invalid NetBIOS Static Route entry, as shown on the configuration screen.

CG <cg_name>:SAP Net Fltr ee ignored - bad net num

Meaning: If a SAP Network Level Filter configured on an IPX interface has a "Network Number" of 00000000, the filter will be ignored and this message will be generated. The circuit group associated with the IPX network interface is **<cg_name>,** and **ee** is the number of the invalid SAP Network Level Filter entry, as shown on the configuration screen.

CG <cg_name>:SAP Srv Fltr ee ignored - bad srv type

Meaning: If a SAP Server Level Filter configured on an IPX interface has a Server Type of FFFF, the filter will be ignored and this message will be generated. The circuit group associated with the IPX network interface is **<cg_name>,** and **ee** is the number of the invalid SAP Server Level Filter entry.

10.11 X.25 Event Messages

This section contains an alphabetical list of event messages generated by the *X.25* managed object. Each message is followed by an explanation of the message contents and a recommended action (if any is required).

bad configuration

Meaning: X.25 had detected an inconsistency in the configuration.

Action: Modify *config*. Refer to the *Configuration Guide.*

call accepted from DTE <dte_addr>

Meaning: An incoming call was accepted from remote DTE address **<dte_addr>**.

call attempt: CCT.[a.b.c.d.]

Meaning: A call has been made to the destination with IP Address **a.b.c.d** on the circuit named **CCT.**

call attempt on CCT

Meaning: A call has been made on the virtual circuit named .

call: <cct_name>.ip_addr.#

Meaning: A DDN or PDN call has been established with the remote host or gateway identified by **ip_addr**. **<cct_name>** identifies the X.25 DDN or PDN circuit and **#** identifies the logical connection number.

call cleared on <svc_name> (C=nn) (D=nn)

Meaning: A call has been properly cleared on point-to-point dedicated virtual circuit **<svc_name>**. **(C=nn)** contains the decimal contents of the *Cause* field (octet 4) of the supervisory header of the packet that CLEARed the call. **(D=nn)** contains the decimal contents of the *Diagnostic Code* field (octet 5) of the same packet. Table 10-1 to Table 10-5 provide a summary listing of these decimal values. These tables are found at the end of this chapter.

call established on <svc_name>

Meaning: A call has been properly established on virtual circuit **<vc_name>**. A call is established by a **CALL REQUEST, INCOMING CALL, CALL ACCEPTED, CALL CONNECTED** packet sequence.

call fail on CCT (C=CAUSE, D=DIAGNOSTIC)

Meaning: The call made on virtual circuit **CCT** has failed. The Cause and Diagnostic codes from the packet that cleared the call are given by **CAUSE** and **DIAGNOSTIC** in decimal values.

clr call from DTE <dte_addr> (address not found)

Meaning: The incoming call from DTE address **<dte_addr>** has been cleared because the remote DTE address was not found in the X.25 Address Map.

Action: Modify `config` by adding an entry for the remote station in the X.25 Address Map for the appropriate X.25 circuit. Refer to the *Configuration Guide.*

clr call from DTE <dte_addr> (max calls active)

Meaning: The incoming call from DTE address **<dte_addr>** has been cleared because the maximum number of calls is already active between the X.25 circuit and the remote station.

Action: Modify `config` by increasing the **Max Conns** parameter in the X.25 Address MAP for the appropriate X.25 circuit. Refer to the *Configuration Guide.*

clr call from DTE <dte_addr> (no host found)

Meaning: The incoming call from DTE address **<dte_addr>** has been cleared because no host circuit was found for the call. For example, if the call was received on a line connected to a DDN network and the line hadn't been configured as a DDN network, then the call cannot be accepted.

Action: Modify `config` by configuring the circuit with the appropriate network type. Refer to the *Configuration Guide.*

clr call from DTE <dte_addr> (no idle circuits)

Meaning: The incoming call from DTE address **<dte_addr>** has been cleared because an idle circuit is not available to receive the incoming call.

clr: <cct_name>.ip_addr.# (C=nn) (D=nn)

Meaning: An established DDN or PDN call to the remote host or gateway identified by **ip_addr** has been cleared. **<cct_name>** identifies the X.25 DDN or PDN circuit and **#** identifies the logical connection number. **(C=nn)** contains the decimal contents of the **Cause** field (octet 4) of the supervisory header of the packet that **CLEAR**ed the call. **(D=nn)** contains the decimal contents of the **Diagnostic Code** field (octet 5) of the same packet. Table 10-1 to Table 10-5 at the end of this chapter provide a summary listing of these decimal values.

disable ignored for <vc_name>, still processing a previous request

Meaning: A user has issued a `DIS[able]` command for the busy virtual circuit, **<vc_name>**.

disable in progress for <vc_name>

Meaning: Virtual circuit **<vc_name>** is being disabled.

enable ignored for <vc_name>, still processing a previous request

Meaning: A user has issued an `E[nable]` command for an already enabled (and busy) virtual circuit.

enable in progress for <vc_name>

Meaning: Virtual circuit **<vc_name>** is being enabled.

fail: CCT.[a.b.c.d.] (C=CAUSE, D=DIAGNOSTIC)

Meaning: A call to the destination with IP address **a.b.c.d.** on the circuit **CCT** has failed. The Cause and Diagnostic codes from the packet that cleared the call are given by **CAUSE** and **DIAGNOSTIC** in decimal values.

High LCN (<xxxx>) > 4095: now High = <yyyy> & Low = <zzzz>

Meaning: The **High LCN** value **xxxx** entered by the user exceeded the maximum value allowed of 4095. The **High LCN** 8value has been adjusted downwards to **yyyy**, and the **Low LCN** value has been adjusted downwards to **zzzz** to bring them within legal range. For example, if the **Low LCN** value entered by the user was 5000, and the **High LCN** value entered by the user was 5010, then the **Low LCN** and **High LCN** values are adjusted downwards to 4085 and 4095, respectively. Thus, the actual number of LCNs desired by the user (in this case, 5010 - 5000 = 10) remains unchanged (4095 - 4085 = 10).

Action: Modify `config`; refer to the *Configuration Guide.*

---

high lcn (xx) < low lcn (yy); using (yy) for both

Meaning: The **high lcn** (logical channel number) configured for an X.25 circuit is lower than the **low lcn** configured for that circuit. The value given by the **low lcn** will be used for both the **low lcn** and **high lcn.**

Action: Modify `config` to configure the high and low lcn values appropriately. Refer to the *Configuration Guide.*

---

high lcn - low lcn > max lcns (xx); using (yy) for high lcn

Meaning: The difference between the **high lcn** value and **low lcn** value is greater than the number of maximum number of **lcn**s allowed for the circuit. The **high lcn** will be adjusted downward to allow the maximum number of **lcns** to be configured.

Action: Modify `config` to configure the **high** and **low lcn** values appropriately. Refer to the *Configuration Guide.*

High LCN (<xxxx>) >4095: now High = <yyyy> & Low = <zzzz>

Meaning: The **High LCN** value **xxxx** that you entered exceeded the maximum value allowed of 4095. The **High LCN** has been adjusted downward to **yyyy** and the and the **Low LCN** value has been adjusted downward to **zzzz** to bring them within the legal range. For example, if you entered a **Low LCN** value of 5000 and a **High LCN** value of 5010, then the **Low** and **High LCN** values are adjusted downward to 4085 and 4095, respectively. The actual number of **LCNs** that you desire (in this case, 5010 - 5000 = 10) remains unchanged (4095 - 4085 = 10).

Action: Modify `config`; refer to the *Configuration Guide.*

---

init failed on slot <#>. 5 Meg ACE required.

Meaning: X.25 service is unavailable on **slot <#>** because this slot contains a 2Mb ACE board. X.25 service requires a 5Mb ACE board.

Action: Install a 5Mb ACE board in **slot <#>**.

---

ioctl error xx occurred in state yy

Meaning: A status request of the packet-level interface has generated an error.

read error xx occurred in state xx

Meaning: A read of the packet-level interface has generated an error.

switch call - <xxx> to <xxx>

Meaning: An incoming Call Request has been switched.

switched call reset [slot n]. clr = clearing code diag = diagnostic

Meaning: A virtual circuit has been reset.

switched VC clear requested

Meaning: A switched virtual circuit is being cleared.

switched VC close timeout detected - retrying [slot n]

Meaning: An internal close request has been lost between slots.

switched VC reset timeout detected - retrying [slot n]

Meaning: An internal reset request has been lost between slots.

<vc_name> is already disabled

Meaning: A user has issued a `DIS[able]` command for the already disabled virtual circuit, **<vc_name>**.

<vc_name> is already enabled

Meaning: A user has issued an `E[nable]` command for the already enabled virtual circuit, **<vc_name>**.

<vc_name> is disabled

Meaning: Indicates that virtual circuit **<vc_name>** is disabled.

<vc_name> is enabled

Meaning: Indicates that virtual circuit **<vc_name>** is enabled and ready to establish or receive a call.

write error xx occurred in state yy

Meaning: A write of the packet level-interface has generated an error.

Table 10-1 Cause Field Codes

Code	Explanation
DCE-generated	
1	Number busy
3	Invalid facility request
5	Network congestion
9	Out of order
11	Access barred
13	Not obtainable
17	Remote procedure error
19	Local procedure error
21	RPOA out of order
25	Reverse charging not available
33	Incompatible destination
41	Fast select not available
57	Ship absent
DTE-generated	
129	Number busy
131	Invalid facility request
133	Network congestion
137	Out of order

Table 10-1 (Continued)Cause Field Codes

Code	Explanation
DCE-generated	
139	Access barred
141	Not obtainable
145	Remote procedure error
147	Local procedure error
149	RPOA out of order
153	Reverse charging not available
161	Incompatible destination
169	Fast select not available
185	Ship absent

Table 10-2 Diagnostic Codes -- Point-to-Point Service

Code	Explanation
0	No additional information
1	Invalid P(S)
2	Invalid P(R)
3 - 15	Not assigned
16	Packet type invalid
17	For state r1
18	For state r2
19	For state r3
20	For state p1
21	For state p2

Table 10-2 (Continued)Diagnostic Codes -- Point-to-Point Service

Code	Explanation
0	No additional information
22	For state p3
23	For state p4
24	For state p5
25	For state p6
26	For state p7
27	For state d1
28	For state d2
29	For state d3
30 - 31	Not assigned
32	Packet not allowable
33	Unidentifiable packet
34	Call on one-way logical channel
35	Invalid packet type on permanent svc
36	Packet on unassigned logical channel
37	Reject not subscribed to
38	Packet too short
39	Packet too long
40	Invalid general format identifier (GFI)
41	Restart or registration packet with non-zero values in inappropriate bits
42	Packet type not compatible with facility
43	Unauthorized interrupt confirmation
44	Unauthorized interrupt

Table 10-2 (Continued)Diagnostic Codes -- Point-to-Point Service

Code	**Explanation**
0	No additional information
45	Unauthorized reject
46 - 47	Not assigned
48	Time expired
49	for incoming call
50	for clear indication
51	for reset indication
52	for restart indication
53 - 63	Not assigned
64	Call Set Up, Call Clearing, Registration problem
65	Facility/registration code not allowed
66	Facility parameter not allowed
67	Invalid called address
68	Invalid calling address
69	Invalid facility/registration length
70	Incoming call barred
71	No logical channel available
72	Call collision
73	Duplicate facility request
74	Non-zero address length
75	Non-zero facility length
76	Facility not provided when expected
77	Invalid CCITT-specified DTE facility

Table 10-2 (Continued)Diagnostic Codes -- Point-to-Point Service

Code	Explanation
0	No additional information
78 - 79	Not assigned
80	Miscellaneous
81	Improper Cause code from DTE
82	Mis-aligned octet
83	Inconsistent Q-bit setting
84 - 111	Not assigned
112	International problem
113	Remote network problem
114	International protocol problem
115	International link out-of-order
116	International link busy
117	Transit network facility problem
118	Remote network facility problem
119	International routing problem
120	Temporary routing problem
121	Unknown DNIC
122	Maintenance action
123 - 127	Not assigned
241	Call cleared -- disabled circuit or failure in X.25 levels 1, 2, or 3
242	Call cleared because svc was disabled

Table 10-3 Diagnostic Codes -- DDN Service

Code	Explanation
84	Invalid EE error code received
85	Invalid (out-of-range) PSN number
86	Software error available.
128	DCE dropped ready line; network forwarding mechanisms unavailable
129	Link level sent BREAK
130	Link came up
131	Link went down
132	Remote DTE restarted
133	Local resources not available for call establishment
134	Remote resources not available for call establishment
135	Remote call collision
136	Remote host dead
137	Remote DCE dead
138	Logical subnet access barred. Remote DTE cannot be reached because of a communities-of-interest prohibition.
139	Connection lost
140	Response lost
141	Calling logical name not authorized or enabled
142	Calling logical name incorrect for this DTE
143	Called logical name not authorized
144	Called logical name not enabled
145	Called logical name has no effective translations
146	Invalid address; logical addressing not used in this network

Table 10-3 (Continued)Diagnostic Codes -- DDN Service

Code	Explanation
84	Invalid EE error code received
147	Declared logical name is now enabled
148	Declared logical name was already enabled
149	Declared logical name is now disabled
150	Declared logical name was already disabled
151	Incoming calls are barred
152	Outgoing calls are barred
153	Cause field is non-zero
154	VC timeout because of idleness (in calls between X.25 and AHIP hosts
155	Destination DTE uses Standard X.25 service
156	Invalid protocol ID (in calls between X.25 and AHIP hosts)
157	Error occurred while opening connection at the AHIP source
160	PVC endpoints are incompatible
161	NAS reselection completed while local DCE was waiting for RESET CONFIRMATION from the local DCE
162	No response by DTE after attempt to bring up virtual circuit
163	PVC is up and restart is complete
164	Network-caused PVC error
165	CPS aggregation deadlock was detected
166	Local DCE received invalid packet while waiting for response to call request from remote DTE
167	Connection closed because of network error

Table 10-4 Diagnostic Codes -- Reserved for Network-Specific Information

Code	Explanation
168	Cannot intercept fast select
169	Cannot intercept RPOA
170	Cannot intercept X.75 call
171	Too much data in intercepted call
172	Bad NAS address
173	Invalid facility for normal CLEAR packet
174	Invalid local reselect address
175	Invalid remote local reselection address
176	Reselection request with fast select
177	Too much data in reselect request
178	Reselect request NUI is greater than fast select
179	Cannot renegotiate fixed facilities
180	Invalid packet received during NAS select
181	Call opened while local DCE was waiting for reply to a CALL REQUEST
181	Call opened while local DCE was waiting for reply to a CALL REQUEST from remote DTE and RESET CONFIRMATION from local DTE
192	Call cleared because of local pre-emption by a higher-precedence connection
193	Call cleared because of remote pre-emption by a higher-precedence connection
194	Requested precedence is too high
195	PVC take-up collision
196	Remote end-point of the PVC is not initialized with specified LCN
197	Hunt groups are not used
198	Hunt group number is not valid

Table 10-4 (Continued) Diagnostic Codes -- Reserved for Network-Specific Information

Code	Explanation
168	Cannot intercept fast select
199	No port in hunt group is available
200	SVC was killed by MC command
201	PVC was reset by MC command
202	Call redirection took too long or too many tries
205	Call cut off because the precedence level was too low

Table 10-5 Diagnostic Codes -- BFE Information

Code	Explanation
224	Entering emergency mode
225	Leaving emergency mode
226	Emergency window is open
227	Call failed because address translation information is required
228	Call failed because emergency window was not open

11 Using Management Information Base Variables

This chapter describes how to access and interpret the management information base (MIB) and its variables.

11.1 Accessing the Management Information Base

The management information base is a repository for all data (variables) gathered and used by the node. Its structure somewhat resembles that of the UNIX file system. It can be represented as a tree-like, branched structure emanating downward from an unnamed root.

Below the root is a configuration-dependent number of "managed objects," defined as software resources that enable network services. Table 11-1 lists all managed objects, along with associated mnemonics. The actual number of managed objects resident within the node is configuration-dependent.

Table 11-1 Managed Objects

Managed Object	Mnemonic
AppleTalk	at
AppleTalk	atmib
Alarms	alarm
Boot Devices	boot
Bridge	lb
Bridge Forwarding Table	lbmib
Buffers	buf
Circuits	cct
Configuration Management	config
DECnet Router	drs
DECnet Routing Table	decnet
Device Drivers	driver
DMAP	dmap
Echo	echo
Exterior Gateway Protocol	egp
Hardware	hw
Internet MIB	mib
IP Router	ip
Key	key
Lines	line
Memory	mem
Name	name
OSPF	ospf
Simple Network Management Protocol	snmp
SVC	svc
System Manager	mgr
TCP Echo Service	echo
TELNET	telnet
TFTP	tftp
Timers	timer
Transmission Control Protocol	tcp
Xerox Router	xrs
IPX Router	ipx
x25	x25

Each managed object serves as a sub-root for an information base of object-specific managed objects. An arbitrary number of branches emanating from each managed object lead to additional sub-roots. These sub-roots further define instances of the managed object, or lead to a single variable or to a variable set that may be compared to the leaves of the information-base tree.

For example, Figure 11-1 illustrates the structure of the ***buf*** information base. This collection of variables describes the node's use of global memory buffers.

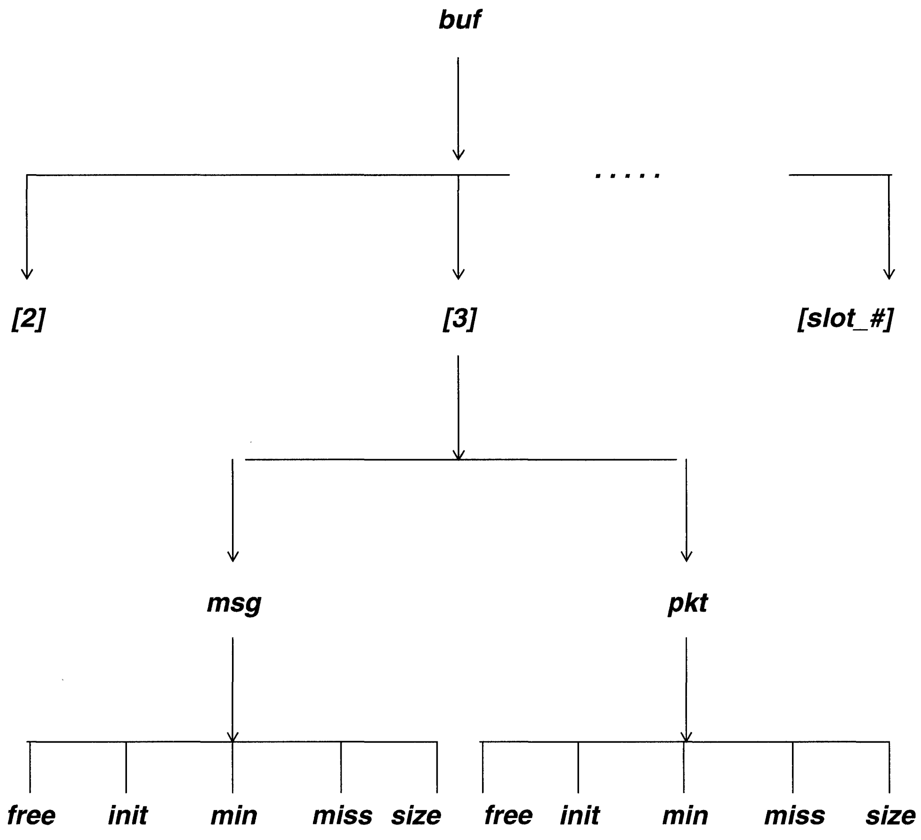


Figure 11-1 *buf* Information Base

As shown in Figure 11-1, the node maintains the ***buf*** information base as a four-level tree. The top-most level of the tree specifies the managed object (***buf***). An arbitrary, configuration-dependent number of branches (one for each ACE board installed in the node) descend from ***buf*** to identify specific object instances (***slot_#***). Two more branches descend from each ***slot_#*** to distinguish two buffer types (***msg*** or ***pkt***). Multiple branches descend from each ***msg*** or ***pkt*** to identify specific variables.

To access a specific variable within the management information base, you must supply the complete variable name. Variable names mirror the hierarchical structure of the information base, and consist of the sequence of managed objects identifiers and managed object instance identifiers that traverse the management information tree.

These variable name components are separated by dots, or enclosed by square brackets.

For example, the ***size*** variable contains the buffer size in bytes. To determine the size of packet buffers on the ACE board in Slot 3, you would supply the following path:

buf[3].pkt.size

Table 11-2 lists all NCL commands that provide access to and/or manipulate the management information base.

Table 11-2 Information Base Access Commands

Command	Function
`GET`	Obtain the value of an information base variable
`LIST`	Display information base structure
`RESET`	Reset (set to 0) the value of an information base variable

11.1.1 Displaying the Information Base

You use the `LIST` command to display the structure of the management information base, as follows:

Syntax: `list {identifier}↵`

or

`l {identifier}↵`

where:

`identifier`

Is the optional information-base pathname identifying an information base root or sub-root. `LIST` displays all the information-tree branches that descend from **{identifier}**. In the absence of an argument, `LIST` displays all node-resident managed objects.

Example: `list↵`

Displays a list of node-resident managed objects.

`l ip.*↵`

Displays the ***ip*** (IP Router) management information base.

The **`LIST`** command displays all or a portion of the management information base structure. It tells you what specific variables the information base contains, and provides you with complete path names to these variables. After obtaining the path name with **`LIST`**, you can use the NCL **`GET`** command to obtain actual variable values (for more information, refer to the section entitled *Obtaining the Value of an Information Base Variable*).

You can use the asterisk character (*) as a wildcard with the **`LIST`** command to display managed, object-specific information bases. Table 11-3 shows how to use **`LIST`** to obtain complete information-base displays for application software modules.

Table 11-3 LIST Command Syntax

Software Module	**Syntax**
DECnet Phase IV Router	`list drs.*↵`
DoD IP Router	`list ip.*↵`
Bridge	`list lb.*↵`
Xerox Router	`list xrs.*↵`
IPX Router	`list ipx.*↵`
AppleTalk Router	`list at.*↵`

The number of managed objects within a node is configuration-dependent. To obtain a list of managed objects resident within the node, type

`list↵` or **`l↵`**

at the NCL prompt. Figure 11-2 shows a portion of a list of managed objects as might be generated by the above command.

```
cct     map=0034     code=2
lb      map=0010     code=3
drs     map=0010     code=4
ip      map=0024     code=5
svc     map=0024     code=9
dmap    map=0034     code=10
buf     map=0034     code=11
mem     map=0034     code=13
name    map=0004     code=14
timer   map=0034     code=15
```

Figure 11-2 Sample Managed Objects Display

The leftmost column in Figure 11-2 lists managed objects within the node. The **map=** column contains a hexadecimal number that designates the node slot(s) within which the managed object resides. To translate the map, start with the least significant digit and move to the left, converting each individual digit to its 4-bit binary equivalent. For example the value "24" translates to:

0010 0100

Each binary digit designates a backplane slot, with the least significant digit of the binary representation designating Slot 0. Using the list in Figure 11-2, you can see that the IP router software is installed in Slot 2 and Slot 5.

The right-hand column, **code=**, contains the object identification code that corresponds to the object listed to its left. If you wish to inspect branches emanating from a managed object, use the `LIST` command followed by either the managed object identifier (from the leftmost column of Figure 11-2) or the object identification code for the managed object (the rightmost column of Figure 11-2).

The following paragraphs explain how to use the `LIST` command to step through branches of a managed object information base. They guide you through the ***buf*** information base previously depicted in Figure 11-1.

To begin inspection of the ***buf*** information base, type either of the following at the NCL prompt:

```
list buf↵

list 11 ↵
```

Note in Figure 11-2 that `11` is the object identification code for buffers, and can be substituted for the mnemonic ***buf***.

In response to the above command, the console screen displays formatted data similar to the following:

```
[2]    map=0004    code=2
[4]    map=0010    code=4
[5]    map=0020    code=5
```

Each horizontal entry designates a single branch from the ***buf*** sub-root. Referring to Figure 11-2, you can see that the ***map*** value for the ***buf-*** managed object is 34. Converting this value to binary representation yields:

```
0011 0100
```

This value indicates that the ***buf-*** managed object resides in slot numbers 2, 4, and 5.

Each slot number appears as a sub-root (**[2]**, **[4]**, and **[5]**) designating a distinct instance of the managed object.

To inspect branches emanating from a slot sub-root (for example Slot 2), type either of the following at the NCL prompt:

```
list buf[2]↵

list 11.2↵
```

The console screen displays formatted data similar to the following:

```
msg    map=0004    code=0
pkt    map=0004    code=1
```

To inspect branches emanating from a buffer type sub-root (for example **pkt**), type one of the following at the NCL prompt:

```
list buf[2].pkt↵

list 11.2.1↵
```

In response, the console screen displays formatted data similar to the following:

```
init   map=0004    code=1
free   map=0004    code=2
min    map=0004    code=3
miss   map=0004    code=4
size   map=0004    code=5
```

As illustrated in Figure 11-3, ***msg*** and ***pkt*** are the lowest-level sub-roots in the buffers information-base structure. If you attempt to display additional sub-roots by, for example, entering the following command,

```
list 11.2.1.1↵
```

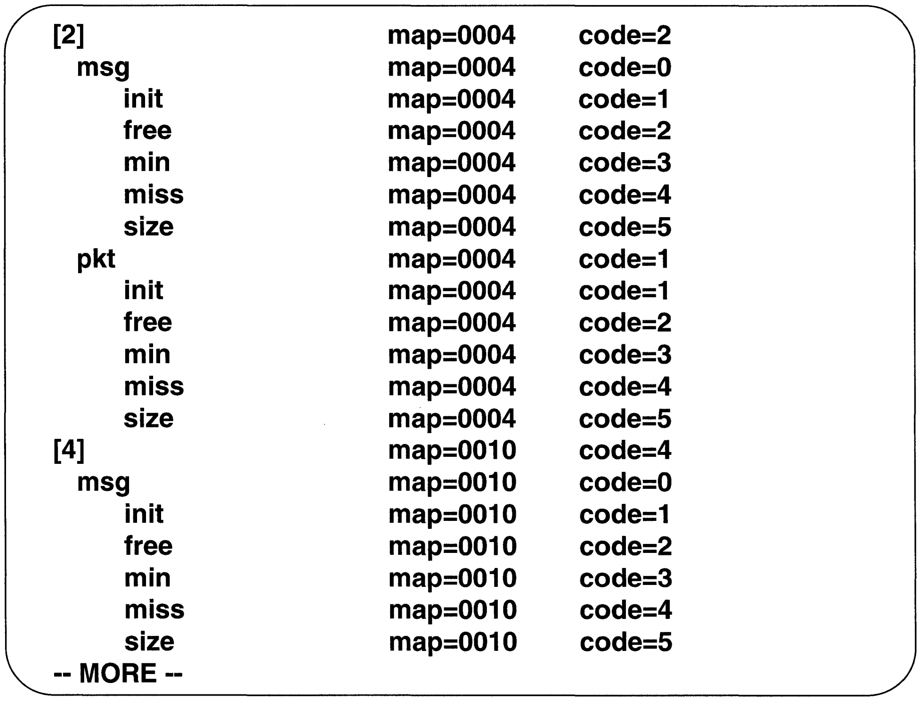
the console screen displays no data and simply returns you to the NCL prompt.

You can use the asterisk character (*) as a wildcard to display the entire ***buf*** information base (or of any other managed-object-specific information base). To display the entire buffers information base, for example, type either of the following at the NCL prompt:

```
list buf.*↵

list 11.*↵
```

In response, the console screen displays a formatted representation of the buffers information base, similar to the sample shown in Figure 11-3.



```
[2]                         map=0004     code=2
  msg                       map=0004     code=0
      init                  map=0004     code=1
      free                  map=0004     code=2
      min                   map=0004     code=3
      miss                  map=0004     code=4
      size                  map=0004     code=5
  pkt                       map=0004     code=1
      init                  map=0004     code=1
      free                  map=0004     code=2
      min                   map=0004     code=3
      miss                  map=0004     code=4
      size                  map=0004     code=5
[4]                         map=0010     code=4
  msg                       map=0010     code=0
      init                  map=0010     code=1
      free                  map=0010     code=2
      min                   map=0010     code=3
      miss                  map=0010     code=4
      size                  map=0010     code=5
-- MORE --
```

Figure 11-3 Sample Buffers Information Base Display

To view additional information base data:

1. **Press `[RETURN]` for one more line.**
2. **Type a number from `1` through `9` to display that number of additional lines.**
3. **Press any other key for an additional screen of data.**
4. **Press the `[LEFTARROW]`(⇐) or `[CTRL/C]` to return to the NCL prompt.**

11.1.2 Obtaining the Value of an Information Base Variable

You use the **`GET`** command to obtain the value of a management information base variable, as follows:

Syntax: `get {identifier}↵`

or

`g {identifier}↵`

where:

`identifier`

Is the optional information-base pathname to the target variable (refer to the section entitled *Displaying the Information Base* for a description of information-base pathnames).

Example: `get telnet.tx_bytes↵`

Retrieves the value of a telnet variable, tx_bytes.

`g lb.lab_net.xmit↵`

Retrieves the value of the variable that contains the number of bridge frames transmitted across the circuit group named ***lab_net***.

When retrieving a variable value, you must provide the complete information-base pathname. You may use object identification codes to identify variables. Refer to the section entitled *Displaying the Information Base* for a description of the use of object identification codes.

You may use the asterisk (*) as a wildcard with **`GET`**. shows how to use **`GET`** to obtain complete information-base variable values for application software modules.

Table 11-4 shows the syntax for the **`GET`** command.

Table 11-4 GET Command Syntax

Software Module	Syntax
DECnet Phase IV Router	`get drs.*↵`
DoD IP Router	`get ip.*↵`
Bridge	`get lb.*↵`
XNS Router	`get xrs.*↵`
IPX Router	`get ipx.*↵`
AppleTalk Router	`get at.*↵`

11.1.3 Resetting the Value of an Information Base Variable

You use the `RESET` command to set the value of a management information base variable or variables to zero, as follows:

Syntax: `set <identifier>↵`

where:

`identifier`

Is the required pathname to the variable.

Example: `reset telnet.tx_bytes↵`

Zeros a telnet variable, ***tx_bytes***.

`reset lb.lab_net.xmit↵`

Zeros the variable that records the number of Spanning Tree Bridge frames transmitted on the circuit group named **lab_net**.

When resetting a variable, you must provide a complete variable pathname. You may use object identification codes to identify variables. Refer to the section entitled *Displaying the Information Base* for a description of object identification codes.

You may use the asterisk (*) as a wildcard with **RESET**. For example, the following command resets all variables associated with the circuit named jrb:

```
reset cct.jrb.*↵
```

11.2 Accessing the Internet MIB

Internet Request for Comments 1156 defines the variable set required for monitoring and controlling various components of the Internet. The node's MIB implementation is fully compliant with all requirements of RFC 1156.

A series of NCL commands work in conjunction with the Simple Network Management Protocol (SNMP) management-agent software and the IP router software to provide access to the Internet MIB. You can use these commands to examine the MIB of a local or remote network node.

Table 11-5 lists all NCL commands that provide access to the standard Internet MIB.

Table 11-5 Internet MIB Access Commands

Command	Function
RGETS	Obtain the value of an Internet MIB variable
RGETMS	Obtain the value of a class of Internet MIB variables
RGETA	Obtain the MIB Address Translation Table
RGETI	Obtain the MIB IP Address Table
RGETR	Obtain the MIB IP Routing Table

11.2.1 Obtaining the Value of a MIB Variable

You use the **RGETS** command to obtain the value of an individual MIB variable from a remote network node, as follows:

Syntax: `rgets <identifier> <addr> {comm}↵`

where:

`identifier`

Is the required pathname to the Internet MIB variable (defined in RFC 1156); **identifier** does not include the Internet MIB prefix (***1.3.6.1.2.1***).

`addr`

Is the required IP address (in dotted decimal notation) of the local or remote target node.

`comm`

Is the optional name of the SNMP community that enables access to the node specified by `<addr>`. In the absence of this argument, RGETS defaults to `public`.

Examples: `rgets 6.4.0  192.32.1.94↵`

Retrieves the value of the Internet MIB variable ***1.3.6.1.2.1.6.4.0*** (maximum number of TCP connections) from the node whose IP address is 192.32.1.94. In response, the console screen displays:

1.3.6.1.2.1.6.4.0 = n

where **n** is the maximum number of TCP connections.

`rgets 7.1.0  192.32.2.194↵`

Retrieves the Internet MIB variable ***1.3.6.1.2.1.7.1.0*** (number of UDP datagrams delivered to UDP users) from the node whose IP address is 192.32.2.194.

In response, the console screen displays:

1.3.6.1.2.1.7.1.0 = n

where **n** is the number of UDP datagrams.

You can use either the **RGETS** or **GET** command to obtain the value of an individual MIB variable from the node.

11.2.2 Obtaining the Values of a MIB Variable Class

You use the **RGETMS** command to obtain the value of a class of MIB variables from either a local or remote network node, as follows:

Syntax: `rgetms <identifier> <addr> {comm}↵`

where:

`identifier`
: Is the required pathname (defined in RFC 1156) to the first variable within the MIB class; `identifier` does not include the standard Internet MIB prefix (***1.3.6.1.2.1***).

`addr`
: Is the required IP address (in dotted decimal notation) of the local or remote target node.

`comm`
: Is the optional name of the SNMP community that enables access to the node specified by `<addr>`. In the absence of this argument, **RGETMS** defaults to `public`.

Example: `rgetms 7 192.32.1.94↵`

Retrieves the values of the Internet MIB User Datagram Protocol (UDP) variable class from the node whose IP address is 192.32.1.94. In response, the console screen displays:

```
1.3.6.1.2.1.7.1.0 = n1
1.3.6.1.2.1.7.2.0 = n2
1.3.6.1.2.1.7.3.0 = n3
1.3.6.1.2.1.7.4.0 = n4
```

where:

n1 is the number of UDP datagrams delivered to UDP users.

n2 is the number of received UDP datagrams for which there was no application at the destination port.

n3 is the number of undeliverable UDP datagrams.

n4 is number of transmitted UDP datagrams.

`rgetms 6.13 192.32.2.194↵`

Retrieves the Internet MIB Transmission Control Protocol (TCP) connection table from the node whose IP address is 192.32.2.194 (see Figure 11-4).

```
1.3.6.1.2.1.6.13.1.1.192.32.2.194.23.192.32.1.167.1665=n1
1.3.6.1.2.1.6.13.1.2.192.32.2.194.23.192.32.1.167.1665=n2
1.3.6.1.2.1.6.13.1.3.192.32.2.194.23.192.32.1.167.1665=n3
1.3.6.1.2.1.6.13.1.4.192.32.2.194.23.192.32.1.167.1665=n4
1.3.6.1.2.1.6.13.1.5.192.32.2.194.23.192.32.1.167.1665=n5
```

Figure 11-4 Sample Internet MIB TCP Connection Table

The values shown in Figure 11-4 are as follows:

- **n1** reflects the TCP connection state.
- **n2** identifies the local IP address for the TCP connection.
- **n3** identifies the local port number for the TCP connection.
- **n4** and **n5** identify the remote IP address and remote port number for the TCP connection.

11.2.3 Obtaining the MIB IP Address Translation Table

You use the `RGETA` command to obtain a formatted version of the Internet MIB Address Translation Table for a local or a remote network node, as follows:

Syntax: `rgeta <addr> {comm}↵`

where:

`addr`

Is the required IP address (in dotted decimal notation) of the local or remote target node.

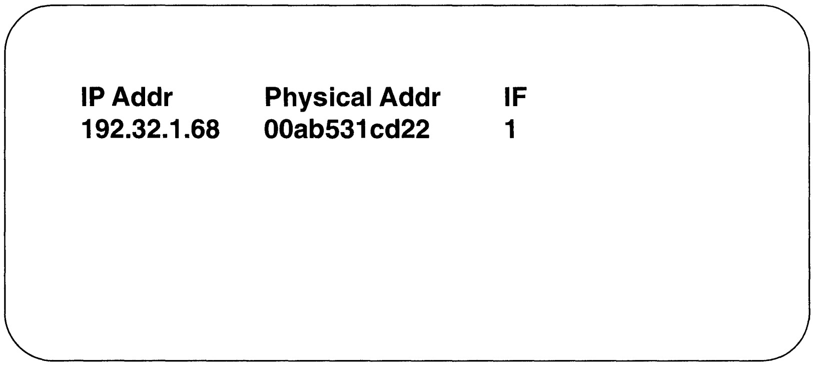
`comm`

Is the optional name of the SNMP community that enables access to the node specified by `<addr>`. In the absence of this argument, `RGETA` defaults to `public`.

Example: `rgeta 192.32.1.94↵`

Displays the Internet MIB address translation table for the node whose IP address is 192.32.1.94.

In response, the console screen displays the Address Translation Table. Figure 11-5 shows a sample MIB IP Address Translation Table.



```
IP Addr        Physical Addr     IF
192.32.1.68    00ab531cd22       1
```

Figure 11-5 Sample MIB IP Address Translation Table

Individual fields in Figure 11-5 are as follows:

- **IP Addr** lists the network address (in dotted decimal notation).
- **Physical Addr** lists the MAC-level (physical) address to which **IP Addr** corresponds.
- **IF** lists the node-assigned interface number on which the address translation is in effect.

11.2.4 Obtaining the MIB IP Address Table

You use the `RGETI` command to obtain a formatted version of the Internet MIB IP Address Table for a local or remote network node, as follows:

Syntax: `rgeti <addr> {comm}↵`

where:

`addr`

Is the required IP address (in dotted decimal notation) of the local or remote target node.

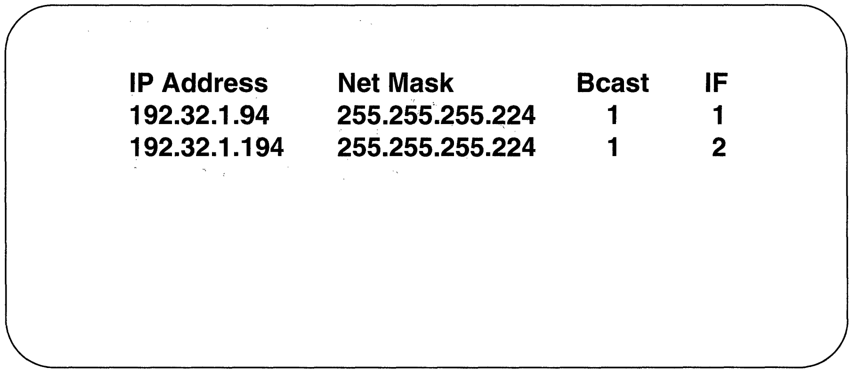
`comm`

Is the optional name of the SNMP community that enables access to the node specified by `<addr>`. In the absence of this argument, `RGETI` defaults to `public`.

Example: `rgeti 192.32.1.94↵`

Displays the Internet MIB IP address table for the node whose IP address is 192.32.1.94.

In response, the console displays the IP Address Table. Figure 11-7 shows a sample MIB IP Address Table.



```
IP Address       Net Mask            Bcast    IF
192.32.1.94      255.255.255.224       1       1
192.32.1.194     255.255.255.224       1       2
```

Figure 11-7 Sample MIB IP Address Table

Individual fields in Figure 11-7 are as follows:

- **IP Address** lists the IP address (in dotted decimal notation).
- **Net Mask** lists the subnet mask (in dotted decimal notation) associated with **IP Address**. The subnet mask is an IP address with all the network bits set to 1, and all the host bits set to 0.
- **Bcast** lists the value of the least significant-bit in the IP broadcast address for **IP Address**.

  In most cases the broadcast address type (all 0s or all 1s) can be determined by examining a single bit. In the event of an explicitly assigned broadcast address, however, this single bit is not significant.
- **IF** lists the node-assigned value that identifies the interface to **IP Address.**

11.2.5 Obtaining the MIB IP Routing Table

You use the **`RGETR`** command to obtain a formatted version of the Internet MIB IP Routing Table for a local or remote network node, as follows:

Syntax: `rgetr <addr> {comm}↵`

where:

`addr`

Is the required IP address (in dotted decimal notation) of the local or remote target node.

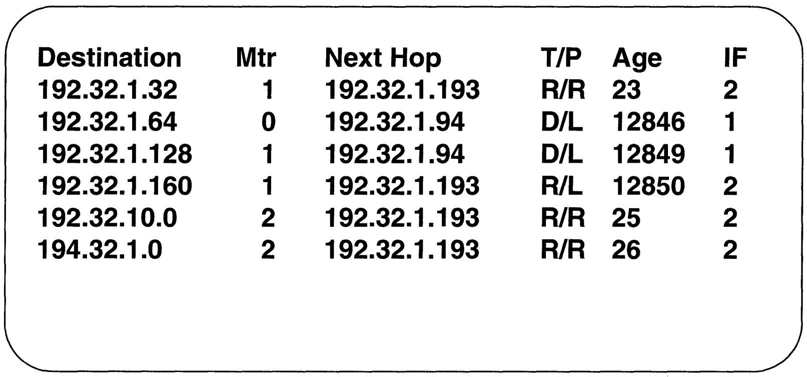
`comm`

Is the optional name of the SNMP community that enables access to the node specified by `<addr>`. In the absence of this argument, RGETR defaults to `public`.

Example: `rgetr 192.32.1.94↵`

Displays the IP routing table for the node whose IP address is 192.32.1.94.

In response, the console displays the IP Routing Table. Figure 11-8 shows a sample MIB IP Routing Table.



```
Destination     Mtr   Next Hop        T/P  Age    IF
192.32.1.32       1   192.32.1.193    R/R  23     2
192.32.1.64       0   192.32.1.94     D/L  12846  1
192.32.1.128      1   192.32.1.94     D/L  12849  1
192.32.1.160      1   192.32.1.193    R/L  12850  2
192.32.10.0       2   192.32.1.193    R/R  25     2
194.32.1.0        2   192.32.1.193    R/R  26     2
```

Figure 11-8 MIB IP Routing Table

Individual fields in Figure 11-8 are as follows:

- **Destination** lists the destination network addresses (in dotted decimal notation).
- **Mtr** lists the cost (hop count) to **Destination** for RIP and EGP; for OSPF, Mtr lists the OSPF cost of the route (for more information, see the section entitled *Displaying the OSPF Routing Table*).
- **Next Hop** lists the address (in dotted decimal notation) of the next hop.
- **T** lists the route type as follows:
  - **D** -- a direct (local) route
  - **I** -- an invalid route
  - **R** -- a remote route
- **P** lists how the route type was learned, as follows:
  - **E** -- Exterior Gateway Protocol
  - **L** -- statically-configured route
  - **R** -- Routing Information Protocol
  - **O** -- Open Shortest Path First
- **Age** lists the number of seconds since the route was learned.
- **IF** lists the node-assigned interface number over which **Next Hop** is reached.

11.3 Accessing a Remote Management Information Base

Two NCL commands work in conjunction with the Simple Network Management Protocol (SNMP) management-agent software and the IP router software to provide access to the management information base of a remote peer. Whereas the previous commands in this chapter access either a local or remote, standard Internet MIB, the commands in this section access the Wellfleet proprietary MIB of a remote FN, LN, or CN. Table 11-6 lists these NCL commands.

Table 11-6 Peer Management Information Base Remote Access Commands

Command	Function
`RGETW`	Obtain the value of a single management information base variable from a remote peer node
`RGETMW`	Obtain the value of a class of management information base variables from a remote peer node

11.3.1 Obtaining the Value of a Remote Variable

You use the **`RGETW`** command to obtain the value of a single management information base variable from a remote peer node, as follows:

Syntax: **`rgetw <identifier> <addr> {comm}↵`**

where:

`identifier`

Is the required object pathname to the peer management information base variable; **`identifier`** does not include the Wellfleet private enterprise prefix (1.3.6.1.4.1.18.1.1).

`addr`

Is the required IP address (in dotted decimal notation) of the remote peer node.

`comm`

Is the optional name of the SNMP community that enables access to the node specified by **`<addr>`**. In the absence of this argument, RGETW defaults to **`public`**.

Examples: **`rgetw 11.2.1.1  192.32.1.94↵`**

Retrieves the Wellfleet management information base variable 11.2.1.1 (the number of packet buffers allocated at node initialization) from the peer node whose IP address is 192.32.1.94.

In response, the console screen displays:

1.3.6.1.4.1.18.1.1.11.2.1.1 = n

where **n** is the number of allocated packet buffers.

```
rgetw 2.1.1  192.32.2.194↵
```

Retrieves the Wellfleet management information base variable 2.1.1 (octets received without error on circuit #1) from the peer node whose IP address is 192.32.2.194.

In response, the console screen displays:

1.3.6.1.4.1.18.1.1.2.1.1 = n

where **n** is the number of octets received without error.

11.3.2 Obtaining the Values of a Remote Variable Class

You use the `RGETMW` command to obtain the value of a class of management information base variables from a remote peer node, as follows:

Syntax: `rgetmw <identifier> <addr> {comm}↵`

where:

`identifier`

Is the required object identification path to the first variable within the peer private enterprise management information base class; `identifier` does not include the Wellfleet private enterprise prefix (1.3.6.1.4.1.18.1.1).

`addr`

Is the required IP address (in dotted decimal notation) of the remote peer node.

`comm`

Is the optional name of the SNMP community that enables access to the node specified by `<addr>`. In the absence of this argument, `RGETMW` defaults to `public`.

Examples: `rgetmw 30.2  192.32.1.94↵`

Retrieves the values of the Wellfleet private enterprise management information base variable class 30.2 (hardware data for the ACE in Slot 2) from the peer node whose IP address is 192.32.1.94.

In response, the console screen displays:

```
1.3.6.1.4.1.18.1.1.30.2.1 = n1
1.3.6.1.4.1.18.1.1.30.2.2 = n2
1.3.6.1.4.1.18.1.1.30.2.3 = n3
1.3.6.1.4.1.18.1.1.30.2.4 = n4
1.3.6.1.4.1.18.1.1.30.2.5 = n5
```

where:

n1 is the ACE serial number.

n2 is the ACE revision level.

n3 is the Link interface module serial number.

n4 is the link interface module revision number.

n5 is the link interface module ID number.

```
rgetmw 21.4  192.32.2.194↵
```

Retrieves the values of the Wellfleet private enterprise management information base variable class 21.4 (data for the T1 line emanating from Slot 4) from the peer node whose IP address is 192.32.2.194.

In response, the console screen displays:

```
1.3.6.1.4.1.18.1.1.21.4.1 = n1
1.3.6.1.4.1.18.1.1.21.4.2 = n2
1.3.6.1.4.1.18.1.1.21.4.3 = n3
1.3.6.1.4.1.18.1.1.21.4.4 = n4
1.3.6.1.4.1.18.1.1.21.4.5 = n5
1.3.6.1.4.1.18.1.1.21.4.6 = n6
1.3.6.1.4.1.18.1.1.21.4.7 = n7
```

where

n1 is the number of instances of carrier loss.

n2 is the number of received Red alarms.

n3 is the number of received Yellow alarms.

n4 is the number of bi-polar violations.

n5 is the number of frame alignment violations.

n6 is the number of faulty superframes received.

n7 is the number of bit errors.

11.4 Accessing a Foreign Management Information Base

Two NCL commands work in conjunction with the Simple Network Management Protocol (SNMP) management-agent software and the IP router software to provide access to the corporate (private) management information base of a remote foreign (non-Wellfleet) node. lists these NCL commands.

Table 11-7 Foreign Management Information Base Remote Access Commands

Command	Function
`RGET`	Obtain the value of a single management information base variable from a remote foreign node
`RGETM`	Obtain the value of a class of management information base variables from a remote foreign node

11.4.1 Obtaining the Value of a Foreign Variable

You use the **RGET** command to obtain the value of a single corporate management information base variable from a remote foreign node, as follows:

Syntax: `rget <identifier> <addr> {comm}↵`

where:

`identifier`

Is the required pathname to the target variable.

`addr`

Is the required IP address (in dotted decimal notation) of the remote foreign node.

`comm`

Is the optional name of the SNMP community that enables access to the node specified by **<addr>.** In the absence of this argument, **RGET** defaults to **public**.

Example: `rget 1.3.6.1.4.1.N.n.n.n <ip_addr>↵`

Retrieves the value of a private enterprise management information base variable from a remote foreign node whose IP address is `<ip_addr>`.

`1.3.6.1.4.1` designates the path to the private enterprise sub-tree.

`N` designates the specific corporate identifier as provided by the Internet Assigned Number authority.

`n.n.n` designates the object identification path assigned by the equipment manufacturer.

11.4.2 Obtaining the Values of a Foreign Variable Class

You use the **RGETM** command to obtain the value of a corporate management information base class of variables from a remote foreign node, as follows:

Syntax: **`rgetm <identifier> <addr> {comm}↵`**

where:

`identifier`

Is the required object identification path to the first variable within the foreign private enterprise management information base class.

`addr`

Is the required IP address (in dotted decimal notation) of the remote foreign node.

`comm`

Is the optional name of the SNMP community that enables access to the node specified by **`<addr>`**. In the absence of this argument, **RGETM** defaults to **`public`**.

Example: `rget 1.3.6.1.4.1.N.n.n.n <ip_addr>↵`

Retrieves the value of a private enterprise management information base variable class from a remote foreign node whose IP address is `<ip_addr>`.

`1.3.6.1.4.1` designates the path to the private enterprise sub-tree.

`N` designates the specific corporate identifier as provided by the Internet Assigned Number authority.

`n.n.n` designates the object identification path assigned by the equipment manufacturer.

12 Management Information Base Variables, A to H

This chapter provides descriptions of the objects (and variables), beginning with the letters *a* through *h*, contained in the management information base.

12.1 AppleTalk Information Base

The ***at*** (AppleTalk) information base contains variables that describe transmission and reception activities across each AppleTalk circuit group; it also contains variables that describe the rejection of certain packets by the AppleTalk router.

The ***at*** information base is a 4-level tree. The top-most level identifies the ***at*** managed object. The next lower level consists of a configuration-dependent number of branches, with each branch identifying a specific AppleTalk circuit group (***cg_name***). The next level consists of six branches, with each branch corresponding to a specific AppleTalk protocol. The lowest level contains the actual ***at*** variables.

Figure 12-1 illustrates the structure of the ***at*** information base.

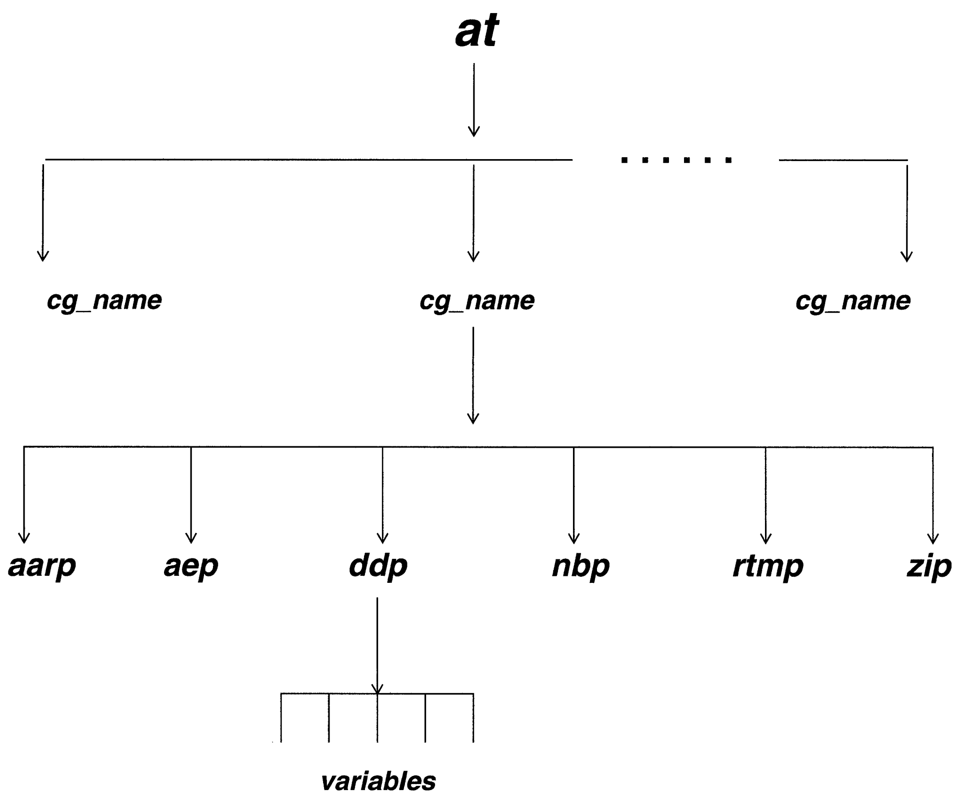


Figure 12-1 ***at*** **Information Base**

You use the NCL `LIST` command to display all, or a portion, of the ***at*** information base, and the NCL `GET` command to obtain the value of any variable within the ***at*** information base.

You express the pathname to any ***at*** variable in the following format:

at.cg_name.at_protocol.variable_name

where:

at

Is the ***at*** managed object.

cg_name

Is the AppleTalk circuit group name.

at_protocol

Is the specific AppleTalk protocol, as follows:

AARP	AppleTalk Address Resolution Protocol
AEP	AppleTalk Echo Protocol
DDP	Datagram Delivery Protocol
NBP	Name Binding Protocol
RTMP	Routing Table Maintenance Protocol
ZIP	Zone Information Protocol

variable_name

Is the variable name.

12.1.1 AARP Variables

The AppleTalk Address Resolution Protocol variables, in alphabetical order, are as follows:

amt_overflow

Contains the number of times **<cg_name>** tried unsuccessfully to store a system address and its corresponding physical address in the AARP Mapping Table.

probe_rx

Contains the number of AARP PROBE packets received by **<cg_name>**.

probe_tx

Contains the number of AARP PROBE packets transmitted by **<cg_name>**.

req_rx

Contains the number of AARP REQUEST packets received by **<cg_name>**.

req_tx

Contains the number of AARP REQUEST packets transmitted by **<cg_name>**.

rsp_rx

Contains the number of AARP RESPONSE packets received by **<cg_name>.**

rsp_tx

Contains the number of AARP RESPONSE packets transmitted by **<cg_name>.**

12.1.2 AEP Variables

The AppleTalk Echo Protocol variables, in alphabetical order, are as follows:

reply_tx

Contains the number of AEP REPLY packets transmitted by **<cg_name>**.

req_rx

Contains the number of AEP REQUEST packets received by **<cg_name>**.

12.1.3 DDP Variables

The AppleTalk Datagram Delivery Protocol variables, in alphabetical order, are as follows:

ddp_bad_cksum

Contains the number of AppleTalk packets dropped by **<cg_name>** because the packet contained an incorrect DDP checksum value.

ddp_fwd

Contains the number of AppleTalk packets forwarded by **<cg_name>**.

ddp_hop_ct_exceed

Contains the number of AppleTalk packets dropped by **<cg_name>** because the packet's hop count was too large.

ddp_no_ir_addr

Contains the number of AppleTalk packets dropped by **<cg_name>** because the next router's physical address could not be resolved.

ddp_pkts_dropped_no_aarp_rsp

Contains the number of AppleTalk packets dropped by **<cg_name>** because the packet's destination system did not respond to AARP REQUEST packets issued by the router.

ddp_pkts_sent_by_aarp

Contains the number of AppleTalk packets forwarded after the destination address was resolved by AARP.

ddp_rx

Contains the number of valid AppleTalk packets received by **<cg_name>**.

ddp_total_drop

Contains the total number of AppleTalk packets dropped by **<cg_name>**.

ddp_unknown_netwk

Contains the number of AppleTalk packets dropped by **<cg_name>** because the destination network was unknown.

ddp_upper_protocol

Contains the number of AppleTalk packets sent to an upper-layer protocol by **<cg_name>**.

12.1.4 NBP Variables

The AppleTalk Name Binding Protocol variables, in alphabetical order, are as follows:

nbp_breq_rx

Contains the number of NBP BROADCAST REQUEST packets received by **<cg_name>**.

nbp_fwdreq_rx

Contains the number of NBP FORWARD REQUEST packets received by **<cg_name>**.

nbp_fwdreq_tx

Contains the number of NBP FORWARD REQUEST packets transmitted by **<cg_name>**.

nbp_lkup_tx

Contains the number of NBP LOOKUP packets transmitted by **<cg_name>**.

12.1.5 RTMP Variables

The AppleTalk Routing Table Maintenance Protocol variables, in alphabetical order, are as follows:

cable_range_conflicts

Contains the number of times an RTMP DATA packet (received on **<cg_name>**) contained a routing table (a target network and a hop count) with a network range that overlapped a routing entry in the AppleTalk Routing Table.

data_rx

Contains the number of RTMP DATA packets received by **<cg_name>**.

data_tx

Contains the number of RTMP DATA packets transmitted by **<cg_name>**.

network_type_conflicts

Contains the number of RTMP DATA packets (received on **<cg_name>**) whose routing tables conflicted with network entries in the AppleTalk Routing Table.

nonextended_netwk

Contains the number of "non-extended" routing tables received by **<cg_name>**.

rdr_rx

Contains the number of RTMP ROUTE DATA REQUEST packets received by **<cg_name>**.

req_rx

Contains the number of RTMP REQUEST packets received by **<cg_name>**.

routing_tbl_overflow

Contains the number of times **<cg_name>** tried unsuccessfully to store a new routing table in the AppleTalk Routing Table.

rsp_tx

Contains the number of RTMP RESPONSE packets transmitted by **<cg_name>**.

ver_mismatch

Contains the number of RTMP DATA packets (received by **<cg_name>**) that were not for AppleTalk Phase 2.

12.1.6 ZIP Variables

The AppleTalk Zone Information Protocol variables, in alphabetical order, are as follows:

getlclzones_rx

Contains the number of ZIP GETLOCALZONES packets received by **<cg_name>**.

getlclzones_tx

Contains the number of ZIP GETLOCALZONES packets transmitted by **<cg_name>**.

getlclzonesreply_rx

Contains the number of ZIP GETLOCALZONESREPLY packets received by **<cg_name>**.

getlclzonesreply_tx

Contains the number of ZIP GETLOCALZONESREPLY packets transmitted by **<cg_name>**.

getnetinfo_rx

Contains the number of ZIP GETNETINFO packets received by **<cg_name>**.

getnetinfo_tx

Contains the number of ZIP GETNETINFO packets transmitted by **<cg_name>**.

getzonelist_rx

Contains the number of ZIP GETZONELIST packets received by **<cg_name>**.

getzonelistreply_tx

Contains the number of ZIP GETZONELISTREPLY packets transmitted by **<cg_name>**.

netinforeply_rx

Contains the number of ZIP NETINFOREPLY packets received by **<cg_name>**.

netinforeply_tx

Contains the number of ZIP NETINFOREPLY packets transmitted by **<cg_name>**.

reply_rx

Contains the number of ZIP REPLY packets received by **<cg_name>**.

reply_tx

Contains the number of ZIP REPLY packets transmitted by **<cg_name>**.

req_rx

Contains the number of ZIP REQUEST packets received by **<cg_name>**.

req_tx

Contains the number of ZIP REQUEST packets transmitted by **<cg_name>**.

12.2 AppleTalk MIB Information Base

The ***atmib*** (AppleTalk management information base) contains variables that describe aggregate transmission and reception activities of the AppleTalk router.

The ***atmib*** information base is a 3-level tree. The top-most level identifies the ***atmib*** managed object. The middle level consists of five branches, with each branch corresponding to a specific AppleTalk protocol. The lowest level contains the actual ***atmib*** variables.

Figure 12-2 illustrates the structure of the ***atmib*** information base.

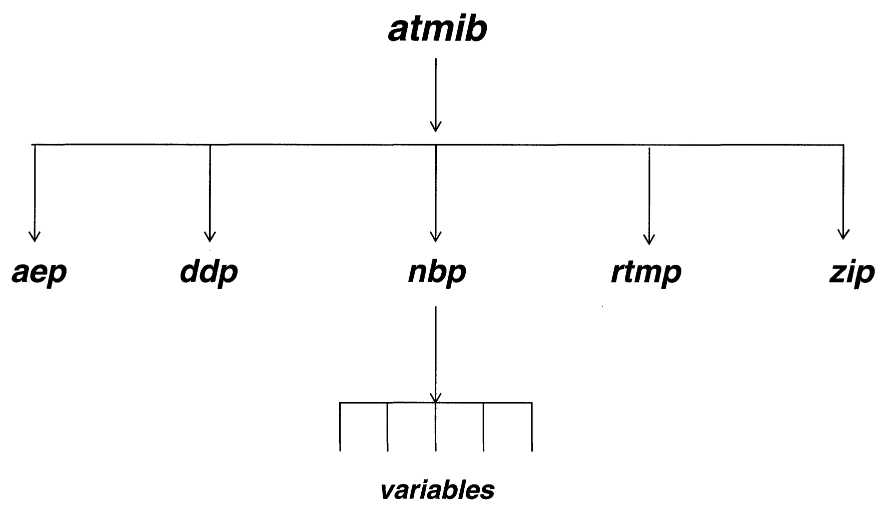


Figure 12-2 ***atmib*** **Information Base**

You use the NCL `LIST` command to display all, or a portion, of the ***atmib*** information base, and the NCL `GET` command to obtain the value of any variable within the ***atmib*** information base.

You express the pathname to any ***atmib*** variable in the following format:

atmib.at_protocol.variable_name

where:

atmib

Is the ***atmib*** managed object.

at_protocol

Is the specific AppleTalk protocol, as follows:

AEP	AppleTalk Echo Protocol
DDP	Datagram Delivery Protocol
NBP	Name Binding Protocol
RTMP	Routing Table Maintenance Protocol
ZIP	Zone Information Protocol

variable_name

Is the variable name.

12.2.1 AEP Variables

The AppleTalk Echo Protocol variables, in alphabetical order, are as follows:

aep_Reply_tx

Contains the total number of AEP REPLY packets transmitted by the AppleTalk router.

aep_Req_rx

Contains the total number of AEP REQUEST packets received by the AppleTalk router.

12.2.2 DDP Variables

The Datagram Delivery Protocol variables, in alphabetical order, are as follows:

ddp_bad_cksum

Contains the total number of AppleTalk packets dropped by the AppleTalk router because the packet contained an incorrect DDP checksum value.

ddp_fwd

Contains the total number of AppleTalk packets forwarded by the AppleTalk router.

ddp_hop_ct_exceed

Contains the total number of AppleTalk packets dropped by the AppleTalk router because the packet's hop count was too large.

ddp_rx

Contains the total number of AppleTalk packets received by the AppleTalk router.

ddp_total_drop

Contains the total number of AppleTalk packets dropped by the AppleTalk router.

ddp_tx

Contains the total number of AppleTalk packets transmitted by the AppleTalk router.

ddp_unknown_netwk

Contains the total number of AppleTalk packets dropped by the AppleTalk router because the destination network was unknown.

ddp_upper_prot

Contains the total number of AppleTalk packets sent to an upper-layer protocol by the AppleTalk router.

12.2.3 NBP Variables

The Name Binding Protocol variables, in alphabetical order, are as follows:

nbp_breq_rx

Contains the total number of NBP BROADCAST REQUEST packets received by the AppleTalk router.

nbp_fwdreq_rx

Contains the total number of NBP FORWARD REQUEST packets received by the AppleTalk router.

nbp_fwdreq_tx

Contains the total number of NBP FORWARD REQUEST packets transmitted by the AppleTalk router.

nbp_lkup_tx

Contains the total number of NBP LOOKUP packets transmitted by the AppleTalk router.

12.2.4 RTMP Variables

The Routing Table Maintenance Protocol variables, in alphabetical order, are as follows:

rtmp_req_rx

Contains the total number of RTMP REQUEST packets received by the AppleTalk router.

rtmp_rsp_tx

Contains the total number of RTMP RESPONSE packets transmitted by the AppleTalk router.

12.2.5 ZIP Variables

The Zone Information Protocol variables, in alphabetical order, are as follows:

zip_getlclzones_rx

Contains the total number of ZIP GETLOCALZONES packets received by the AppleTalk router.

zip_getlclzones_tx

Contains the total number of ZIP GETLOCALZONES packets transmitted by the AppleTalk router.

zip_getlclzonesreply_rx

Contains the total number of ZIP GETLOCALZONESREPLY packets received by the AppleTalk router.

zip_getlclzonesreply_tx

Contains the total number of ZIP GETLOCALZONESREPLY packets transmitted by the AppleTalk router.

zip_getnetinfo_rx

Contains the total number of ZIP GETNETINFO packets received by the AppleTalk router.

zip_getnetinfo_tx

Contains the total number of ZIP GETNETINFO packets transmitted by the AppleTalk router.

zip_getzonelist_rx

Contains the total number of ZIP GETZONELIST packets received by the AppleTalk router.

zip_netinforeply_rx

Contains the total number of ZIP NETINFOREPLY packets received by the AppleTalk router.

zip_netinforeply_tx

Contains the total number of ZIP NETINFOREPLY packets transmitted by the AppleTalk router.

zip_reply_rx

Contains the total number of ZIP REPLY packets received by the AppleTalk router.

zip_reply_tx

Contains the total number of ZIP REPLY packets transmitted by the AppleTalk router.

zip_req_rx

Contains the total number of ZIP REQUEST packets received by the AppleTalk router.

zip_req_tx

Contains the total number of ZIP REQUEST packets transmitted by the AppleTalk router.

12.3 Alarm Information Base

The ***alarm*** information base contains variables that describe the scheduling and issuance of system-generated alarms.

The ***alarm*** information base is a 3-level tree. The top-most level identifies the ***alarm*** managed object. The middle level consists of a configuration-dependent number of branches, with each branch identifying a slot number (***slot_#)***. The bottom level contains the actual ***alarm*** variables.

Figure 12-3 illustrates the structure of the ***alarm*** information base.

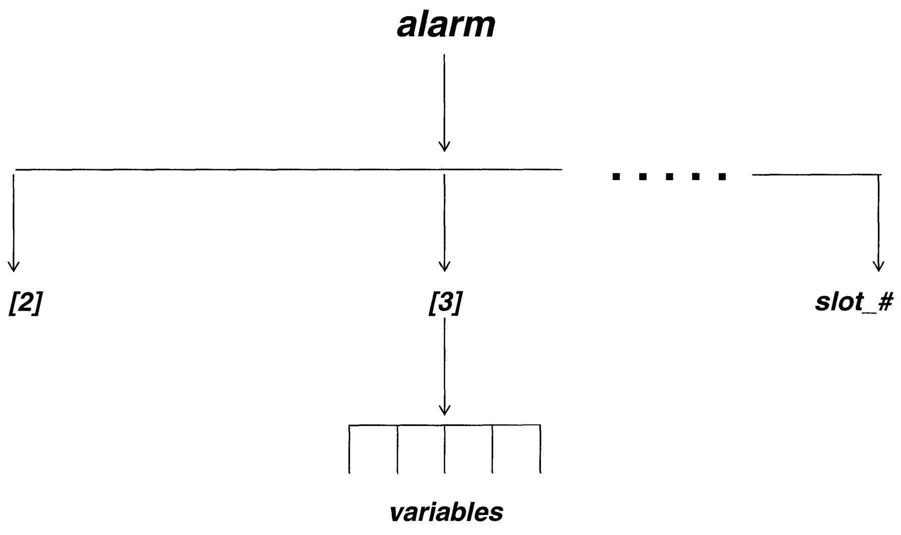


Figure 12-3 ***alarm*** **Information Base**

You use the NCL `LIST` command to display all, or a portion of, the ***alarm*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***alarm*** variable in the following format:

alarm[slot_#].variable_name

where:

alarm

Is the ***alarm*** managed object.

[slot_#]

Is the slot number (allowable range is 2 to 14 depending on system type).

variable_name

Is the variable name.

The ***alarm*** variables, in alphabetical order, are as follows:

cancel_cnt

Contains the number of alarms cancelled by the system prior to the expiration of the alarm timer.

expire_cnt

Contains the number of alarm-related interrupts. Such an interrupt is generated when the alarm timer reaches zero.

race_cnt

Contains the number of simultaneous occurrences of the expiration of the alarm timer and the cancellation of a previously scheduled alarm.

set_cnt

Contains the number of alarms scheduled by the system.

12.4 Boot Information Base

The ***boot*** information base contains multiple instances (one for each active system slot) of a single control object, ***boot,*** the sole function of which is to generate system management messages.

The NCL `LIST` and `GET` commands provide no additional information regarding the ***boot*** information base.

12.5 Bridge Information Base

The ***lb*** (learning bridge) information base contains variables that describe the reception and transmission of packets across each circuit group associated with the bridge.

The ***lb*** information base is a 3-level tree. The top-most level identifies the bridge managed object. The middle level consists of a configuration-dependent number of branches, with each branch identifying a specific circuit group (***cg_name***). The bottom level contains the actual ***lb*** variables.

Figure 12-4 illustrates the structure of the ***lb*** information base.

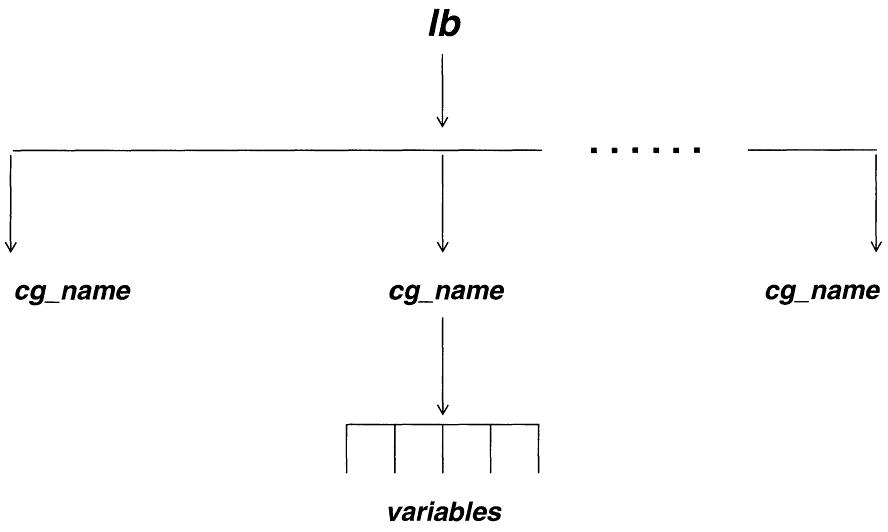


Figure 12-4 *lb* Information Base

You use the NCL `LIST` command to display all, or a portion of, the ***lb*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***lb*** variable in the following format:

lb.cg_name.variable_name

where:

lb

Is the ***lb*** managed object.

cg_name

Is the circuit group name.

variable_name

Is the variable name.

The ***lb*** variables, in alphabetical order, are as follows:

---

drop_dst_addr

Contains the number of packets dropped by circuit group **<cg_name>** in accordance with global destination address filters specified within the `config` file.

---

drop_invalid_ri

Contains the number of specifically routed-frames dropped by **<cg_name>** because the routing path contained a duplicate LAN ID.

---

drop_dst_local

Contains the number of packets dropped by circuit group **<cg_name>** because the source and destination address were on the same (local) network.

---

drop_listen

Contains the number of packets dropped by circuit group **<cg_name>** while it was in the spanning-tree Learning state. While in the Learning state, **<cg_name>** receives both network-generated bridge protocol data units (BPDUs) and end- station-generated traffic. This traffic is subjected to the learning process but not relayed. Time spent in the Learning state is governed by the Forward Delay parameter. Upon expiration of the forward delay timer, circuit group **<cg_name>** enters the Forwarding state.

---

drop_loadbal_noprotcf

Contains the number of packets dropped by circuit group **<cg_name>** because the protocol contained in the Type field did not match the expected protocol value.

drop_no_cg_from_cgm

Contains the number of packets dropped by circuit group **<cg_name>** because of a change in state of the circuit group during processing time. This variable usually indicates that a circuit group has gone down (been disabled) during packet processing.

drop_protocol

Contains the number of packets dropped by circuit group **<cg_name>** in accordance with global or local protocol filters specified by the `config` file.

drop_src_addr

Contains the number of packets dropped by circuit group **<cg_name>** in accordance with local source address filters specified by the `config` file.

flood

Contains the number of packets flooded by circuit group **<cg_name>**; **<cg_name>** floods packets if it has not yet learned the location of the packet's destination address.

fwd_dst_addr

Contains the number of packets forwarded by circuit group **<cg_name>** in accordance with global destination address filters specified by the `config` file.

fwd_load_bal

Contains the number of packets forwarded by circuit group **<cg_name>** in accordance with load balancing options specified by the `config` file.

fwd_mcast_addr

Contains the number of packets forwarded by circuit group **<cg_name>** in accordance with global multicast address filters specified by the `config` file.

fwd_protocol

Contains the number of packets forwarded by circuit group **<cg_name>** in accordance with global or local protocol filters specified by the `config` file.

recv

Contains the number of packets received by circuit group **<cg_name>**.

recv_cfg

Contains the number of configuration BPDUs received by circuit group **<cg_name>**.

recv_tcn

Contains the number of topology change notification BPDUs received by circuit group **<cg_name>**.

srbcast_fwd

Contains the total number of all paths, broadcasting routing and spanning-tree, broadcast-routing frames forwarded on circuit group **<cg_name>**.

srbcast_rx

Contains the total number of all paths broadcasting routing and spanning-tree, broadcast-routing frames received on circuit group **<cg_name>**.

srf_dropnotinroute

Contains the number of specifically routed frames dropped on circuit group **<cg_name>** because the bridge was not contained in the routing path.

srf_fwd

Contains the number of specifically routed frames forwarded on circuit group **<cg_name>**.

srf_rx

Contains the number of specifically routed frames received on circuit group **<cg_name>**.

xmit

Contains the number of packets transmitted by circuit group **<cg_name>**.

xmit_cfg

Contains the number of configuration BPDUs sent by circuit group **<cg_name>**.

xmit_tcn

Contains the number of topology change notification BPDUs sent by circuit group **<cg_name>**.

12.6 Bridge Forwarding/Filtering Table Information Base

The ***lbmib*** (bridge forwarding/filtering table) information base contains data on the forwarding and filtering of bridge frames.

The system maintains ***lbmib*** information as a table containing an arbitrary number of entries. The Bridge software adds table entries as it learns the locations of physical addresses. Each table entry consists of five fields.

Figure 12-5 illustrates the structure of the ***lbmib*** table.

address	***src***	***dst***	***cg***	***if***
address	***src***	***dst***	***cg***	***if***
address	***src***	***dst***	***cg***	***if***
⋮	⋮	⋮	⋮	⋮
address	***src***	***dst***	***cg***	***if***
address	***src***	***dst***	***cg***	***if***
address	***src***	***dst***	***cg***	***if***

Figure 12-5 *lmib* Table Structure

You use the NCL **RGETB** command to access the ***lbmib*** table.

The individual entries in the ***lmib*** table, in alphabetical order, are as follows:

address

Contains a physical address (expressed as a 12-digit hexadecimal number) of a connected LAN device.

cg

Contains the name of the circuit group that provides a connection to ***address***.

dst

Contains the disposition of frames which contain ***address*** in the destination address field of the Ethernet header. **F** indicates that frames are forwarded; **D** indicates that frames are dropped.

if

Contains the system-assigned interface number that corresponds to **<cg>**.

src

Contains the disposition of frames which contain ***address*** in the source address field of the Ethernet header. **F** indicates that frames are forwarded; **D** indicates that frames are dropped.

12.7 Buffer Information Base

The ***buf*** (buffer) information base contains variables that describe the system's use of two types of global memory buffers: ***message buffers*** that facilitate internal communications taking place over the VME bus; and ***packet buffers*** that facilitate external communications by temporarily storing incoming or outgoing data packets.

The system maintains ***buf*** information as a 4-level tree. The top-most level identifies the ***buf*** managed object. The next lower level consists of a configuration-dependent number of branches, with each branch identifying a slot number (***slot_#***). The next level consists of two branches, with each branch identifying a buffer type (***msg*** or ***pkt)***. The lowest level contains the actual ***buf*** variables. The information base contains the same variables for both message and packet buffers.

Figure 12-6 illustrates the structure of the ***buf*** information base.

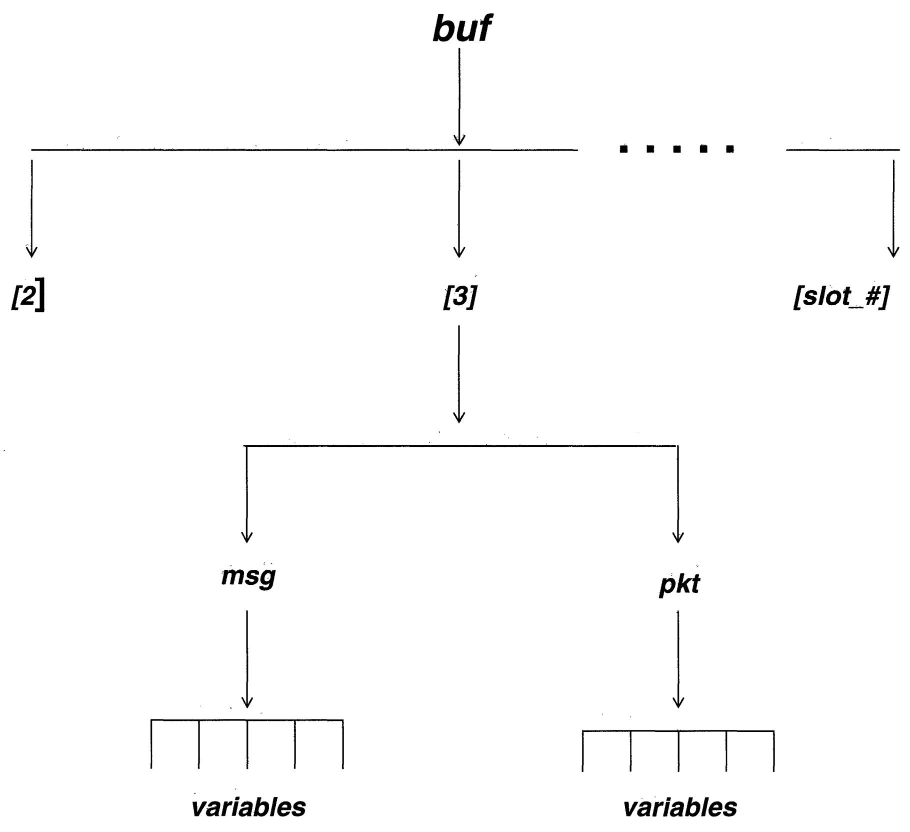


Figure 12-6 *buf* Information Base

You use the NCL `LIST` command to display all, or a portion of, the ***buf*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***buf*** variable in the following format:

buf[slot_#].type.variable_name

where:

buf

Is the ***buf*** managed object.

[slot_#]

Is the slot number (allowable range is 2 to 14 depending on system type).

type

Is the buffer type (***msg*** for message buffers or ***pkt*** for packet buffers).

variable_name

Is the variable name.

The ***buf*** variables, in alphabetical order, are as follows:

free

Contains the number of message or packet buffers on **[slot_#]** available for internal VME transfers or for external transfers, respectively. Because system operations and application software modules impose some overhead on global memory buffers, the number of buffers available for data transfers is less than the total number of buffers allocated when the system boots.

init

Contains the number of message or packet buffers on **[slot_#]** allocated when the system booted.

min

Contains the smallest number of message or packet buffers available on **[slot_#]** since the system booted.

miss

Contains the number of times that the system was unable to obtain either a message buffer or a packet buffer on **[slot_#]**. Failure to obtain a buffer indicates that all buffers were busy. This parameter is directly related to ***min.*** If ***miss*** is greater than 0, ***min*** must equal 0. Conversely, if ***miss*** equals 0, then ***min*** must be greater than 0.

size

Contains the size of the message or packet buffer in bytes.

12.8 Circuit Information Base

The ***cct*** (circuit) information base contains variables that describe transmit and receive activities across each frame relay, LAN, point-to-point, and SMDS circuit. (Transmit and receive activities across X.25 Point-to-Point, DDN, and PDN circuits and X.25 switched virtual circuits are described by the ***x25*** information base.)

The system maintains ***cct*** information as a 3-level tree. The top-most level identifies the ***cct*** managed object. The middle level consists of a configuration-dependent number of branches, with each branch identifying a circuit (***cct_name***). The bottom level contains the actual ***cct*** variables. The system records similar, but not identical, variables for LAN and point-to-point circuits.

Figure 12-7 illustrates the structure of the ***cct*** information base.

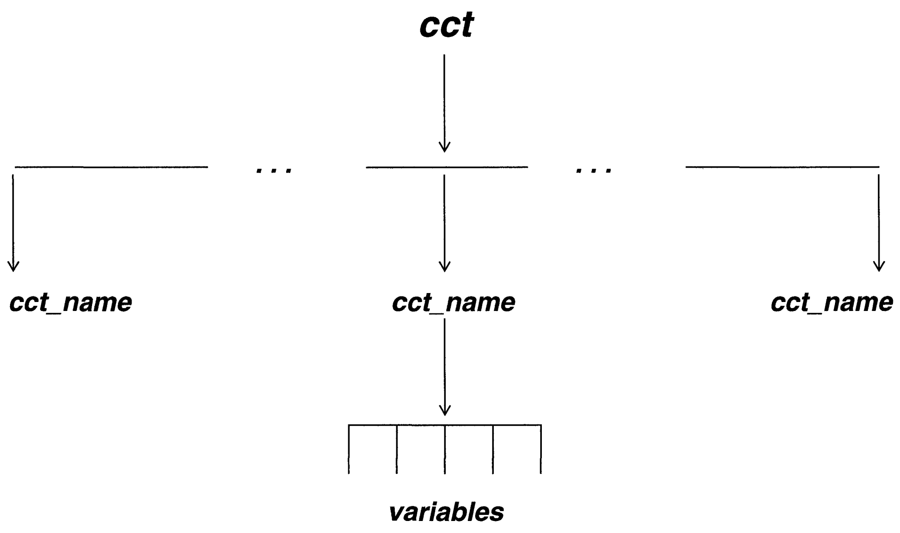


Figure 12-7 *cct* Information Base

You use the NCL **`LIST`** command to display all, or a portion of, the ***cct*** information base, and the NCL **`GET`** command to obtain the value of any variable within the information base.You express the pathname to any ***cct*** variable in the following format:

cct.cct_name.variable_name

where:

cct

Is the ***cct*** managed object.

cct_name

Is the circuit name.

variable_name

Is the variable name.

12.8.1 Frame Relay Management Information Base

The ***frame relay*** information base is composed of three tables:

- Data Link Connection Management Interface (Dlcmi) Table
- Circuit Table
- Error Table

The Management Information Base for Frame Relay contains the values for these tables, which are described in the following sections.

The Frame Relay MIB tables are organized under the experimental MIB ("exmib") under the number 18 in a 6-level tree. The top-most level identifies the ***frame relay*** managed object. The next lower level contains the three tables: Dlcmi, circuit, and error. The next two levels identify the table entries and variables for the Dlcmi and error tables, while the circuit table levels contain circuit values, Dlcmi values, and below that, the variables. Figure 12-8 shows this organization.

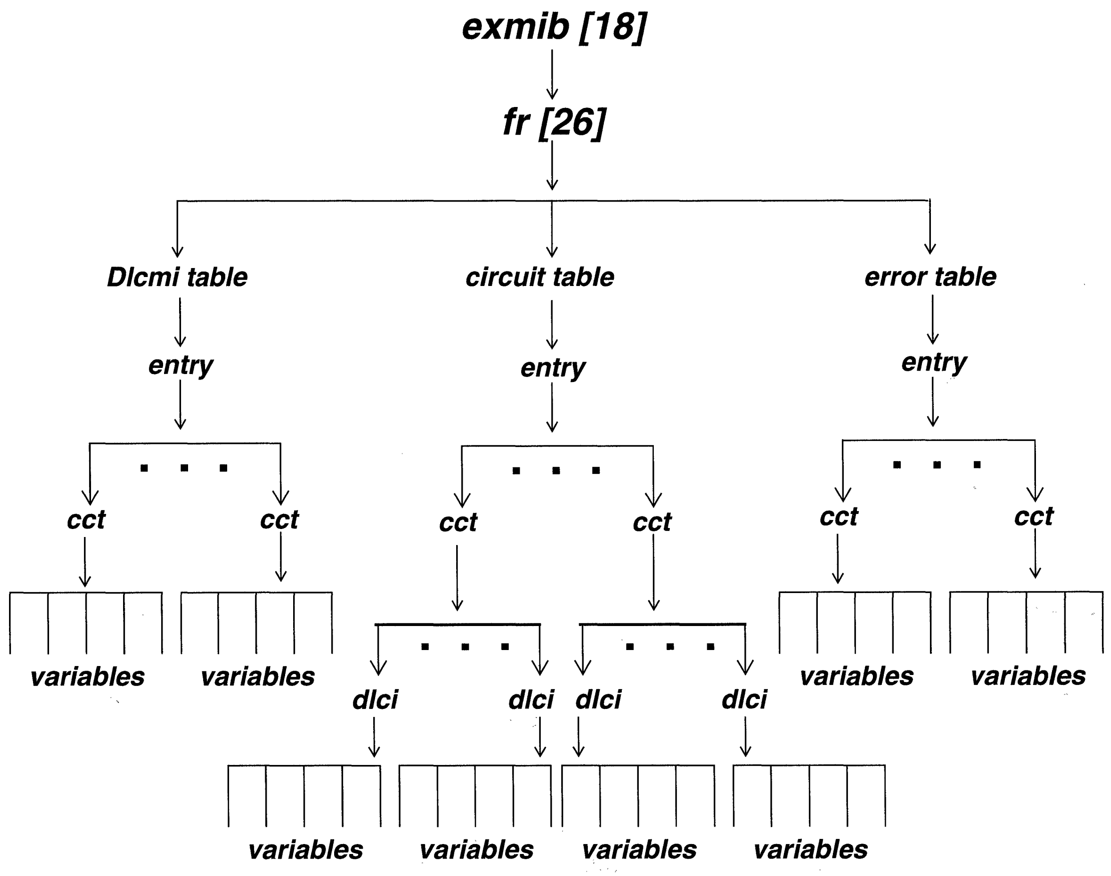


Figure 12-8 ***frame relay*** **Information Base**

Each Frame Relay MIB item may be accessed through Network Command Language (NCL) from the console screen. You use the NCL **`LIST`** command to display all or a portion of the Frame Relay information base, and you use the NCL **`GET`** command to obtain the value of any variable within the information base.There are two methods of access, one using the names of the branches, and the other using the numbers.

exmib.fr.<table name>.<table entry>

where:

<table name>

Is the name of the ***fr*** table

<table entry>

Is the actual value of the table entry.

Or, using the second method of access:

26.26.1.1.5.8

where

8

Is the Frame Relay circuit number, which is system-assigned.

5

Is the MIB variable.

12.8.1.1 Data Link Connection Management Interface Table

The Frame Relay MIB variables, in alphabetical order, for the Data Link Connection Management Interface Table are as follows:

frDfrDlcmiActive

States which Data Link Connection Management scheme is active and, by implication, which DLCI is uses, on the Frame Relay interface. There are four possible values:

Disabled (1) indicates that there is not interface management running on this interface.

LmiRev1 (2) is the original Local Management Interface (LMI).

ansiT1-617-D (3) indicates the use of ANSI Annex D of ANSI T1-617.

ansiT1-617-B (4) indicates the use of ANSI Annex D of ANSI T1-617 (not supported by Wellfleet).

Access methods:

exmib.fr.DlcTble.entry.active

26.26.1.1.2.<cct>

where ***<cct>*** indicates the circuit.

frDlcmiAddress

States which address format is in use on this Frame Relay interface. There are four possible values:

Q921 (1) specifies use of the first Frame Relay address format with support for a 13-bit DLCI with no DE, FECN, or BECN bit support. It is mostly obsolete with 2-byte support only.

Q922March90 (2) specifies use of March 1990 draft of CCITT specification Q922 with support for an 11-bit DLCI with no DE bit.

Q922November90 (3) specifies use of November 90 draft of CCITT specification Q922 with support for an 10-bit DLCI with support for DE, FECN, and BECN bits.

Q9222 (4) specifies use of the final draft of CCITT specification Q922 with support for all of Q922November90 plus a control byte in the extended address format.

Access methods:

xmib.fr.DlcTble.entry.addr

26.26.1.1.3.<cct>

where ***<cct>*** indicates the circuit.

frDlcmiAddressLen

States the address length, in octets, to be used over this interface. By agreement, all PVCs within a given interface must use the same encoding and encoding length for the DLCI (addresses). In the case of Q922 format, the length indicates the entire length of the address including the control portion.

exmib.fr.DlcTble.entry.addrlen

26.26.1.1.4.<cct>

where ***<cct>*** indicates the circuit.

frDlcmiErrorThreshold

Is the number of errors within a given monitored number of events necessary to cause the interface to be shut down. The default is three errors.

exmib.fr.DlcTble.entry.errthr

26.26.1.1.7.<cct>

where ***<cct>*** indicates the circuit.

frDlcmiIfIndex

Is the index into the Dlcmi Table. It corresponds to the Frame Relay circuit for which table information is requested.

exmib.fr.DlcTble.entry.index

26.26.1.1.1.<cct>

where ***<cct>*** indicates the circuit.

frDlcmiFullEnquiryInterval

Is the number of status enquiry intervals that pass before issuance of a full status enquiry message. The default is six intervals.

exmib.fr.DlcTble.entry.enqInt

26.26.1.1.6.<cct>

where ***<cct>*** indicates the circuit.

frDlcmiMaxSupportedPVCs

Indicates the maximum number of Permanent Virtual Circuits (PVCs) allowed for this interface. This is usually dictated by the Frame Relay network.

exmib.fr.DlcTble.entry.maxpvc

26.26.1.1.9.<cct>

where ***<cct>*** indicates the circuit.

frDlcmiMonitored Events

Is the number of events for which errors are collected in order to monitor for an interface shutdown due to excessive errors. The default is 4 events.

exmib.fr.DlcTble.entry.monevnt

26.26.1.1.8.<cct>

where ***<cct>*** indicates the circuit.

frDlcmiMulticast

Indicates whether the Frame Relay provider offers a Multicast Service. There are two possible values for this field:

nonBroadcast (1) for those networks that do not support multicasting, and

broadcast (2) for those that do.

exmib.fr.DlcTble.entry.multcast

26.26.1.1.10.<cct>

where ***<cct>*** indicates the circuit.

frDlcmiPollingInterval

Is the number of seconds between successive status enquiry messages. The default value is 10 seconds.

exmib.fr.DlcTble.entry.pollint

26.36.1.1.5.<cct>

where ***<cct>*** indicates the circuit.

12.8.1.2 Frame Relay Circuit Table Variables

The circuit table contains information about the various connections (PVCs) that are presently defined within the Frame Relay interface. The Frame Relay MIB variables, in alphabetical order, for the Circuit Table are as follows:

frCircuitBECNreceived

Is the number of frames received from the network indicating backward congestion since the circuit was created.

exmib.fr.CctTbl.entry.BECN.<cct>.<ddd>

26.26.2.1.5.<cct>.<ddd>

where ***<cct>*** indicates the circuit and ***<ddd>*** indicates the dlci.

frCircuitCreationTime

Is the value of sysUpTime when the circuit was created, whether by the Data Link Connection Management or by a set request.

exmib.fr.CctTbl.entry.crtd.<cct>.<ddd>

26.26.2.1.8.<cct>.<ddd>

where ***<cct>*** indicates the circuit and ***<ddd>*** indicates the dlci.

frCircuitDlci

Is the Data Link Connection Identifier for this circuit.

exmib.fr.CctTbl.entry.dlci

26.26.2.1.2.<cct>.<ddd>

where ***<cct>*** indicates the circuit and ***<ddd>*** indicates the dlci.

frCircuitFECNreceived

Is the number of frames received from the network indicating forward congestion since the circuit was created.

exmib.fr.CctTbl.entry.fecn.<cct>.<ddd>

26.26.2.1.4.<cct>.<ddd>

where ***<cct>*** indicates the circuit and ***<ddd>*** indicates the dlci.

frCircuitFramesReceived

Is the number of frames received over this circuit since it was created.

exmib.fr.CctTbl.entry.rcvd.<cct>.<ddd>

26.26.2.1.7.<cct>.<ddd>

where ***<cct>*** indicates the circuit and ***<ddd>*** indicates the dlci.

frCircuitFramesSent

Is the number of frames sent from this circuit since it was created.

exmib.fr.CctTbl.entry.sent.<cct>.<ddd>

26.26.2.1.6.<cct>.<ddd>

where ***<cct>*** indicates the circuit and ***<ddd>*** indicates the dlci.

frCircuitIfIndex

Is the circuit corresponding to the Frame Relay interface.

exmib.fr.CctTbl.entry.index

26.26.2.1.1.<cct>

where ***<cct>*** indicates the circuit.

frCircuitLastTimeChange

Is the value of sysUpTime when last there was a change in the circuit state.

exmib.fr.CctTbl.entry.change.<cct>.<ddd>

26.26.2.1.9.<cct>.<ddd>

where ***<cct>*** indicates the circuit and ***<ddd>*** indicates the dlci.

frCircuitOctetsReceived

Indicates the number of octets received over this circuit since it was created.

exmib.fr.CctTbl.entry.o_rcvd.<cct>.<ddd>

26.26.1.9.<cct>.<ddd>

where ***<cct>*** indicates the circuit and ***<ddd>*** indicates the dlci.

frCircuitOctetsSent

Indicates the number of octets sent from this circuit since it was created.

exmib.fr.CctTbl.entry.o_sent.<cct>.<ddd>

26.26.2.1.7<cct>.<ddd>

where ***<cct>*** indicates the circuit and ***<ddd>*** indiates the dlci.

frCircuitState

Indicates whether the particular circuit is operational. These entries are created by the Data Link Connection Management exchange. There are three possible values for this entry:

Invalid (1) is used to delete an entry manually.

Active (2) is an indication that the circuit is up and ready for use.

Inactive (3) is used to show that the circuit is present but not ready for use.

exmib.fr.CctTbl.entry.state.<cct>.<ddd>

26.26.2.1.3.<cct>.<ddd>

where ***<cct>*** indicates the circuit and ***<ddd>*** indiates the dlci.

12.8.1.3 Frame Relay Error Table Variables

This table describes errors encountered on the Frame Relay interface. It is indexed by circuit number. The error table variables, in alphabetical order, are as follows:

---

frErrData

Is an octet string containing as much of the error packet as possible. As a minimum, it must contain the two octets of the DLCI.

exmib.fr.ErrTbl.entry.data.<cct>

26.26.3.1.3.<cct>

where ***<cct>*** indicates the circuit.

---

frErrIndex

Is the index into the error table which corresponds to the circuit number.

exmib.fr.ErrTbl.entry.index.<cct>

26.26.3.1.1.<cct>

where ***<cct>*** indicates the circuit.

---

frErrTime

Is the value of ***sysUpTime*** at which the error was detected.

exmib.fr.ErrTbl.entry.time.<cct>

26.26.3.1.4.<cct>

where ***<cct>*** indicates the circuit.

frErrType

Is the type of error that was last seen on this interface. The following values are valid for this field.

unknownErr	***(1)***
receiveShort	***(2)***
receiveLong	***(3)***
illegalDLCI	***(4)***
unknownDLCI	***(5)***
lmiProtoErr	***(6)***
lmiUnknownIE	***(7)***
lmiSequenceErr	***(8)***
lmiUnknownRpt	***(9)***

exmib.fr.ErrTbl.entry.type.<cct>

26.26.3.1.2.<cct>

where ***<cct>*** indicates the circuit.

12.8.2 LAN Circuit Variables

The ***LAN*** circuit variables, in alphabetical order, are as follows:

alig_error_rx

Contains the number of non-aligned frames received by circuit **<cct_name>**. A non-aligned frame does not end on a byte boundary.

babl_error_tx

Contains the number of babbles on circuit **<cct_name>**. A babble occurs when the transmitter portion of the Local Area Network Controller for Ethernet (LANCE) transmits an Ethernet or IEEE 802.3 frame containing more than 1518 bytes (this byte count does not include the 64-bit frame preamble and synchronization bits). In such an instance, the transmitter completes sending the entire frame, and increments this counter.

buferr_tx

Contains the number of transmit buffer errors. A transmit buffer error indicates corruption of the transmit descriptor ring associated with circuit **<cct_name>**. Usually such corruption takes the form of a break in the data chain (a series of pointers to multiple buffers that contain consecutive portions of a lengthy frame). A transmit buffer error disables the transmitter portion of the LANCE.

cerr

Contains the number of transceiver self-test failures on circuit **<cct_name>**. Some transceivers assert the collision signal during the inter-packet delay period to verify the channel between the transceiver and the LANCE. Such self-tests are usually referred to as SQE (Signal Quality Error) or heartbeat. A test failure (defined as the absence of a collision signal within 2.0μs of the cessation of transmission) may indicate transceiver malfunction or a faulty transmission path between the LANCE and the transceiver.

deferred_tx

Contains the number of deferred transmissions on circuit **<cct_name>**. A deferred transmission indicates that the physical medium was busy (the carrier sense signal was **.TRUE.**) when the LANCE had a frame for transmittal. In such an instance, the LANCE waits for the carrier sense signal to go to **.FALSE.**, pauses for an interframe spacing interval, and then attempts to transmit the waiting frame.

dls_ret_rx

Contains the number of frames, received by circuit **<cct_name>**, that were later returned by the data-link service (DLS) software. Within the system software architecture, DLS resides between the driver and application software. It performs such services as multiplexing/ demultiplexing or encapsulation/deencapsulation. DLS can return frames for numerous reasons (many of which are application-specific): for example, because of unknown internal service- access points (ISAPs); because of user-specified filtering requirements contained within the `config` file; or because of lack of enabled entities (for example, IP, DECnet, etc.).

dls_ring_cnt

Contains the current number of packets received by circuit **<cct_name>** and currently in transit to DLS.

excessv_colln_tx

Contains the number of excessive collision errors on circuit **<cct_name>**. An excessive collision error occurs when the LANCE has detected collisions on the medium in 16 successive attempts to transmit a frame. After 16 unsuccessful transmission attempts, the LANCE drops the frame, increments this counter, and transmits the next frame on its transmit queue.

fcs_error_rx

Contains the number of frames received by circuit **<cct_name>** that contained an erroneous checksum.

frames_rx_ok

Contains the number of frames received without error by circuit **<cct_name>**.

frames_tx_ok

Contains the number of frames transmitted without error by circuit **<cct_name>**.

frams_incomp_rx

Contains the number of incomplete frames received by circuit **<cct_name>**. An incomplete frame is identified by failure to set the end-of-packet bit in the receive message descriptor.

hw_flt_drop

Contains the number of frames that were dropped by **<cct_name>** because the hardware filter table indicated that the source and destination address were local.

hw_flt_free

Contains the number of available (empty) entries in the hardware filter table associated with **<cct_name>**. Note that ***hw_flt_size*** minus ***hw_flt_used*** is equal to ***hw_flt_free.***

hw_flt_size

Contains the current capacity of the hardware filter table associated with **<cct_name>**. Filtering table resources are dynamically allocated (in increments of 256 locations) on an as-needed basis.

hw_flt_tbl_size

Contains the maximum size of the hardware filter table associated with **<cct_name>**. Certain Dual Synchronous/Dual Ethernet Link Modules are equipped with a dual Ethernet filter accelerator (DEFA). A DEFA is a gate-array device that reads and stores Ethernet source and destination addresses at wire speed. The appearance of both source and destination addresses within the hardware filter table indicates that the source and destination are on the local network; consequently, the frame is dropped.

hw_flt_used

Contains the number of current entries in the hardware filter table associated with **<cct_name>**.

lack_resc_error_rx

Contains the number of times circuit **<cct_name>** lost frames because of insufficient buffer space.

late_colln_tx

Contains the number of late collisions on circuit **<cct_name>**. A late collision is one detected after the transmission of at least 64 bytes. In such an instance, the LANCE does *not* retransmit the frame. Rather, it terminates the transmission, increments this counter, and transmits the next frame in its transmit queue.

lcar_tx

Contains the number of instances of carrier loss on **<cct_name>**. Loss of carrier indicates that the input (receive-enable signal) went **.FALSE.** during a controller-initiated transmission.

mac_addr

Contains the 12-digit hexadecimal representation of the 48-bit Ethernet address used by circuit **<cct_name>**.

merr

Contains the number of memory errors. A memory error indicates that the LANCE, after becoming bus master, failed to receive a ready signal within 25.6µs of asserting a memory address on the data/address bus. A memory error disables the LANCE.

octets_rx_ok

Contains the number of octets (bytes) received without error by circuit **<cct_name>**.

octets_tx_ok

Contains the number of octets (bytes) transmitted without error by circuit **<cct_name>**.

oflo_rx

Contains the total number of overflows on circuit **<cct_name>**. An overflow occurs when the LANCE could not keep pace with flow of incoming data and lost part or all of an incoming frame.

rcv_desc_cnt

Contains the current number of receiver data buffers available to the LANCE.

test_cmd_rx

Contains the number of 802.2 Logical Link Control (LLC) `Test` commands received by circuit **<cct_name>**. `Test` commands seek to provide a basic verification of the LLC-to-LLC transmission path. Because the system does respond to such commands, the values contained in this variable and in ***test_rsp_tx*** should be equal.

test_cmd_tx

Contains the number of 802.2 Logical Link Control (LLC) `Test` commands issued on circuit **<cct_name>**. Because the system does not provide a user interface to this facility, the value contained in this variable should be 0, as should the value of ***test_rsp_rx.***

test_rsp_rx

Contains the number of 802.2 Logical Link Control (LLC) `Test` responses received by circuit **<cct_name>**. Receipt of a `Test` response requires the previous transmission of a `Test` command. Because the system does not issue `Test` commands, the value contained in this variable should be 0, as should the value of ***test_cmd_tx.***

test_rsp_tx

Contains the number of 802.2 Logical Link Control (LLC) `Test` responses issued on circuit **<cct_name>**. `Test` responses reply to `Test` commands and verify the LLC-to-LLC transmission path. The value contained in this variable should equal the value contained in ***test_cmd_rx.***

to_long_error_rx

Contains the number of frames, received by circuit **<cct_name>**, that exceeded 1518 bytes in length (this byte count does not include the 64-bit frame preamble and synchronization bits).

total_rx_error

Contains the total number of receive errors on circuit **<cct_name>**. This value equals the sum of ***alig_error_rx, fcs_error_rx, frams_incomp_rx, lack_resc_error_rx, oflo_rx,*** and ***to_long_error_rx.***

total_tx_error

Contains the total number of transmit errors on circuit **<cct_name>**. This value equals the sum of ***babl_error_tx,***

bufferr_tx, excessv_colln_tx, late_colln_tx, lcar_tx, and ***uflo_tx.***

uflo_tx

Contains the total number of underflows on circuit **<cct_name>**. An underflow occurs when the transmitter portion of the LANCE truncates a frame because of the late receipt of data from memory.

unrecog_pdu

Contains the number of unrecognized 802.2 Logical Link Control protocol data units (PDUs) received on circuit **<cct_name>**. An unrecognized PDU is one whose control field indicates other than a UI (Unnumbered Information) frame, XID (Exchange Identification) frame, or Test frame.

xid_cmd_rx

Contains the number of 802.2 Logical Link Control (LLC) `XID` (Exchange Identification) commands received by circuit **<cct_name>**. `XID` commands request LLC service-type and receive window capacity data from a remote host. Because the system does respond to such commands, the value contained in this variable should equal the value contained in ***xid_rsp_tx***.

xid_cmd_tx

Contains the number of 802.2 Logical Link Control (LLC) `XID` commands issued on circuit **<cct_name>**. Because the system does not provide a user interface to this facility, the value contained in this variable should be 0, as should the value of ***xid_rsp_rx.***

xid_rsp_rx

Contains the number of 802.2 Logical Link Control (LLC) `XID` responses received by circuit **<cct_name>**. Receipt of a `XID` response requires the previous transmission of a `XID` command. Because the system does not issue `XID` commands, the value contained in this variable should be 0, as should the value of ***xid_cmd_tx***.

xid_rsp_tx

Contains the number of 802.2 Logical Link Control (LLC) `XID` responses issued on circuit **<cct_name>**. `XID` responses reply to `XID` commands and provide LLC service-type and receive-window-capacity data. The value contained in this variable should equal the value contained in ***xid_cmd_rx.***

xmt_desc_cnt

Contains the current number of frames awaiting transmission by the LANCE.

12.8.3 Point-to-Point Information Base

The PPP sub-system contains information on the PPP MIB Link Quality Table and PPP circuit event messages. The PPP MIB variables for the Link Quality Table are organized under ***exmib.ppp.link_quality_table.entry***.

You use the NCL `LIST` command to display all, or a portion of , the ***ppp*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***ppp*** variable in the following format:

exmib.ppp.link_quality_table.entry.variable_name.cct

where:

exmib.ppp

Is the ***ppp*** managed object.

link_quality_table.entry

Is the Link Quality Table entry.

variable_name.

Is the variable name.

cct

Is the circuit number of the ***ppp*** circuit.

The MIB numbers for the above hierarchy are: ***26.18.4.1***. For example, the PPP MIB Link Quality Table variable ***in_tx_lqrs*** would be accessed via the following MIB hierarchy.

26.18.4.1.2.8

where:

cct

Is the circuit number of the PPP circuit. In the above example

8

Is the PPP circuit number

2

Refers to the MIB variable ***in_tx_lqrs***.

The system contains ***ppp*** information in a 5-level tree. The top-most level identifies the ***ppp*** managed object. The next level consists of a configuration -dependent number of branches, with each branch identifying a slot number. The next level contains the ***ppp*** Link Quality Table, while the next level contains the ***ppp*** Link Quality Entries. The bottom level contains the ***ppp*** variables.

Figure 12-9 illustrates the structure of the ***ppp*** information base.

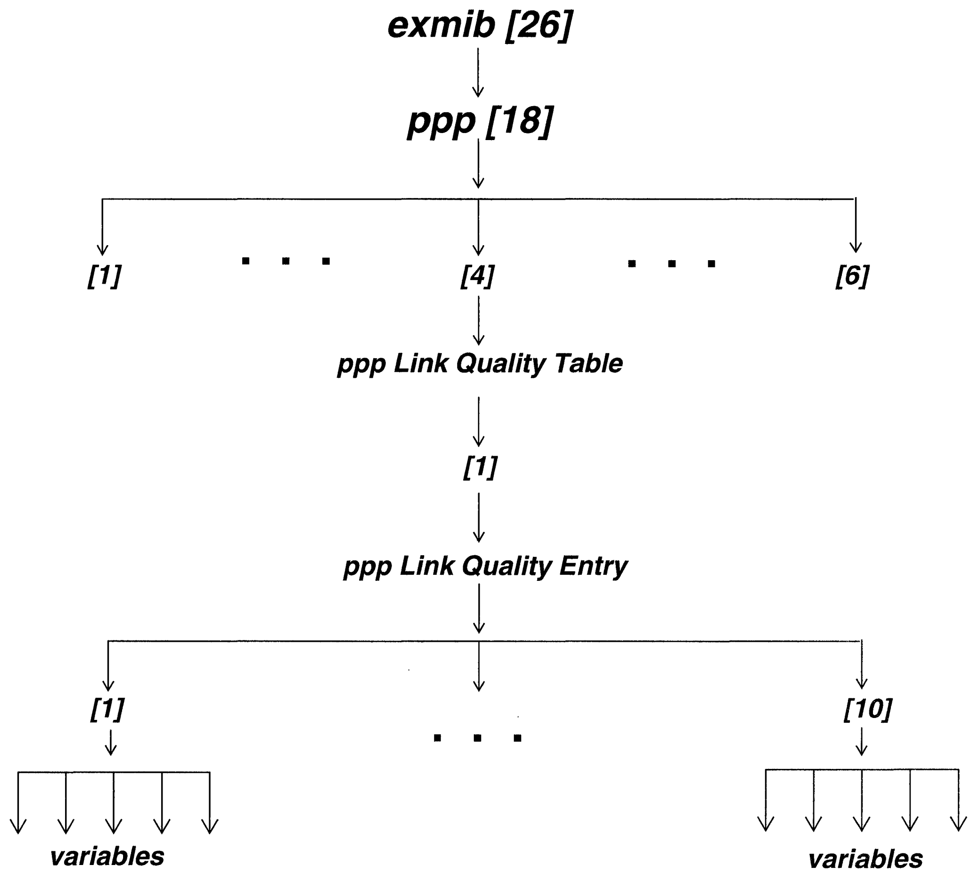


Figure 12-9 ***ppp*** **Information Base**

12.8.3.1 PPP Link Quality Table Variables

The PPP MIB variables for the Link Quality Table are as follows:

index

Is a unique value for each PPP link interface. The index value is the circuit number of the PPP circuit.

in_rx_bytes

Is a 32-bit state variable indicating the number of bytes which were received on the inbound link during the last period.

in_rx_pkts

Is a 32-bit state variable indicating the number of packets which were received on the inbound link during the last period.

in_tx_bytes

Is a 32-bit state variable indicating the number of bytes which were transmitted on the inbound link during the last period. In other words, ***in_tx_bytes*** indicates the number of bytes the remote peer station transmitted during the last period.

in_tx_lqrs

Is an 8-bit state variable indicating the number of Link Quality Report (LQR) packets that the remote peer remote station had to transmit so that the local end could receive exactly one LQR. The ***in_tx_lqrs*** variable defines the length of the period over which ***in_tx_packets, in_tx_bytes, in_rx_packets,*** and ***in_rx_bytes*** were measured.

in_tx_pkts

Is a 32-bit state variable indicating the number of packets which were transmitted on the inbound link during the last period. In other words, ***in_tx_pkts*** indicates the number of packets the remote peer station transmitted during the last period.

last_out_tx_bytes

Is a 32-bit state variable storing the value of the ***Out-Tx-Octets-Ctr*** from the last received Link Quality Report packet. In other words, ***last_out_tx_bytes*** indicates the total number of bytes transmitted by the remote peer station since the LCP reached the **OPEN** state.

last_out_tx_pkts

Is a 32-bit state variable storing the value of the ***Out-Tx-Packets-Ctr*** from the last received Link Quality Report packet. In other words, ***last_out_pkts*** indicates the total number of packets transmitted by the remote peer station since the LCP reached the **OPEN** state.

last_in_rx_pkts

Is a 32-bit state variable storing the value of the ***In-Rx-Packets-Ctr*** from the last received Link Quality Report packet. In other words, ***last_in_rx_pkts*** indicates the total number of packets transmitted by the peer (remote) station since the LCP reached the OPEN state.

pppLinkQualityEntry { pppLinkQualityTable 1 }

Provides Link Quality Management information about a particular PPP Link.

12.8.3.2 PPP Circuit Variables

The ***point-to-point*** circuit variables, in alphabetical order, are as follows:

bad_frames_rx

Contains the aggregate number of erroneous frames received by circuit **<cct_name>**.

dls_ret_rx

Contains the number of frames, received by circuit **<cct_name>**, that were later returned by the data- link service (DLS) software. Within the system software architecture, DLS resides between the driver software and the application software. It performs such services as multiplexing/ demultiplexing and encapsulation/ deencapsulation. DLS can return frames for numerous reasons (many of which are application-specific): for example, because of unknown internal service-access points (ISAPs); because of user-specified filtering requirements contained within the `config` file; or because of lack of enabled entities (for example, IP, DECnet, etc.).

dls_ring_cnt

Contains the current number of packets received by circuit **<cct_name>** and transferred to the DLS receiver queue.

frames_rx_ok

Contains the number of frames received without error by circuit **<cct_name>**.

frames_tx_ok

Contains the number of frames transmitted without error by circuit **<cct_name>**.

frams_incomp_rx

Contains the number of incomplete frames received by circuit **<cct_name>**. An incomplete frame is identified by failure to set the end-of-long-frame bit in the receive message descriptor.

frmr_frames_rx

Contains the count of unnumbered FRMR (Frame Reject) frames received by circuit **<cct_name>**. A FRMR frame reports an error condition. The remote end of the point-to-point circuit transmits a FRMR frame in response to a previous command requesting an unavailable action or service. Because the service or command is unavailable, a FRMR frame does not request retransmission of the erroneous frame.

lack_resc_error_rx

Contains the number of instances that circuit **<cct_name>** lost a frame because it could not obtain a receive buffer.

merr

Contains the number of memory errors. A memory error indicates that the MK5025 Link Level Controller, after becoming bus master, failed to receive a ready signal within 256 increments of the system clock after asserting a memory address on the data/address bus. A memory error disables the Link Level Controller.

octets_rx_ok

Contains the number of octets (bytes) received without error by circuit **<cct_name>**.

octets_tx_ok

Contains the number of octets (bytes) transmitted without error by circuit **<cct_name>**.

oflo_rx

Contains the total number of overflows on circuit **<cct_name>**. An overflow occurs when the Receiver FIFO buffer is full when the MK5025 Link Level Controller was ready to input data.

rcv_desc_cnt

Contains the current number of receiver data buffers available to the MK5025 Link Level Controller.

rejects_rx

Contains the count of supervisory REJ (Reject) frames received by circuit **<cct_name>**. A REJ frame is a negative acknowledgment and requests the retransmission of specified I (Information) frames.

rejects_tx

Contains the number of supervisory REJ (Reject) frames transmitted by circuit **<cct_name>**. A REJ frame is a negative acknowledgment and requests the retransmission of specified I (Information) frames.

runts_rx

Contains the aggregate number of frames of insufficient length received by circuit **<cct_name>**.

t1_tos

Contains the number of T1 timeouts. The T1 timer measures the interval between command transmission and the receipt of a response. If a response is not received within this interval, the MK5025 Link Level Controller increments this counter and then retransmits the command with the P bit set to require an immediate response.

test_cmd_rx

Contains the number of 802.2 Logical Link Control (LLC) `Test` commands received by circuit **<cct_name>**. `Test` commands seek to provide a basic verification of the LLC to LLC transmission path. Because the system does respond to such commands, the value contained in this variable should equal the value contained in ***test_rsp_tx.***

test_cmd_tx

Contains the number of 802.2 Logical Link Control (LLC) `Test` commands issued on circuit **<cct_name>**. Because the system does not provide a user interface to this facility, the value contained in this variable should be 0, as should the value of ***test_rsp_rx.***

test_rsp_rx

Contains the number of 802.2 Logical Link Control (LLC) `Test` responses received by circuit **<cct_name>**. Receipt of a `Test` response requires the previous transmission of a `Test` command. Because the system does not issue `Test` commands, the value contained in this variable should be 0, as should the value of ***test_cmd_tx.***

test_rsp_tx

Contains the number of 802.2 Logical Link Control (LLC) `Test` responses issued on circuit **<cct_name>**. `Test` responses reply to `Test` commands and verify the LLC-to-LLC transmission path. The value contained in this variable and the value contained in ***test_cmd_rx*** should be equal.

total_rx_error

Contains the total number of receive errors on circuit **<cct_name>**. This value equals the sum of ***bad_frames_rx, frams_incomp_rx, frmr_frames_rx, lack_resc_error_rx, oflo_rx, rejects_rx***, and ***runts_tx.***

uflo_tx

Contains the total number of underflows on circuit **<cct_name>**. An underflow occurs when the transmitter portion of the MK5025

Link Level Controller truncates a frame because of late receipt of data from memory.

unrecog_pdu

Contains the number of unrecognized 802.2 Logical Link Control protocol data units (PDUs) received by circuit **<cct_name>**. An unrecognized PDU is one whose control field indicates other than a UI (Unnumbered Information) frame, an XID (Exchange Identification) frame, or a Test frame.

xid_cmd_rx

Contains the number of 802.2 Logical Link Control (LLC) `XID` (Exchange Identification) commands received by circuit **<cct_name>**. `XID` commands request LLC service-type and receive window capacity data from a remote host. Because the system does respond to such commands, the value contained in this variable should equal the value contained in ***xid_rsp_tx***.

xid_cmd_tx

Contains the number of 802.2 Logical Link Control (LLC) `XID` commands issued on circuit **<cct_name>**. Because the system does not provide a user interface to this facility, the value contained in this variable should be 0, as should the value of ***xid_rsp_rx***.

xid_rsp_rx

Contains the number of 802.2 Logical Link Control (LLC) `XID` responses received by circuit **<cct_name>**. Receipt of a `XID` response requires the previous transmission of a `XID` command. Because the system does not issue `XID` commands, the value contained in this variable should be 0, as should the value of ***xid_cmd_tx***.

xid_rsp_tx

Contains the number of 802.2 Logical Link Control (LLC) XID responses issued on circuit **<cct_name>**. XID responses reply to `XID` commands and provide LLC service type and receive window capacity data. The value contained in this variable should equal the value contained in ***xid_cmd_rx.***

xmt_desc_cnt

Contains the current number of frames awaiting transmission by the MK5025 Link Level Controller.

12.8.4 FDDI Circuit Variables

The ***FDDI*** information base contains variables that describe transmission and reception activities across each FDDI circuit.

The Wellfleet system maintains ***FDDI*** circuit information as a 4-level tree. The top-most level (***cct***) identifies the circuit managed object. The middle level consists of a configuration-dependent number of branches, with each branch identifying a specific circuit (***cct_name***). The lowest level contains the actual variables.

You use the NCL `LIST` command to display all, or a portion of the FDDI information base, and the NCL `GET` command to obtain the value of any variable within the information base

You express the pathname to any ***cct*** variable in the following format:

cct.cct_name.smt_group.variable_name

where:

cct

Is the FDDI circuit managed object.

cct_name

Is the FDDI circuit name.

smt_group

Specifies a station management statistic.

variable_name

Is the variable name.

The ***FDDI*** circuit variables, in alphabetical order, are as follows:

frames_tx_dropped

Contains the number of frames not transmitted due to abnormal ring status or lack of resources.

ieee_addr

Contains the IEEE MAC address of this circuit.

ring_status

Identifies the current state of the ring: UP or DOWN.

ring_up_ct

Contains the number of ring status **UP** events.

smt.cfm_state

Identifies the current state of the Configuration Management (CFM) state machine: **THRU_A, THRU_B, WRAP_A, WRAP_B, ISOLATED, WRAP_S, THRU_AB, WRAP_AB,** or **UNKNOWN.**

smt.downstream_mac

Contains the SMT MAC address of this circuit's downstream neighbor.

smt.ecm_state

Identifies the current state of the Entity Coordination Management (ECM) state machine: **IN, OUT, INSERT, TRACE, LEAVE, PATH_TEST, CHECK, DEINSERT,** or **UNKNOWN**.

smt.pcm_state_phy_a

Identifies the current state of the Physical Connection Management (PCM) state machine for phy A: **ACTIVE, CONNECT, OFF, BREAK, TRACE, NEXT, SIGNAL, JOIN, VERIFY, MAINT,** or **UNKNOWN**.

smt.pcm_state_phy_b

Identifies the current state of the Physical Connection Management (PCM) state machine for phy B: **ACTIVE, CONNECT, OFF, BREAK, TRACE, NEXT, SIGNAL, JOIN, VERIFY, MAINT,** or **UNKNOWN.**

smt.rmt_state

Identifies the current state of the Ring Management (RMT) state machine: **RING_OP, ISOLATED, NON_OP, DETECT, NON_OP_DUP, RING_OP_DUP, TRACE, DIRECTED,** or **UNKNOWN.**

smt.source_mac

Contains the SMT MAC address of this circuit.

smt.upstream_mac

Contains the SMT MAC address of this circuit's upstream neighbor.

12.8.5 Token Ring Circuit Variables

The ***Token Ring*** information base contains variables that describe transmission and reception activities across each Token Ring circuit.

12.8.5.1 Adapter Check Variables

The ***adapter check*** variables monitor unrecoverable hardware or software errors specific to the token ring adapter (the chip set that provides the interface between the Token Ring medium and the Wellfleet system). The ***adapter check*** variables, in alphabetical order, are as follows:

adpt_bad_dio_par

Contains the number of times the token ring adapter detected a bad parity value on data passed from the Wellfleet system to the adapter through a direct I/O access.

adpt_dma_rd_abort

Contains the number of times the token ring adapter aborted a DMA (direct memory access) read operation from the Wellfleet system. Such an abort is caused by:

1. Excessive parity errors
2. Excessive bus errors
3. The expiration of a 10-second timer while the token ring adapter waits for the completion of a DMA bus operation.

adpt_dma_wr_abort

Contains the number of times the token ring adapter aborted a DMA (direct memory access) write operation to the Wellfleet system. Such an abort is caused by:

1. Excessive parity errors
2. Excessive bus errors
3. The expiration of a 10-second timer while the token ring adapter waits for the completion of a DMA bus operation.

adpt_parity_err

Contains the number of times the token ring adapter detected a bus parity error on the adapter's internal bus.

adpt_ring_underrun

Contains the number of times the token ring adapter detected an internal DMA underrun when receiving from the ring.

total_adapt_error

Contains the aggregate count of adapter errors. This count is the sum of: ***adpt_bad_dio_par, adpt_dma_rd_abort, adpt_dma_wr_abort, adpt_parity_err,*** and ***adpt_ring_underrun.***

12.8.5.2 Adapter Error Log Variables

The ***adapter error log*** variables read various error counters maintained by the token ring adapter. The ***error log*** variables, in alphabetical order, are as follows:

log_ari_fci_err

Contains the number of times the token ring adapter detected that its upstream neighbor was unable to set the Address Recognized Indicator (ARI) or Frame Copied Indicator (FCI) bits of received frames.

log_burst_err

Contains the number of times the token ring adapter detected the absence of signal transitions for five half-bit times between the

Starting and Ending Delimiters, or between the Ending and Starting Delimiters.

log_dma_bus_err

Contains the number of DMA bus errors that do not exceed the abort threshold.

log_dma_par_err

Contains the number of DMA parity errors that do not exceed the abort threshold.

log_frm_cpy_err

Contains the number of times the token ring adapter (while in the receive/repeat mode) recognized a frame addressed to its specific address but found the ARI bits not equal to zero. Such a condition indicates a possible line hit or duplicate address.

log_line_err

Contains the number of times the token ring adapter (while in the receive/repeat mode) recognized a line error. A line error is recorded when one of the following conditions exist:

1. A code violation exists between the Starting and Ending delimiters of a frame.
2. A code violation exists in a token.
3. A Frame Check Sequence (FCS) error exists.

When the token ring adapter increments ***log_line_err***, it also sets the Error Detected Indicator (EDI) of the faulty frame to ensure that no further adapters will count the error.

log_lost_frm

Contains the number of times the token ring adapter (while in the transmit/stripping mode) failed to receive the end of a frame that it had previously transmitted.

log_rx_congest

Contains the number of times the token ring adapter (while in the repeat mode) recognized a frame addressed to its specific address, but was unable to copy the frame because no buffer space was available.

log_token_err

This variable is active only in the Active Monitor station. It contains the number of times the Active Monitor token ring adapter detected an error with the token protocol. Such token errors are as follows:

1. A token with a non-zero priority has the Monitor Count bit equal to one, indicating a circulating high-priority token.

2. A frame has the Monitor Count bit equal to one, indicating a circulating frame.

3. No token or frame is received within a 10-millisecond window.

4. The Starting Delimiter/Token sequence has a code violation in an area where code violations must not occur.

total_log_error

Contains the aggregate count of adapter log errors. This count is the sum of: ***log_ari_fci_err, log_burst_err, log_dma_bus_err, log_dma_par_err, log_frm_cpy_err, log_line_err, log_lost_frm, log_rx_congest,*** and ***log_token_err.***

12.8.5.3 Miscellaneous Error Counts

Miscellaneous error count variables, in alphabetical order, are as follows:

total_rx_error

Contains the sum of total ***log_rx_congest*** and ***rx_suspended***.

total_tx_error

Contains the sum of: ***tx_congestion, tx_frm_err, tx_frm_size_err, tx_ill_frm_fmt, tx_list_err, tx_odd_addr,*** and ***tx_threshold.***

12.8.5.4 Receive Status Variables

The ***receive status*** variables contain cumulative counts of certain fields contained in the RECEIVE parameter list. The ***receive status*** variables, in alphabetical order, are as follows:

rx_frm_cmpl (rint)

Contains the number of times the token ring adapter received a frame and then generated an adapter to attached system interrupt. Such an interrupt is enabled or disabled by the Frame Interrupt bit (bit 4 of **RECEIVE_CSTAT**). Because frames can be received faster than the adapter can cause the interrupts, each increment of this counter can report the reception of one or multiple frames.

rx_suspended

Contains the number of times the token ring adapter processed a RECEIVE parameter list that contained both an odd address in the Forward Pointer field (indicating the end of the list) and a Frame Complete bit equal to 0 (indicating more to follow).

12.8.5.5 Ring Status Variables

The ***ring status*** variables monitor significant events on the Wellfleet system's Token Ring interface. The ***ring status*** variables, in alphabetical order, are as follows:

ring_auto_rem

Contains the number of times that the token ring adapter detected an internal hardware error (either within the chip set itself, or within the lobe, which is the physical star wiring between the adapter and the wire concentrator). After the detection of such an error, the token ring adapter removes itself from the ring.

ring_cnt_overflow

Contains the number of times a token ring adapter error counter incremented from 254 to 255.

ring_hard_err

Contains the number of times the token ring adapter transmitted or received beacon frames to or from the ring. The beacon process is used to recover the ring after a ring station has sensed a hard error (which renders the ring inoperable). Hard errors are caused by: (1) wire faults, (2) frequency errors, or (3) incoming signal loss. The station that detects such a hard error transmits, or beacons, information that isolates the failure location.

ring_lobe_wire

Contains the number of times that the token ring adapter detected a short or open circuit between the adapter and the wire concentrator.

ring_one_station

Contains the number of times that the token ring adapter sensed that it was the only station on the ring.

ring_rem_station

Contains the number of times that the token ring adapter removed itself from the ring after receiving a Remove Ring Station Frame. A Remove Ring Station Frame is issued by the network manager and forces an adapter to remove itself from the ring.

ring_recover

Contains the number of times that the token ring adapter observed Claim Token Frames on the ring. A ring station transmits a Claim Token Frame when it detects that the ring does not contain an active monitor, or that the active monitor is not functioning properly.

ring_sig_loss

Contains the number of times the token ring adapter detected a loss of signal on the ring.

ring_soft_err

Contains the number Report Error Frames transmitted by the token ring adapter. A Report Error Frame is transmitted in response to a soft error. A soft error is one that temporarily degrades system performance but may be recovered using standard adapter protocols.

ring_tx_beacon

Contains the number of times the token ring adapter transmitted beacon frames to the ring. The transmission of a beacon frame indicates that the adapter has detected a hard error. This count specifies the number of hard errors detected by the adapter; the difference between this count and ***hard_err*** specifies the number of hard errors detected by other ring stations.

total_ring_error

Contains the aggregate count of ring errors. This count is the sum of: ***ring_auto_rem, ring_cnt_overflow, ring_hard_err, ring_lobe_wire, ring_one_station, ring_rem_station, ring_ring_recover, ring_sig_loss, ring_soft_err,*** and ***ring_tx_beacon.***

12.8.5.6 Transmit Status Variables

The ***transmit status*** variables contain cumulative counts of the fields contained in the TRANSMIT parameter list, as well as cumulative counts of manipulations of bit fields contained in the **TRANSMIT_CSTAT** control word. The ***transmit status*** variables, in alphabetical order, are as follows:

tx_frm_cmpl (tint)

Contains the number of times the token ring adapter transmitted a frame and then generated an adapter to attached system interrupt. Such an interrupt is enabled or disabled by the Frame Interrupt bit (bit 4 of **TRANSMIT_CSTAT)**. Because frames can be transmitted faster than the adapter can cause the interrupts, each increment of this counter can report the transmission of one or multiple frames.

tx_frm_err

Contains the number of times the token ring adapter recorded a transmit frame error. A transmit frame error occurs when the token ring adapter finds an erroneous Start Frame bit (bit 2 of **TRANSMIT_CSTAT**); either the bit is equal to 1 on a list that is not the anticipated start of a frame, or the bit is equal to 0 on an anticipated start of frame.

tx_frm_size_err

Contains the number of times the token ring adapter recorded a transmit frame size error. A transmit frame size error occurs when the Frame Size count (bytes 7 and 8 of the TRANSMIT parameter list) is not equal to the sum of the data count fields contained in the parameter list, or when the Frame Size count is less than the required header plus one byte of Information field date, or when the Frame Size count is equal to 0 (except in lists that define I frames).

tx_ill_frm_fmt

Contains the number of times the token ring adapter recorded an illegal frame format error. An illegal frame format error occurs when bit 0 of the Frame Control (FC) field is equal to 1.

tx_list_err

Contains the number of times the token ring adapter recorded an error in one of the lists that compose the frame. If such an error occurs, the token ring adapter terminates the **`TRANSMIT`** command; the attached system must issue another **`TRANSMIT`** command to continue.

tx_odd_adr

Contains the number of times the token ring adapter processed a TRANSMIT parameter list that contained both an odd address in the Forward Pointer field (indicating the end of the list) and an End Frame Bit equal to 0 (indicating more to follow).

tx_threshold

Contains the number of times that the token ring adapter recorded a transmit threshold error. A transmit threshold error occurs when the Frame Size count (bytes 7 and 8 of the TRANSMIT parameter list) exceeds the buffer capacity allocated at system initialization.

12.9 Configuration Management Information Base

The Wellfleet system maintains ***config*** information about the hardware and software configuration of the system.

You use the NCL `LIST` command to display all, or a portion, of the ***config*** information base, and the NCL **`GET`** command to obtain the value of any variable within the information base.

The ***config*** variables are as follows:

cfg_fname

Contains the name of the text file from which the current configuration was read.

config.key.<sw_option>

Where <***sw_option***> is any of the following routing/bridging protocols:

lb	-- Bridge
dr	-- Decnet Routing
ip	-- Ip Routing
x25	-- X.25
xns	-- XNS/IPX Routing
at	-- AppleTalk Routing

These variables contain a 1 if option x is present, or a 0 if the option is absent.

config.bootp_file_svr

Contains the IP address of the server that downloaded the image and the configuration files to the system.

config.bootp_rq_svr

Contains the IP address of the **BOOTP** system that responded to the **BOOTP** broadcast request.

load_image

Contains the name of the binary load image file from which the system booted.

ver_major

Contains the major version number of the resident software.

ver_minor

Contains the minor version number of the resident software.

config.hwtable.entry.<entry>

Contains the hardware (hw) table that lists the resident hardware.

Individual hardware table entries are as follows:

ace_serial_num[n]

Contains the serial number of the processor board in slot n.

ace_rev[n]

Contains the revision level of the processor board in slot n.

mod_serial_num[n]

Contains the serial number of the link module in slot n.

mod_rev[n]

Contains the revision level of the link module in slot n.

mod_id[n]

Contains the ID number of the link module in slot n. ID numbers are documented in Table 9-1.

slot_no[n]

Contains the slot in which the boards described in this entry are present.

12.10 DECnet Phase IV Circuit Group Information Base

The ***drs*** (DECnet routing service) information base contains variables that describe transmission and reception activities across each DECnet circuit group; it also contains variables that describe the rejection of certain packets by the DECnet router.

The system maintains ***drs*** circuit group specific information as a 4-level tree. The top-most level identifies the ***drs*** managed object. The next lower level identifies the managed object instance (***cg***). The next level contains a configuration-dependent number of branches, with each branch identifying a specific DECnet circuit group ***(cg_name***). The lowest level contains the actual ***drs*** variables.

The system maintains ***drs aggregate rejection*** information as a 3-level tree. The top-most level identifies the ***drs*** managed object. The middle level identifies the managed object instance (***total***). The lowest level contains the variables.

Figure 12-10 illustrates the structure of the ***drs*** information base.

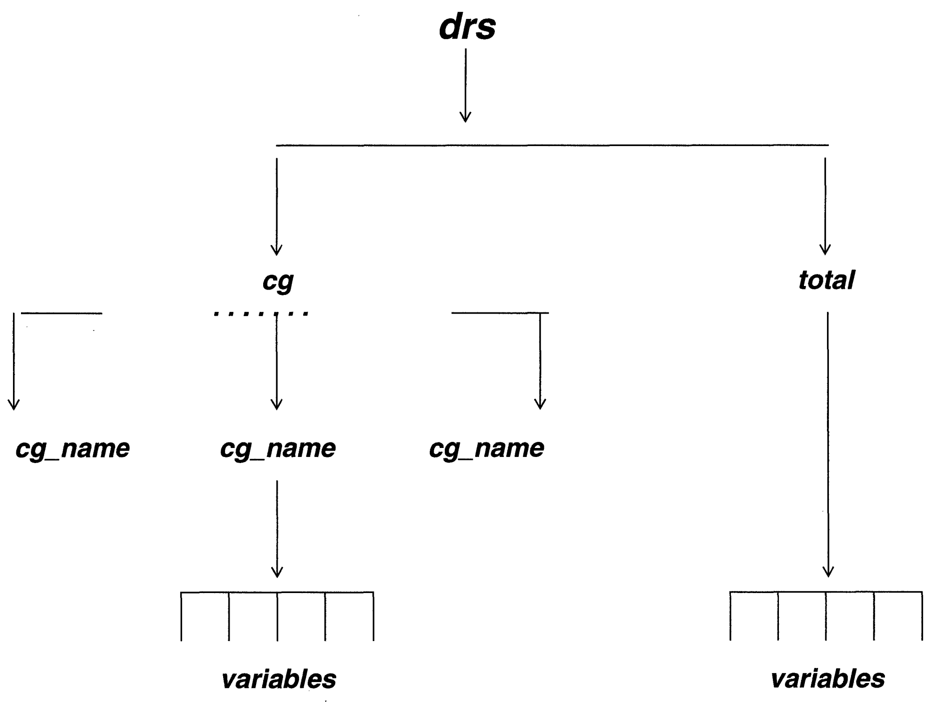


Figure 12-10 *drs* Information Base

You use the NCL `LIST` command to display all, or a portion of, the ***drs*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***drs*** circuit group variable in the following format:

drs.cg.cg_name.variable_name

where:

drs

Is the ***drs*** managed object.

cg

Is the ***cg*** (circuit-group specific) managed object instance.

cg_name

Is the circuit group name.

variable_name

Is the variable name.

You express the pathname to any other ***drs*** variable in the following format:

drs.total.variable_name

where:

drs

Is the ***drs*** managed object.

total

Is the ***total*** (aggregate rejection) managed object instance.

variable_name

Is the variable name.

12.10.1 Decnet Circuit-Group-Specific Variables

The ***drs*** circuit group specific variables, in alphabetical order, are as follows:

drop

Contains the number of packets dropped by circuit group **<cg_name>**.

trans_pkts_recv

Contains the number of data packets received by circuit group **<cg_name>**.

trans_pkts_sent

Contains the number of data packets sent by circuit group **<cg_name>**.

12.10.2 DECnet Aggregate Variables

The ***drs*** aggregate rejection variables, in alphabetical order, are as follows:

aged_pkt_loss

Contains the number of packets dropped by the DECnet router because the packet had transited too many routers prior to reaching its destination. The maximum number of routers that a packet can transit is determined by the Area Max. Hops and Max Hops parameters. Area Max. Hops specifies the number of routers that a packet can transit before reaching its destination area; Max Hops specifies the number of routers that a packet can transit before reaching its destination system.

node_unreach

Contains the number of packets dropped by the DECnet router because the destination system (while within range) was unreachable.

node_out_of_range

Contains the number of packets dropped by the DECnet router because the destination system resided in an area having a number greater than that designated by the Max. Area parameter, or because the destination system number exceeded that designated by the Max. Nodes parameter.

oversized_pkt_loss

Contains the number of packets dropped by the DECnet router because the packet size exceeded the capacity of the transmission media.

pkt_format_error

Contains the number of packets dropped by the DECnet router because some portion of the packet header could not be parsed.

route_update_loss

Contains the number of topology packets dropped by the DECnet router because the packet contained information beyond the router's capacity.

12.11 DECnet Configuration Information Base

The ***decnet configuration*** information base contains variables that describe DECnet global and interface-specific configuration parameters.

The system maintains ***decnet*** global and interface-specific information as a 2-level tree. The upper level identifies the ***decnet*** managed object. The lower level contains the actual ***decnet*** variables.

Figure 12-11 illustrates the structure of the ***decnet*** information base.

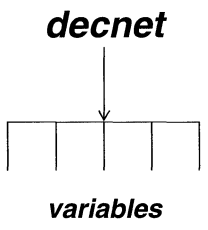


Figure 12-11 *decnet* Information Base

You use the NCL `LIST` command to display all, or a portion of, the *decnet* information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***decnet*** global variable in the following format:

decnet.variable_name

where:

decnet

Is the ***decnet*** managed object.

variable_name

Is the variable name.

You express the pathname to any ***decnet*** interface specific variable in the following format:

decnet.iftab.if.variable_name[#]

where:

decnet

Is the ***decnet*** managed object.

iftab.if

Is the interface specific managed object instance.

variable_name

Is the variable name.

[#]

Is the system-assigned interface number

12.11.1 DECnet Global Variables

The ***decnet global*** variables, in alphabetical order, are as follows:

amaxcst

Contains the maximum inter-area transit cost.

amaxhop

Contains the maximum number of areas that a packet can traverse from source to destination.

area

Contains the number the local DECnet area.

bcrtimer

Contains the time interval between topology packets.

enadj

Contains the maximum number of adjacent systems.

maxarea

Contains the number of areas in the network.

maxnode

Contains the maximum number of systems per network area.

maxvisit

Contains the maximum number of times a packet can pass through the same router.

nmaxcst

Contains the maximum system-to-system transit cost.

nmaxhop

Contains the maximum number of hops that a packet can transit from source to destination.

node

Contains the router's DECnet node (system) number.

state

Contains the router state: ***1*** indicates that the router is enabled and forwarding packets; ***2*** indicates that the router is disabled.

12.11.2 DECnet Interface-Specific Variables

The ***decnet interface-specific*** variables, in alphabetical order, are as follows:

ccst

Contains the relative circuit cost assigned to the interface.

cname

Contains the name of the circuit group that enables the decnet interface.

htime

Contains the time interval between decnet hello packets.

index

Contains the system-assigned interface number.

numrtr

Contains the number of routers associated with the interface.

rprior

Contains the router priority.

state

Contains the interface state: ***1*** indicates that the router is enabled and forwarding packets; ***2*** indicates that the router is disabled.

12.12 DMAP Information Base

The ***dmap*** (Direct Memory Access Processor) information base contains variables that describe DMA transfers between ACE printed circuit boards.

One variable set documents the messaging processor's use of message buffers to transmit data over the VME bus. An identical variable set documents the processor's use of packet buffers to store data for eventual transmission across LAN or WAN facilities. A third variable set documents the transfer of message and packet buffers from the messaging processor to the protocol processor.

The system maintains ***dmap buffer utilization*** information as a complex 6-level tree. The top-most level identifies the ***dmap*** managed object. The next level consists of a configuration-dependent number of branches, with each branch identifying a slot number (***slot_#***). Level 3 distinguishes between buffer types (***msg*** or ***pkt***). Level 4 contains actual ***dmap*** variables, and a single branch (***chn***) to lower levels. These lower levels provide information concerning data transfers across specific dma channels (VME slot pairs).

The system maintains ***dmap inter-processor transfer*** information as a 3-level tree. The top-most level identifies the ***dmap*** managed object. The middle level consists of a configuration-dependent number of branches, with each branch identifying a slot number (***slot_#***). The next level contains a single variable (***to_cpu***).

Figure 12-12 illustrates the structure of the ***dmap*** information base.

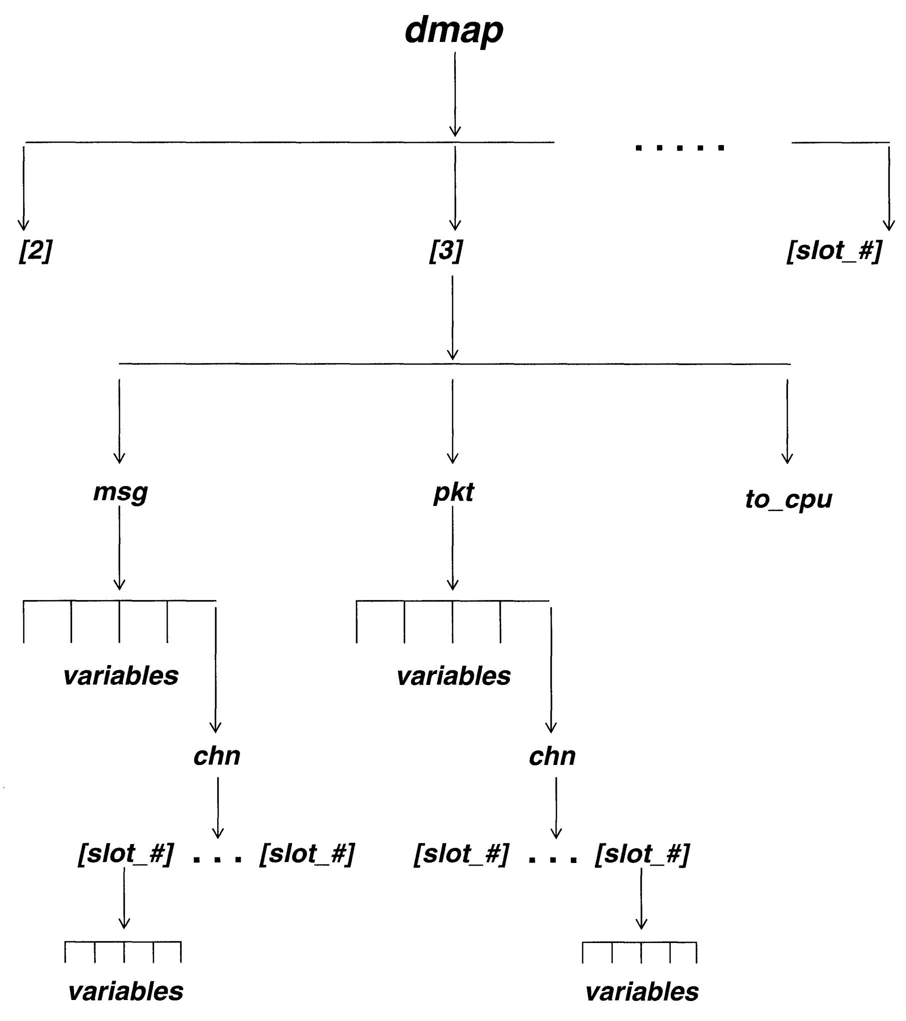


Figure 12-12 ***dmap*** **Information Base**

You use the NCL `LIST` command to display all, or a portion of, the ***dmap*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***dmap*** message or packet buffer variable in the following format:

dmap[slot_#].type.variable_name

where:

dmap

Is the ***dmap*** managed object.

[slot_#]

Is the slot number (allowable range is 2 to 14 depending on system type).

type

Is the buffer type (***msg*** for message buffers or ***pkt*** for packet buffers).

variable_name

Is the variable name.

You express the pathname to any ***dmap*** channel-specific variable in the following format:

dmap[slot_#].type.chn.[dest_slot_#]variable_name

where:

dmap

Is the ***dmap*** managed object.

[slot_#]

Is the slot number (allowable range is 2 to 14 depending on system type).

type

Is the buffer type (***msg*** for message buffers or ***pkt*** for packet buffers).

chn

Is the channel-specific data sub-root.

dest_slot_#

Is the destination slot.

variable_name

Is the variable name.

You express the pathname to the ***dmap*** inter-processor variable in the following format:

dmap[slot_#].to_cpu

where:

dmap

Is the ***dmap*** managed object.

[slot_#]

Is the slot number (allowable range is 2 to 14 depending on system type).

to_cpu

Is the inter-processor variable.

12.12.1 DMAP Buffer-Specific Variables

The ***dmap buffer-specific*** variables, in alphabetical order, are as follows:

dma

Contains the number of successful dma transfers from the messaging processor.

drop

Contains the number of message or packet buffers dropped (not transmitted) by the messaging processor because of lack of resources.

due

Contains the number of message or packet buffers owed to other dmaps (those dmaps resident on another slot). An owed buffer is one that has already been requested, but not, as yet, supplied.

from_cpu

Contains the number of dmap transfers from the protocol processor.

prealloc

Contains the total number of message or packet buffers on other slots pre-allocated for dmap transfers from **[slot_#].**

12.12.2 DMAP Channel-Specific Variables

The ***dmap channel-specific*** variables, in alphabetical order, are as follows:

dma

Contains the number of successful dma transfers from the **[slot_#]** to **[dest_slot_#]**.

drop

Contains the number of message or packet buffers, sourced at **[slot_#]** and destined for **[dest_slot_#]**, dropped by the messaging processor because of lack of resources.

que

Contains the current number of queue items awaiting transfer from **[slot_#]** to **[dest_slot_#]**.

12.12.3 DMAP Inter-Processor Variable

The ***dmap inter-processor*** variable ***dmap[slot_#].to_cpu*** contains the total number of message or packet buffers received by this ***dmap*** and passed to the protocol processor.

12.13 Driver Information Base

The ***driver*** information base contains multiple instances (one for each active slot) of a single control object, ***driver***, whose sole function is to generate system management messages.

The NCL `LIST` and `GET` commands provide no additional information regarding the ***driver*** information base.

12.14 Echo Service Information Base

The ***echo*** information base contains variables that describe the Transmission Control Protocol echo service.

The system maintains ***echo*** information as a 2-level tree, with the upper level identifying the ***echo*** managed object (***echo***) and the lower level containing the actual ***echo*** variables.

Figure 12-13 illustrates the structure of the ***echo*** information base.

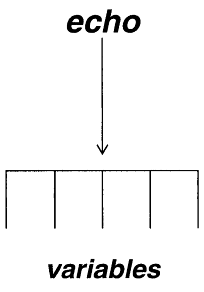


Figure 12-13 ***echo*** **Information Base**

You use the NCL `LIST` command to display all, or a portion of, the ***echo*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

The pathname to any ***echo*** variable takes the following format:

echo.variable_name

where:

echo

Is the ***echo*** managed object.

variable_name

Is the variable name.

The ***echo*** variables, in alphabetical order, are as follows:

---

mem_err

Contains the number of memory errors.

---

mem_use

Contains the total number of bytes used by the echo service.

---

no_mem

Contains the number of times echo was unable to function because of lack of memory resources.

rx_bytes

Contains the total number of bytes transmitted during echo sessions.

sess_cur

Contains the current number of echo sessions.

sess_tot

Contains the total number of echo sessions since the system last booted.

tx_bytes

Contains the total number of bytes received during echo sessions.

12.15 EGP Information Base

The ***egp*** (Exterior Gateway Protocol) information base contains variables that describe the transmission and reception of messages by the EGP software.

The system maintains ***egp*** information as a 2-level tree, with the upper level identifying the ***egp*** managed object and the lower level containing the actual ***egp*** variables.

Figure 12-14 illustrates the structure of the ***egp*** information base.

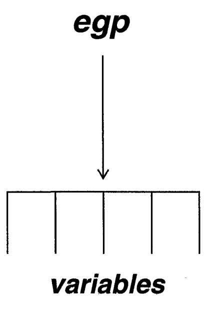


Figure 12-14 *egp* Information Base

You use the NCL `LIST` command to display the ***egp*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***egp*** variable in the following format:

egp.variable_name

where:

egp

Is the ***egp*** managed object.

variable_name

Is the variable name.

The ***egp*** variables, in alphabetical order, are as follows:

bad_asn

Contains the number of received EGP messages that contained an unrecognized autonomous system number in the EGP header.

bad_cod

Contains the number of received EGP messages that contained an unrecognized value in the Code field of the EGP header.

bad_hel

Contains the number of unexpected Hello neighbor reachability messages received by EGP. A Hello message is unexpected when received from a passive peer.

bad_ihu

Contains the number of unexpected I Hear You neighbor reachability messages received by EGP. An I Hear You message is unexpected when received in the absence of a prior Hello message.

bad_stt

Contains the total number of received EGP messages that contained an unrecognized value in the Status field of the EGP header.

bad_sum

Contains the total number of received EGP messages that contained a faulty checksum in the EGP header.

bad_type

Contains the total number of received EGP messages that contained an unrecognized value in the Type field of the EGP header.

bad_ver

Contains the total number of received EGP messages that contained an invalid EGP version number in the EGP header. The system's EGP implementation supports EGP Version 2.

cmdoos

Contains the number of times EGP received an out-of-sequence command message. An out-of-sequence message indicates that a prior message, issued by an EGP peer, has been missed.

cmdrej

Contains the number of times EGP refused to respond to a received command. Such refusal could be generated by receipt of a neighbor acquisition message from an unknown autonomous system, or by receipt of a faulty EGP message (for example, one with a bad checksum) from a known neighbor.

err

Contains the number of times EGP received a message that, while appropriate for the system's current state, was otherwise erroneous.

mem_use

Contains the number of memory bytes used by the EGP protocol. Memory requirements include protocol overhead, the state machine model, and timers.

mem_wait

Contains the number of instances in which EGP delayed operations because of unavailable memory resources.

no_mem

Contains the number of instances in which EGP was unable to function because of insufficient memory resources.

nop

Contains the number of instances that EGP received an otherwise valid EGP message inappropriate or inapplicable to its current state.

pkts_rcv

Contains the total number of received EGP messages.

pkts_snt

Contains the total number of transmitted EGP messages.

sess_cur

Contains the current number of EGP peers. An EGP peer is a remote routing device with which the system exchanges routing information.

sess_tot

Contains the aggregate number of EGP peers. This counter is incremented when the system moves from the EGP Idle state to either the Acquisition state or Down state.

tmrrej

Contains the number of t3 (abort timer) timeouts.

12.16 Hardware Information Base

The ***hw*** (hardware) information base contains variables that identify printed circuit boards (PCBs) by serial number, revision level, and (in the case of link modules) board type.

The system maintains ***hw*** information as a 3-level tree. The top-most level identifies the ***hw*** managed object. The middle level consists of a configuration-dependent number of branches, with each branch identifying a slot number (***slot_#***). The bottom level contains the actual ***hw*** variables.

Figure 12-15 illustrates the structure of the ***hw*** information base.

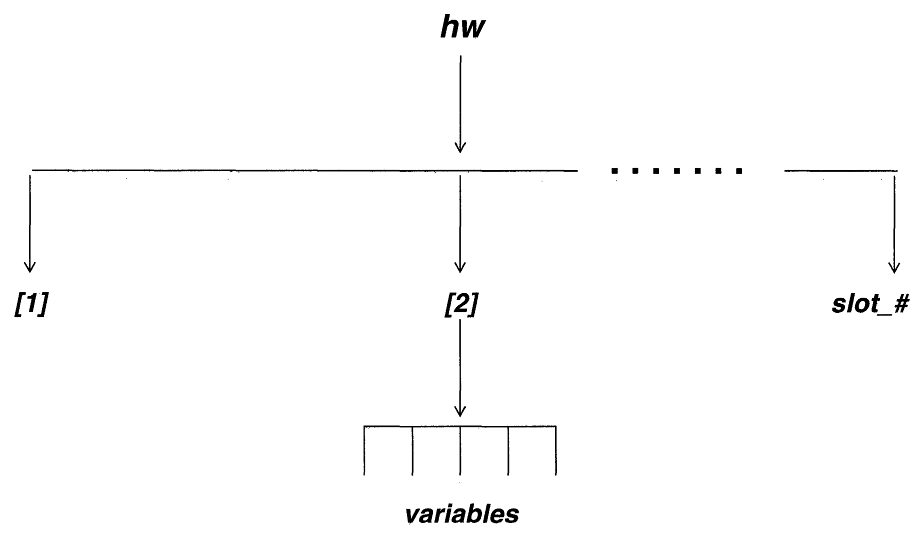


Figure 12-15 *hw* Information Base

You use the NCL `LIST` command to display all, or a portion of, the ***hw*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***hw*** variable in the following format:

hw[slot_#].variable_name

where:

hw

Is the ***hw*** managed object.

[slot_#]

Is the slot number (allowable range is 1 to 14 depending on system type).

variable_name

Is the variable name.

The ***hw*** variables, in alphabetical order, are as follows:

ace_rev

Contains the revision level of the ACE board in **[slot_#]**. If **[slot_#]** is 1, this variable contains the revision level of the system-controller board.

ace_serial_num

Contains the serial number of the ACE board in **[slot_#]**. If **[slot_#]** is 1, this variable contains the serial number of the system-controller board.

mod_id

Contains a decimal value that identifies the link module in **[slot_#].** The system software uses a hexadecimal representation of this value to identify the link module installed within each backplane slot. Table 9-1 lists the equations between ***mod_id*** values and specific link modules.

mod_rev

Contains the revision level of the link module in **[slot_#]**. If **[slot_#]** is 1, this variable contains the revision level of the system I/O board.

mod_serial_num

Contains the serial number of the link module in **[slot_#]**. If **[slot_#]** is 1, this variable contains the serial number of the system I/O board.

13 Management Information Base Variables, I to Z

This chapter provides descriptions of the objects (and variables), beginning with the letters *i* through *z*, contained in the management information base.

13.1 Internet MIB

The system is fully compliant with the standard ***Internet MIB***, as defined in Internet Request for Comments (RFC) 1156.

The NCL **RGETS** or **RGETMS** commands provide the most efficient means to obtain the value of any variable(s) within the standard ***Internet MIB***. **RGETS** returns a single variable value, while **RGETMS** returns the values of all the variables within an ***Internet MIB*** variable class.

You can also use the **RGETA**, **RGETI**, and **RGETR** commands to retrieve specific ***MIB*** tables. **RGETA** returns the ***MIB*** Address Resolution Table; **RGETI** returns the ***MIB*** IP Address Table; and **RGETR** returns the ***MIB*** IP Routing Table.

13.2 IP Information Base

The ***ip*** information base contains variables that describe transmission and reception activities across each IP interface; it also contains variables describing the IP routing table.

The system maintains ***ip*** interface information as a 4-level tree. The top-most level identifies the ***ip*** managed object. The next lower level identifies the interface instance of the managed object (***ip_interface***). The next level consists of a configuration-dependent number of branches, with each branch identifying a specific IP interface address (***ip_address***). The lowest level contains the actual ***ip*** variables.

The system maintains ***ip*** routing table information as a 4-level tree. The top-most level identifies the ***ip*** managed object. The next two lower levels identify the routing table instance of the managed object (***ip_route_table*** and ***net_tree***). The lowest level contains the variables.

Figure 13-1 illustrates the structure of the ***ip*** information base.

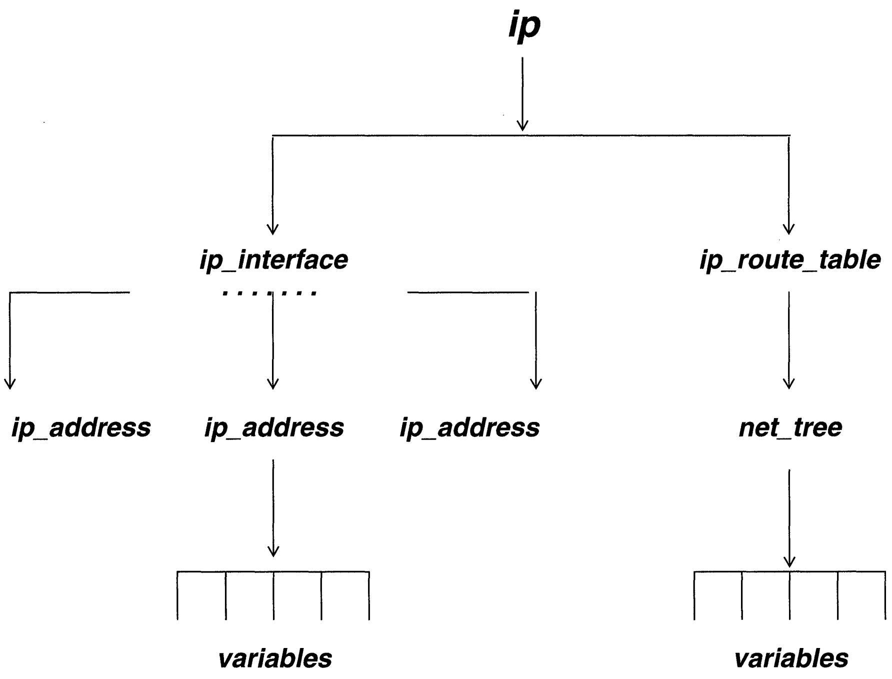


Figure 13-1 ***ip*** **Information Base**

You use the NCL **`LIST`** command to display all, or a portion of, the ***ip*** information base, and the NCL **`GET`** command to obtain the value of any variable within the information base.

You express the pathname to any *ip* interface variable in the following format:

ip.ip_interface.ip_address.variable_name

where:

ip

Is the ***ip*** managed object.

ip_interface

Is the ***ip_interface*** (interface-specific) instance of the managed object.

ip_address

Is the specific IP address expressed in dotted decimal notation.

variable_name

Is the variable name.

You express the pathname to any ***ip*** routing table variable in the following format:

ip.ip_route_table.net_tree.variable_name

where:

ip

Is the ***ip*** managed object.

ip_route_table

Is (along with ***net_tree***) the routing table instance of the managed object.

net_tree

Is (along with ***ip_route_table***) the routing table instance of the managed object.

variable_name

Is the variable name.

13.2.1 IP Interface Variables

The ***ip interface*** variables, in alphabetical order, are as follows:

address

Contains the 32-bit IP address of interface **<ip_address>** expressed as a decimal integer.

drop.dest_unknown

Contains the number of IP datagrams dropped by interface **<ip_address>** because the IP header contained an unknown or corrupt IP destination address.

drop.filtered

Contains the number of IP datagrams dropped by interface **<ip_interface>** in response to source- address or destination-address filters established at system configuration.

drop.frag_error

Contains the number of IP datagrams dropped by interface **<ip_interface>** because of its inability to fragment a datagram. Interface **<ip_interface>** forwards datagrams up to 1500 bytes in length; longer datagrams must be fragmented. Should the datagram originator forbid fragmenting (by setting the Do Not Fragment bit in the IP header), interface **<ip_address>** drops the datagram, increments this counter, and issues an Internet Control Message Protocol (ICMP) Destination Unreachable message.

drop.header_format

Contains the number of IP datagrams dropped by interface **<ip_address>** because of an unparsable or corrupt IP header.

drop.reassembly_busy

Contains the number of IP datagrams dropped by interface **<ip_interface>** because of the lack of resources at the message destination.

drop.ttl_exceeded

Contains the number of IP datagrams dropped by interface **<ip_interface>** because the IP header Time field had reached 0.

drop.xsumerror

Contains the number of IP datagrams dropped by interface **<ip_interface>** because of a faulty IP header checksum.

icmp_rx.dest_unreachable

Contains the number of ICMP Destination Unreachable messages received by interface **<ip_address>**. Such messages indicate that the originating system cannot route or deliver datagrams.

icmp_rx.echo_request

Contains the number of ICMP Echo Request messages received by interface **<ip_address>**. Such messages test whether a destination system is reachable.

icmp_rx.frag_error

Contains the number of ICMP Destination Unreachable messages with a Code field value of 4 (indicating that a datagram could not be fragmented) received by interface **<ip_address>**. Such messages indicate that an IP datagram previously routed through interface **<ip_address>** cannot be handled by a subsequent router.

icmp_rx.param_problem

Contains the number of ICMP Parameter Problem messages received by interface **<ip_address>**. Such messages report faulty IP datagram headers.

icmp_rx.redirect

Contains the number of ICMP Redirect messages received by interface **<ip_address>**. Such messages inform the recipient of a more optimum IP route.

icmp_rx.ttl

Contains the number of ICMP Time Exceeded messages received by interface **<ip_address>**. Such messages are generated when a datagram's hop count reaches 0.

icmp_rx.xsum_error

Contains the number of received ICMP messages dropped by interface **<ip_interface>** because of a faulty ICMP checksum.

icmp_tx.dest_unreachable

Contains the number of ICMP Destination Unreachable messages transmitted by interface **<ip_address>**. Such messages indicate that the originating system cannot route or deliver datagrams.

icmp_tx.echo_reply

Contains the number of ICMP Echo Reply messages transmitted by interface **<ip_address>**. Such messages test whether a destination system is reachable.

icmp_tx.frag_error

Contains the number of ICMP Destination Unreachable messages with a Code field of 4 (indicating that a datagram could not be fragmented) transmitted by interface **<ip_address>**. Such messages indicate that **<ip_address>** cannot fragment an IP datagram longer than 1500 bytes, or that the DF bit of the IP datagram was set (do not fragment).

icmp_tx.param_problem

Contains the number of ICMP Parameter Problem messages transmitted by interface **<ip_address>**. Such messages report faulty IP headers.

icmp_tx.redirect

Contains the number of ICMP Redirect messages transmitted by interface **<ip_address>**. Such messages inform the recipient of a more optimum IP route.

icmp_tx.ttl

Contains the number of ICMP Time Exceeded messages transmitted by interface **<ip_address>**. Such messages are generated when a datagram's hop count reaches zero.

mask

Contains the 32-bit sub-net mask of interface **<ip_address>** expressed as an unformatted decimal integer.

rx

Contains the total number of IP datagrams received by interface **<ip_address>**.

tx

Contains the total number of IP datagrams transmitted by interface **<ip_address>**.

ulp

Contains the total number of IP datagrams delivered by the router to one of three upper level protocols (Internet Control Message Protocol, Transmission Control Protocol, or User Datagram Protocol) for processing.

13.2.2 IP Routing Table Variables

The ***ip routing table*** variables, in alphabetical order, are as follows:

cache_hits

Contains the number of times the router found the next hop to a destination address in its routing cache. Upon receiving a packet for forwarding, the router first checks its cache to determine the next hop to the destination address. If the routing cache does not contain this information, the router then accesses the IP routing table.

mem_used

Contains the number of bytes of memory used by the IP routing table.

node_count

Contains the current number of networks contained in the IP routing table.

node_depth

Contains the number of levels in the IP routing table.

search_count

Contains the number of searches through the IP routing table. Upon receiving a packet for forwarding, the router first checks its cache to determine the next hop to the destination address. If the routing cache does not contain this information, the router then accesses the IP routing table.

search_depth

Contains the total depth of searches through the routing tree.

13.3 Key Information Base

The ***key*** information base mirrors the state of specific software modules. Access to the key information base allows you to view remotely which software options are installed in the system.

The system maintains ***key*** information as a 2-level tree, with the upper level identifying the ***key*** managed object and the lower level containing the actual ***key*** variables.

Figure 13-2 illustrates the structure of the ***key*** information base.

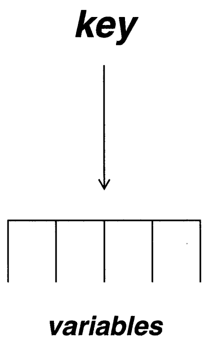


Figure 13-2 *key* Information Base

You use the NCL `LIST` command to display the ***key*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***key*** variable in the following format:

key.variable_name

where:

key

is the ***key*** managed object.

variable_name

Is the variable name.

The ***key*** variables, in alphabetical order, are as follows:

---

drs

Contains a code value indicating the presence or absence of the DECnet Router: 1 indicates that the DECnet Router software resides within the system; 0 indicates that the DECnet Router software is not present within the system.

---

ip

Contains a code value indicating the presence or absence of the IP Router: 1 indicates that the IP Router software resides within the system; 0 indicates that the IP Router software is not present within the system.

---

lb

Contains a code value indicating the presence or absence of the Bridge: 1 indicates that the Bridge software resides within the system; 0 indicates that the Bridge software is not present within the system.

---

xns

Contains a code value indicating the presence or absence of the IPX/XNS Router: 1 indicates that the IPX/XNS Router software resides within the system; 0 indicates that the IPX/XNS Router software is not present within the system.

---

x25

Contains a code value indicating the presence or absence of X.25 service: 1 indicates that the X.25 software resides within the system; 0 indicates that the X.25 software is not present within the system.

13.4 Line Information Base

The ***line*** information base contains variables that describe alarm detection and error conditions across each T1 or E1 line.

The system maintains ***line*** information as a three- level tree. The top-most level identifies the ***line*** managed object. The middle level consists of a configuration-dependent number of branches, with each branch identifying a specific physical connector. The lowest level contains the actual variables. The system maintains distinct variables for T1 and E1 lines.

Figure 13-3 illustrates the structure of the ***line*** information base.

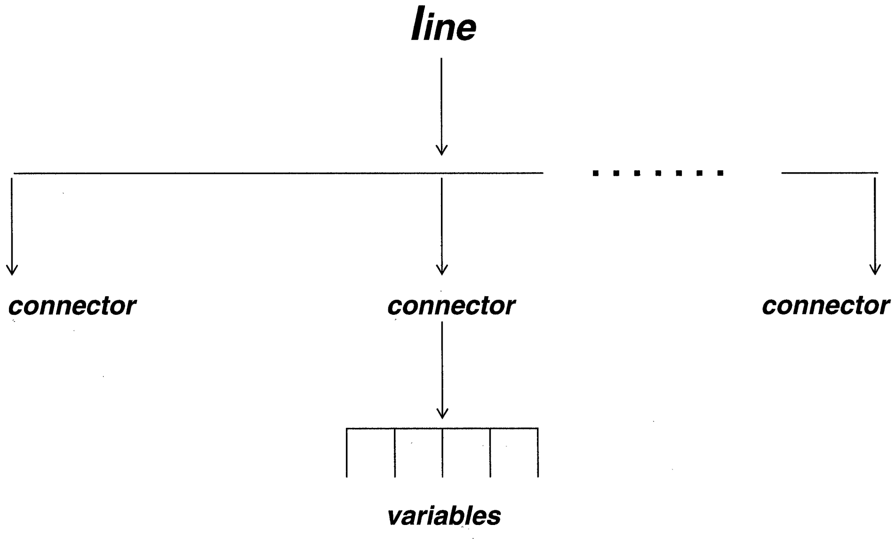


Figure 13-3 *line* Information Base

You use the NCL `LIST` command to display all, or a portion of, the ***line*** information base, and the NCL `GET` command to obtain the value of any variable within the information base. You express the pathname to any ***line*** variable in the following format:

line.conn.variable_name

where:

line

Is the ***line*** managed object.

conn

Is a specific T1 or E1 connector. T1 lines are physically enabled by a DS1 connector, while E1 lines are physically enabled by an E1 connector. Thus, ***slot5_ds1_1*** specifies a T1 line physically connected to the DS1_1 connector in Slot 5, and ***slot4_e1_2*** specifies an E1 line physically connected to the E1_2 connector in Slot 4.

variable_name

Is the variable name.

13.4.1 T1 Line Variables

The ***line T1*** variables, in alphabetical order, are as follows:

bipolar_vio

Contains the number of T1 bipolar violations received on the line enabled by **<slot#_ds1_#>**. A bipolar violation occurs when two consecutive ones of the same polarity are received.

bit_err

Contains the number of fT-bit errors (terminal framing pattern errors) and FPS (Framing Pattern Sequence) errors.

err_sframe

Contains the number of times the line enabled by **<slot#_ds1_#>** received a superframe (D4 frame) that contained an error.

out_of_frame

Contains the number of times the line enabled by **<slot#_ds1_#>** lost frame alignment. Frame alignment is lost when the receiver fails to detect the frame alignment pattern.

rcv_lcar

Contains the number of times the line enabled by **<slot#_ds1_#>** lost the carrier signal. Loss of carrier is defined as the reception of an excessive number of consecutive zeros.

rred_alarm

Contains the number of Red Alarms received on the line enabled by **<slot#_ds1_#>**. A Red Alarm indicates a locally detected failure in a sink device. The remote site generates a Red Alarm when it detects a local failure (for example, loss of synchronization, incoming signal failure, or hardware malfunctions).

ryel_alarm

Contains the number of Yellow Alarms received on the line enabled by **<slot#_ds1_#>**. A Yellow Alarm is a Remote Alarm Indication (RAI) sent back to the originating source of a failed T1 transmission path. The reception of a Yellow Alarm denotes that the remote site is not receiving valid T1 data frames.

13.4.2 E1 Line Variables

The ***line E1*** variables, in alphabetical order, are as follows:

bipolar_vio

Contains the number of E1 bipolar violations received on the line enabled by **<slot#_e1_#>**. A bipolar violation occurs when two consecutive ones of the same polarity are received.

crc4_err

Contains the number of times the line enabled by **<slot#_e1_#>** received a frame that contained an erroneous checksum.

frame_err

Contains the number of times the line enabled by **<slot#_e1_#>** received a frame that contained an error.

mfs_err

Contains the number of times the line enabled by **<slot_e1_#>** received loss of multiframe synchronization.

rx_dx_mfm_alm

Contains the number of times the line enabled by **<slot#_e1_#>** received a loss of multiframe alignment alarm from the remote end of the line.

rx_rem_alm

Contains the number of times the line enabled by **<slot#_e1_#>** received an alarm from the remote end of the line.

rx_sig_all_1

Contains the number of instances of receiving a framed all-1s signal.

rx_unfrm_all_1

Contains the number of instances of receiving an unframed all-1s signal.

sync_loss

Contains the number of instances of loss of synchronization.

13.5 Manager Information Base

The ***mgr*** (manager) information base contains two proprietary control objects, ***auto_enable*** and ***disk***:

- ***auto_enable*** attributes reflect the value of the global Auto Enable parameter (refer to the *Configuration Guide* for information on setting this parameter).
- ***disk*** attributes reflect the current status of the system disk drive.

The NCL `LIST` and `GET` commands provide no additional information regarding the ***mgr*** information base.

13.6 Memory Information Base

The ***mem*** (memory) information base contains variables that describe system memory management.

The system maintains ***mem*** information as a 4-level tree. The top-most level identifies the ***mem*** managed object. The next lower level consists of a configuration- dependent number of branches, with each branch identifying a slot number (***slot_#)***. The next level consists of two branches, with each branch identifying a memory type (***local*** or ***global***). The lowest level contains the actual ***mem*** variables. The information base contains the same variables for both global and local memory.

Figure 13-4 illustrates the structure of the ***mem*** information base.

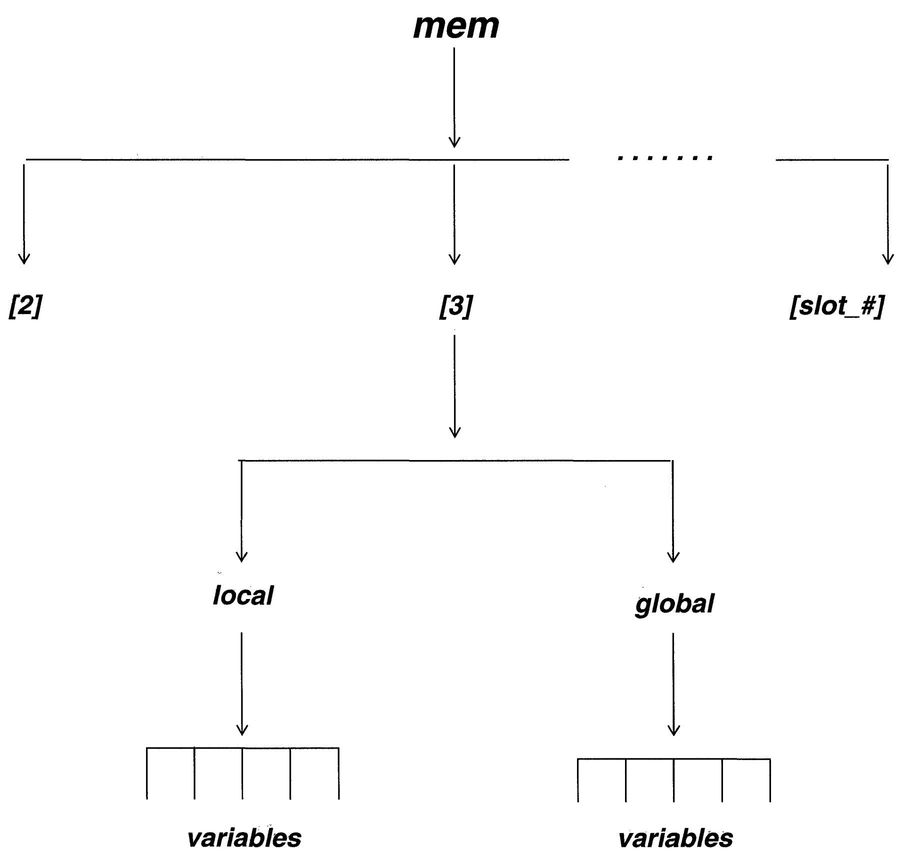


Figure 13-4 *mem* Information Base

You use the NCL `LIST` command to display all, or a portion of, the ***mem*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***mem*** variable in the following format:

mem[slot_#].type.variable_name

where:

mem

Is the ***mem*** managed object.

type

Is the memory type (***local*** or ***global***).

variable_name

Is the variable name.

The ***mem*** variables, in alphabetical order, are as follows:

alloc_bytes

Contains the number of currently allocated bytes.

alloc_seg_cnt

Contains the number of currently allocated memory segments, and thus reflects a degree of memory fragmentation.

free_bytes

Contains the number of available (unallocated) bytes.

free_seg_cnt

Contains the current number of free memory segments. A free memory segment is an unused contiguous memory block of greater than 16 bytes.

slab_cnt

Contains the current number of discretely managed memory areas. Each slab is further broken down into smaller contiguous areas called segments. For global memory, ***slab_cnt*** should contain 1.

total_bytes

Contains the number of currently installed bytes.

13.7 Name Information Base

The ***name*** information base contains variables that describe the operations and structure of the name server.

The system maintains ***name*** information as a 3-level tree. The top-most level identifies the ***name*** managed object. The middle level consists of a single branch to the system master slot (***[2]***). The bottom level contains the actual ***name*** variables.

Figure 13-5 illustrates the structure of the ***name*** information base.

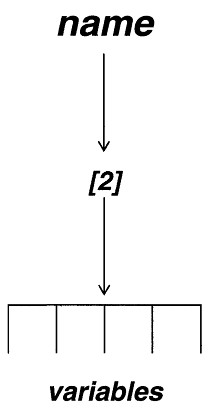


Figure 13-5 *name* Information Base

You use the NCL `LIST` command to display all, or a portion of the ***name*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***name*** variable in following format:

name[2].variable_name

where:

name

Is the ***name*** managed object.

[2]

Is the system master slot.

variable_name

Is the variable name.

The ***name*** variables, in alphabetical order, are as follows:

alias_cnt

Contains the number of aliased objects added to the name-server table. This variable should always contain 0.

create_cnt

Contains the number of objects added to the name-server table.

cur_cnt

Contains the current number of objects in the name-server table.

cur_mem_used

Contains the current number of bytes occupied by the name-server table.

delete_cnt

Contains the number of objects deleted from the name-server table.

find_cnt

Contains the number of name-server table accesses that resulted in successful resolutions.

list_cnt

This counter is not currently implemented. It should always contain 0.

update_cnt

Contains the number of updates to name-server table entries.

13.8 SNMP Information Base

The ***snmp*** (Simple Network Management Protocol) information base contains variables that describe the transmission and reception of User Data Protocol (UDP) datagrams delivered to, or originated by the SNMP management agent software.

The system maintains ***snmp*** information as a 2-level tree, with the upper level identifying the ***snmp*** managed object and the lower level containing the actual ***snmp*** variables.

Figure 13-6 illustrates the structure of the ***snmp*** information base.

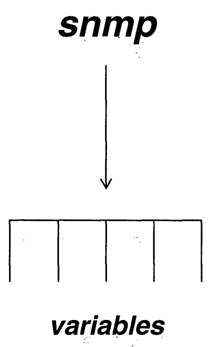


Figure 13-6 *snmp* Information Base

You use the NCL `LIST` command to display all, or a portion of, the ***snmp*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***snmp*** variable in the following format:

snmp.variable_name

where:

snmp

Is the ***snmp*** managed object.

variable_name

Is the variable name.

The ***snmp*** variables, in alphabetical order, are as follows:

snmpbadcommunity

Contains the number of incoming requests dropped by the SNMP management-agent software because either (1) the community name was unknown, or (2) the originating host was not a member of a known community.

snmpbadtype

Contains the number of incoming requests dropped by the SNMP management-agent software because the protocol data unit (PDU) specified an invalid operation code.

snmpinpkts

Contains the number of User Data Protocol (UDP) datagrams delivered to the SNMP management-agent software.

snmpoutpkts

Contains the number of User Data Protocol (UDP) datagrams originated by the SNMP management-agent software.

snmpprocerrs

Contains the number of syntactically correct SNMP messages that could not be processed -- such messages might request an unknown variable, contain an invalid instance identification, or contain a violation of access-control restrictions.

snmptotalrequested

Contains the total number of individual variables whose values have been requested in incoming SNMP PDUs. An approximation or the number of variables requested in each SNMP PDU can be computed be dividing ***snmptotalrequested*** by ***snmpinpkts***.

13.9 System Services Information Base

The ***svc*** (system services) information base contains variables that describe the private memory management function of the system management entity. This function maintains dynamic information on the memory space available to active tasks. The ***svc*** information base also contains counts of hardware interrupts.

The system maintains ***svc memory management*** information as a 5-level tree. The top-most level identifies the ***svc*** managed object. The next lower level contains a configuration-dependent number of branches, with each branch identifying a slot number (***slot_#***). The next level identifies the memory management instance (***sme***). The next level identifies one of four specific memory management tables (***ref*** for the reference table, ***act*** for the action table, ***sym*** for the symbol table, and ***txt*** for text string table). The lowest level contains the actual ***svc*** memory management variables. The system maintains identical variable sets for all tables.

The system maintains ***svc task-specific memory availability*** information as a 4-level tree. The top-most level identifies the managed object (***svc***). The next lower level contains a configuration-dependent number of branches, with each branch identifying a slot number (***slot_#***). The next level identifies the task-specific memory availability instance (***task_free_stk***). The lowest level contains the actual ***svc*** memory availability variables.

The system maintains ***svc hardware interrupt*** information as a 3-level tree. The top-most level identifies the managed object (***svc***). The middle level contains a configuration- dependent number of branches, with each branch identifying a slot number (***slot_#***). The next level identifies the hardware- interrupt variables (***spurious_int*** or ***nmi_ints***).

Figure 13-7 illustrates the structure of the ***svc*** information base.

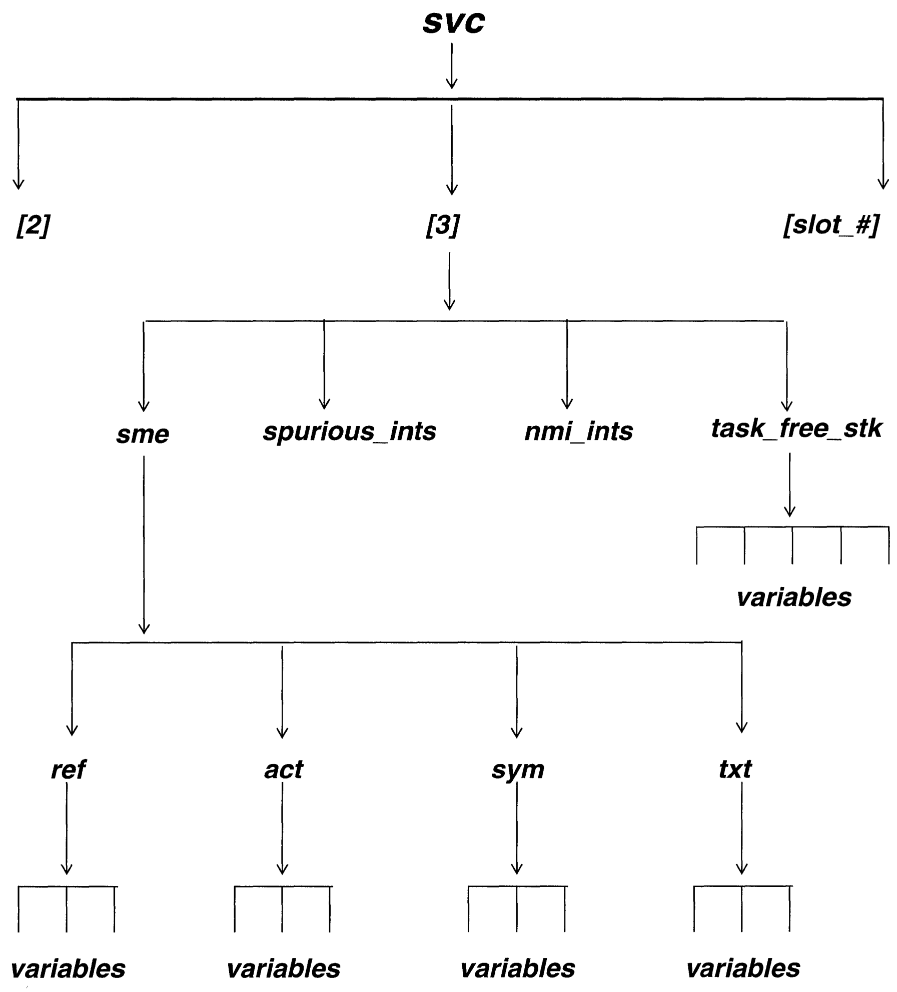


Figure 13-7 ***svc*** **Information Base**

You use the NCL **LIST** command to display all, or a portion of, the ***svc*** information base, and the NCL **GET** command to obtain the value of any variable within the information base.

You express the pathname to any ***svc memory management*** variable in the following format:

svc[slot_#].sme.table_type.variable_name

where:

svc

Is the ***svc*** managed object.

[slot_#]

Is the slot number (allowable range is 2 to 14 depending on system type).

sme

Is the memory management instance.

table_type

Is the table type (***ref*** for reference, ***act*** for action, ***sym*** for symbol, ***txt*** for text).

variable_name

Is the variable name.

You express the pathname to any ***svc memory availability*** variable in the following format:

svc[slot_#].task_free_stk[#]

where:

svc

Is the ***svc*** managed object.

[slot_#]

Is the slot number (allowable range is 2 to 14 depending on system type).

task_free_stk

Is the memory availability instance.

[#]

Is the system-assigned number that identifies the task.

You express the pathname to any ***svc hardware-interrupt*** variable in the following format:

svc[slot_#].variable_name

where:

svc

Is the ***svc*** managed object.

[slot_#]

Is the slot number (allowable range is 2 to 14 depending on system type).

variable_name

Is the variable name (***spurious_ints*** or ***nmi_ints***).

13.9.1 SVC Memory Management Variables

The ***svc memory management*** variables, in alphabetical order, are as follows:

alloc

Contains the number of objects allocated, and in use, in ***space***.

install

Contains the current number of objects contained in ***space***.

space

Contains the size of the memory area in bytes.

13.9.2 SVC Task-Specific Memory Availability Variables

The ***svc task-specific memory management*** variables, in alphabetical order, are as follows:

task_free_stk[#]

Contains the size (in bytes) of the memory stack currently available to task ***[#]***.

13.9.3 SVC Hardware Interrupt Variables

The ***svc hardware-interrupt*** variables, in alphabetical order, are as follows:

nmi_ints

Contains the number of non-maskable interrupts.

spurious_ints

Contains the number of spurious interrupts.

13.10 TCP Information Base

The ***tcp*** (Transmission Control Protocol) information base contains variables that describe the exchange of ***tcp*** segments between communicating ***tcp*** peer entities.

The system maintains ***tcp*** information as a 2-level tree, with the upper level identifying the ***tcp*** managed object and the lower level containing the actual ***tcp*** variables.

Figure 13-8 illustrates the structure of the ***tcp*** information base.

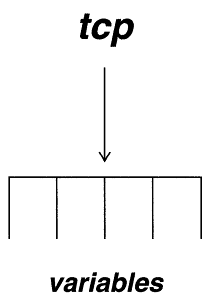


Figure 13-8 *tcp* Information Base

You use the NCL **`LIST`** command to display all, or a portion of, the ***tcp*** information base, and the NCL **`GET`** command to obtain the value of any variable within the information base.

You express the pathname to any ***tcp*** variable in the following format:

tcp.variable_name

where:

tcp

Is the ***tcp*** managed object.

variable_name

Is the variable name.

The ***tcp*** variables, in alphabetical order, are as follows:

acks_snt

Contains the number of positive acknowledgments transmitted by the system during TCP sessions.

app_dropped

Contains the number of TCP segments dropped because of invalid segment, port, and/or protocol information.

app_notcb

Contains the number of instances the TCP port was not found in the hash table. Consequently, TCP drops the segment.

badack

Contains the number of received faulty acknowledgment segments.

badflg

Contains the number of received TCP segments that contained unknown or faulty flags.

badopt

Contains the number of received TCP segments that requested unknown/unsupported TCP options.

badrst

Contains the number of received faulty TCP reset segments.

badseg

Contains the number of faulty received TCP segments.

badsum

Contains the number of received TCP segments that contained a bad checksum value.

dupack

Contains the number of duplicate acknowledgment segments.

dupseg

Contains the number of duplicate received TCP segments.

erract

Contains the total number of error messages sent by TCP.

hashcolls

Contains the number of times hashing of the TCP port information produced a collision with a previous port.

hashhits

Contains the number of times the hash was successful when matching TCP ports.

hashmiss

Contains the number of times the hash was missed during a TCP port search.

mem_err

Contains the number of memory errors.

mem_use

Contains the number of bytes of memory required for TCP operations.

net_dropped

Contains the number of TCP segments dropped because of an invalid data path.

net_notcb

Contains the number of times the hash table search of active TCP ports failed, and reset of the connection was attempted.

no_mem

Contains the number of instances TCP could not function because of lack of memory resources.

nopact

Contains the number of times a received TCP segment required no action.

pkts_rcv

Contains the number of TCP segments received by the system during TCP sessions.

pkts_snt

Contains the number of TCP segments transmitted by the system during TCP sessions.

rcv_full

Contains the number of times TCP segments were dropped because the receive window was closed.

rehashes

Contains the number of times the TCP port table was rehashed. The table is rehashed when a connection is closed, or when the control block hash table requires rehashing.

reseq

Contains the number of packets resequenced by the system.

reseq_drop

Contains the number of elements dropped because of resequencing.

reseq_full

Contains the number of times TCP had a full resequencing queue.

reseq_mer

Contains the number of resequenced packets that were linked to form a larger packet.

retx

Contains the number of packets retransmitted by the system.

retx_full

Contains the number of times the TCP retransmit buffer was full.

rx_bytes

Contains the number of bytes received by the system during TCP sessions.

segoos

Contains the number of segments received out of sequence.

sess_cur

Contains the current number of TCP connections.

sess_tot

Contains the number of TCP connections since the system last booted.

snd_full

Contains the number of times the TCP send window was full.

toobig

Contains the number of times a TCP segment was not sent because the receive window size was too small.

tx_bytes

Contains the number of bytes transmitted by the system during TCP sessions.

13.11 Telnet Information Base

The ***telnet*** information base contains variables that describe virtual-terminal connections between the system and a remote device.

The system maintains ***telnet*** information as a 2-level tree, with the upper level identifying the ***telnet*** managed object and the lower level containing the actual ***telnet*** variables.

Figure 13-9 shows the structure of the ***telnet*** information base.

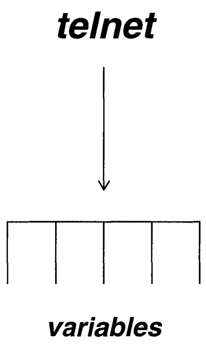


Figure 13-9 *telnet* Information Base

You use the NCL `LIST` command to display all, or a portion of, the ***telnet*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***telnet*** variable in the following format:

telnet.variable_name

where:

telnet

Is the ***telnet*** managed object.

variable_name

Is the variable name.

The ***telnet*** variables, in alphabetical order, are as follows:

inp.bad_opt

Contains the number of incoming TCP segments that requested unknown or unsupported telnet options.

inp.dropped

Contains the number of incoming TCP segments that were dropped because of lack of processing resources.

inp.no_if

Contains the number of incoming TCP segments that were dropped for lack of an interface.

inp.too_big

Contains the number of incoming TCP segments that were dropped because they exceeded the MTU.

mem_err

Contains the number of memory errors.

mem_use

Contains the number of bytes of memory used by Telnet.

no_mem

Contains the number of instances telnet was unable to function because of the lack of memory resources.

out.dropped

Contains the number of outgoing TCP segments that were not sent because of lack of processing resources.

out.iac_fnd

Contains the number of outgoing telnet escape sequences that were not transmitted.

out.no_if

Contains the number of outgoing TCP segments that were dropped for lack of an interface.

out.too_big

Contains the number of outgoing TCP segments that were dropped because they exceeded the MTU.

rx_bytes

Contains the number of bytes received by the system while connected to a remote terminal by means of Telnet.

sess_cur

Contains the current number of Telnet connections. The system supports a maximum of two simultaneous telnet sessions.

sess_tot

Contains the number of Telnet connections since the system last booted.

tx_bytes

Contains the number of bytes transmitted by the system while connected to a remote terminal by means of Telnet.

13.12 TFTP Information Base

The ***tftp*** (Trivial File Transfer Protocol) information base contains variables that describe file transfers between the system and a remote device.

The system maintains ***tftp*** information as a 2-level tree, with the upper level identifying the ***tftp*** managed object. The lower level contains the actual ***tftp*** variables.

Figure 13-10 shows the structure of the ***tftp*** information base.

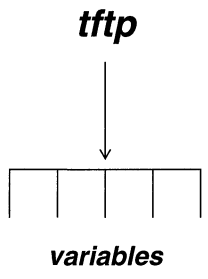


Figure 13-10 ***tftp*** **Information Base**

You use the NCL `LIST` command to display all, or a portion of, the ***tftp*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***tftp*** variable in the following format:

tftp.variable_name[#]

where:

tftp

Is the ***tftp*** managed object.

variable_name

Is the variable name.

[#]

Is the system-assigned interface number name.

The ***tftp*** variables, in alphabetical order, are as follows:

aborted

Contains the number of TFTP sessions that were prematurely terminated.

errin

Contains the number of received TFTP ERROR packets (Opcode = 5).

errout

Contains the number of transmitted TFTP ERROR packets (Opcode = 5).

filesin

Contains the number of files successfully transmitted to the system from a remote device by means of the TFTP.

filesout

Contains the number of files successfully transmitted by the system to a remote device by means of the TFTP.

rrqin

Contains the number of received TFTP READ REQUEST packets (Opcode = 1).

rxmits

Contains the total number of TFTP packets retransmitted by the system.

wrqin

Contains the number of received TFTP WRITE REQUEST packets (Opcode = 2).

13.13 Timer Information Base

The ***timer*** information base contains variables that describe the scheduling and issuance of system-generated timers.

The system maintains ***timer*** information as a 3-level tree. The top-most level identifies the ***timer*** managed object. The middle level consists of a configuration-dependent number of branches, with each branch identifying a slot number (***slot_#***). The bottom level contains the actual ***timer*** variables.

Figure 13-11 illustrates the structure of the ***timer*** information base.

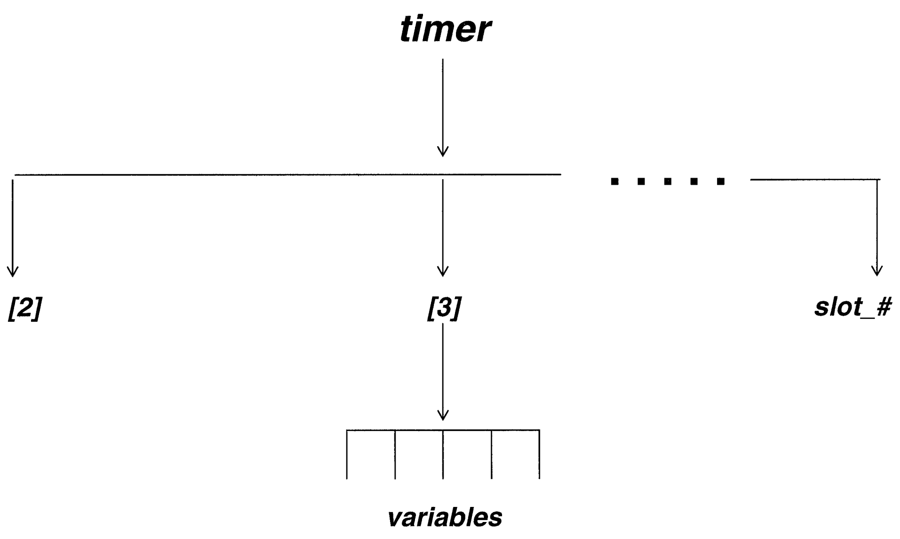


Figure 13-11 ***timer*** **Information Base**

You use the NCL `LIST` command to display all, or a portion of, the ***timer*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

The pathname to any ***timer*** variable takes the following format:

timer[slot_#].variable_name

where:

timer

Is the ***timer*** managed object.

[slot_#]

Is the slot number (allowable range is 2 to 14 depending on system type).

variable_name

Is the variable name.

The ***timer*** variables, in alphabetical order, are as follows:

cancel_cnt

Contains the number of timers cancelled by the system prior to the expiration of the timer.

expire_cnt

Contains the number of timer-related interrupts. Such an interrupt is generated when the timer reaches zero.

race_cnt

Contains the number of simultaneous occurrences of the expiration of the timer and the cancellation of the timer.

set_cnt

Contains the number of timers scheduled by the system.

13.14 XNS Information Base

The ***xrx*** information base is a composed of:

- A set of variables that describe transmission and reception of packets across each XNS interface
- A set of variables that describe aggregate Internet Datagram Protocol (IDP)/Error Protocol activity
- An XNS addressing table
- An XNS routing table

The system maintains ***xrx interface-specific*** information as a 3-level tree. The top-most level identifies the ***xrx*** managed object. The middle level consists of a configuration-dependent number of branches, with each branch identifying a specific Xerox interface address (***if***). The lowest level contains the actual ***xrx*** interface-specific variables.

The system maintains ***xrx IDP/Error Protocol*** information as a 4-level tree. The top-most level identifies the ***xrx*** managed object. The next two lower levels identify the IDP instance of the managed object (***mib*** and ***idp***). The lowest level contains the IDP variables.

Figure 13-12 illustrates the structure of the ***xrx*** information base.

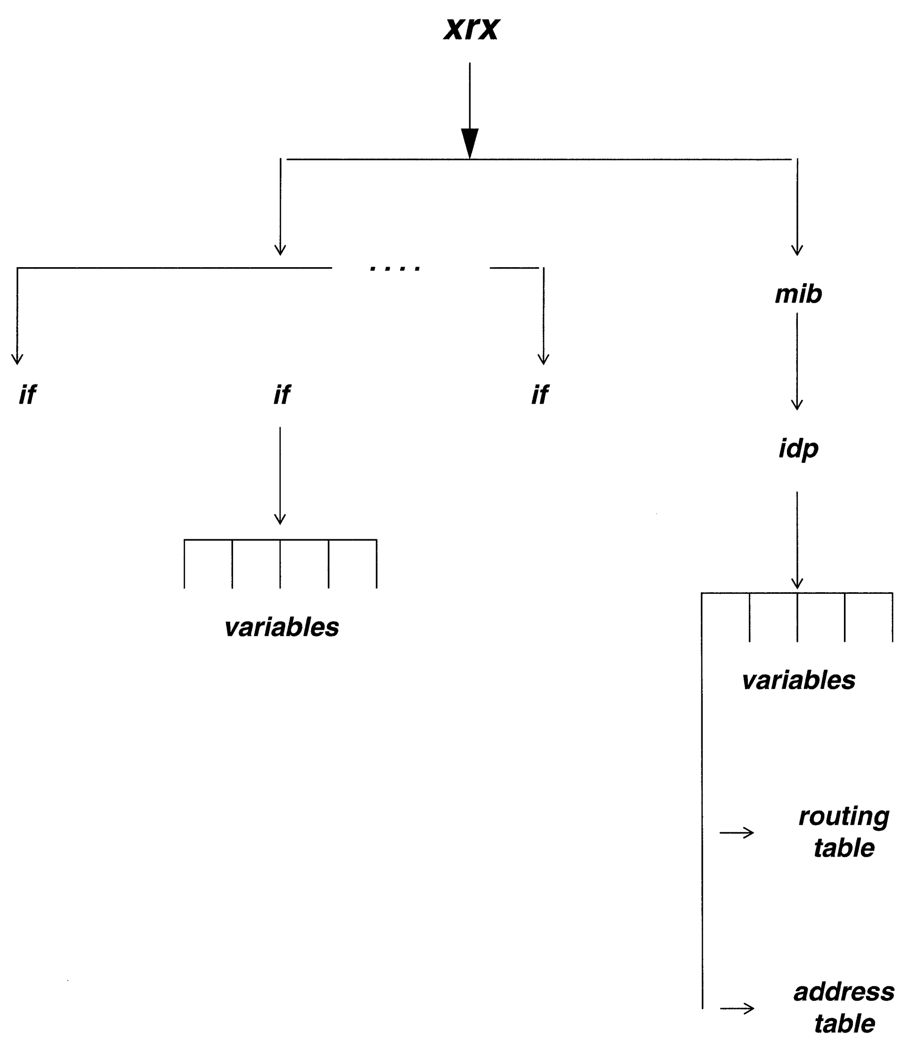


Figure 13-12 ***xrx*** **Information Base**

You use the NCL **`LIST`** command to display all, or a portion of, the ***xrx*** information base, and the NCL **`GET`** command to obtain the value of any variable within the information base. You use the NCL **`RGETXR`** command to access ***xrx*** tables.

You express the pathname to any ***xrx interface-specific*** variable in the following format:

xrx.if.variable_name

where:

xrx

Is the ***xrx*** managed object.

if

Is the specific XNS address expressed as an 8-digit hexadecimal value.

variable_name

Is the variable name.

You express the pathname to any ***xrx IDP/Error Protocol*** variable in the following format:

xrx.mib.idp.variable_name

where:

xrx

Is the ***xrx*** managed object.

mib

Is (along with ***idp***) the protocol- specific instance of the managed object.

idp

Is (along with ***mib***) the protocol-specific instance of the managed object.

variable_name

Is the variable name.

13.14.1 XNS Interface-Specific Variables

The ***xrx interface-specific*** variables, in alphabetical order, are as follows:

drop

Contains the total number of IDP datagrams dropped (for whatever reason) by interface **<if>**.

rx

Contains the total number of IDP datagrams received by interface **<if>**.

tx

Contains the total number of IDP datagrams transmitted by interface **<if>**.

ulp

Contains the total number of IDP datagrams delivered by the router to an upper-level protocol (for example, RIP, Echo, Error) for processing.

13.14.2 XNS Protocol Variables

The ***xrx protocol*** variables, in alphabetical order, are as follows:

errsdestbadsock

Contains the number of destination-host- generated Error Protocol packets, with an Error Number of 2, that were relayed by the router. This Error Number indicates that the destination host received an IDP packet addressed to an unknown socket.

errsdestchksum

Contains the number of destination-host-generated Error Protocol packets, with an Error Number of 1, that were relayed by the router. This Error Number indicates that the destination host received an IDP packet that contained a faulty or corrupted checksum.

errsdesthdrlen

Contains the number of packets rejected by a destination host because the packet header was of insufficient length.

errsdestnoresrcs

Contains the number of destination-host-generated Error Protocol packets, with an Error Number of 3, that were relayed by the router. This Error Number indicates that the destination host discarded an IDP packet because of lack of processing resources.

errsdestproto

Contains the number of packets rejected by a destination host because the protocol type field contained an invalid or unknown value.

errsdestunspec

Contains the number of destination-host- generated Error Protocol packets, with an Error Number of 0, that were relayed by the router. This Error Number indicates that the destination host rejected an IDP packet for unspecified reasons.

errssuppressed

Contains the number of packets dropped because their length was below minimum.

errsxitchksum

Contains the number of router-generated Error Protocol packets with an Error Number of 1001. This Error Number indicates that the router received an IDP packet that contained a faulty or corrupted checksum.

errsxithopcnt

Contains the number of router-generated Error Protocol packets with an Error Number of 1003. This Error Number indicates that the packet had passed through more than the maximum number of routers before arriving at its destination.

errsxittoobig

Contains the number of router-generated Error Protocol packets with an Error Number of 1004. This Error Number indicates that the packet is too long for the router to relay.

errsxitunreach

Contains the number of router-generated Error Protocol packets with an Error Number of 1002. This Error Number indicates that the router cannot reach the packet destination.

errsxitunspec

Contains the number of router-generated Error Protocol packets with an Error Number of 1000. This Error Number indicates that the router rejected an IDP packet for unspecified reasons.

forwarding

Contains an integer switch indicating the system's function within the extended Xerox network. A value of 1 indicates that the system is acting as a gateway (routes and forwards datagrams); a value of 2 indicates that the system is acting as a host (does not route and forward datagrams).

forwdatagrams

Contains the number of received IDP datagrams not addressed to the Xerox router. The router attempts to forward these datagrams to their ultimate destination.

inadderrors

Contains the number of IDP datagrams discarded because of invalid destination address fields.

indelivers

Contains the number of IDP datagrams delivered to IDP user protocols (including Error Protocol).

indiscards

Contains the number of valid input IDP datagrams discarded because of insufficient system resources (lack of buffer space).

inhderrors

Contains the number of IDP datagrams discarded because of errors in their IDP headers. Such errors include faulty checksums, format errors, and transport control (hop count) errors.

inreceives

Contains the total number of all IDP datagrams (including those received in error) received from all interfaces.

inunknownprotos

Contains the number of IDP datagrams discarded because of an incorrect or corrupted value in the Protocol Type field in the IDP header.

outdiscards

Contains the number of valid output IDP datagrams discarded because of insufficient system resources (lack of buffer space).

outnoroutes

Contains the number of IDP datagrams discarded because no route could be found to transmit them to their destination.

outrequests

Contains the number of IDP datagrams generated by local IDP-user protocols (including Error).

13.15 IPX Information Base

The ***ipx*** information base is a composed of:

- A set of variables that describe transmission and reception of packets across each IPX interface
- A set of variables that describe aggregate Internet Datagram Protocol (IDP) Protocol activity
- IPX addressing table
- A SAP table
- An IPX routing table

The system maintains ***ipx interface-specific*** information as a 3-level tree. The top-most level identifies the ***ipx*** managed object. The middle level consists of a configuration-dependent number of branches, with each branch identifying a specific IPX interface address (***if***). The lowest level contains the actual ***ipx*** interface-specific variables.

The system maintains ***ipx IDP Protocol*** information as a 4-level tree. The top-most level identifies the ***ipx*** managed object. The next two lower levels identify the IDP instance of the managed object (***mib*** and ***idp***). The lowest level contains the IDP variables.

Figure 13-13 illustrates the structure of the ***ipx*** information base.

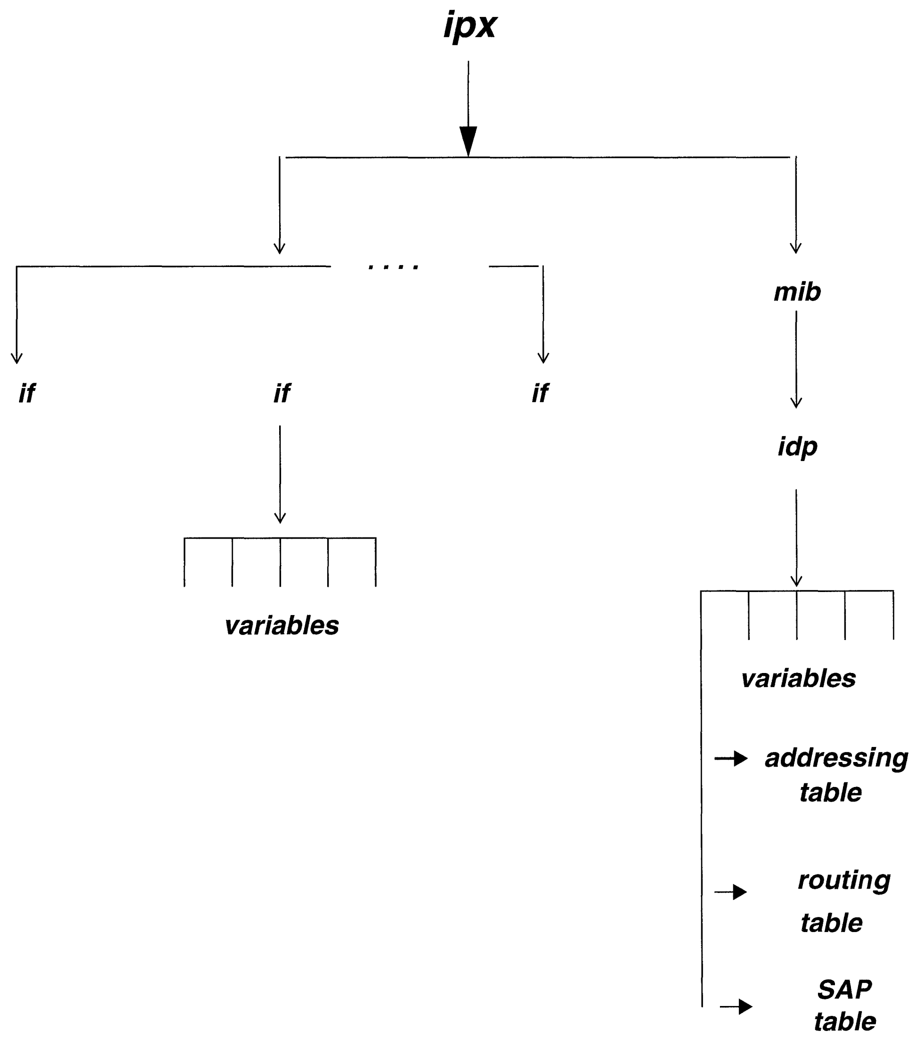


Figure 13-13 *ipx* Information Base

You use the NCL `LIST` command to display all, or a portion of, the ***ipx*** information base, and the NCL `GET` command to obtain the value of any variable within the information base. You use the NCL `RGETIR` and `RGETIS` commands to access ***ipx*** tables.

You express the pathname to any ***ipx interface-specific*** variable in the following format:

ipx.if.variable_name

where:

ipx

Is the ***ipx*** managed object.

if

Is the specific IPX address expressed as an 8-digit hexadecimal value.

variable_name

Is the variable name.

You express the pathname to any ***ipx IDP Protocol*** variable in the following format:

ipx.mib.idp.variable_name

where:

ipx

Is the ***ipx*** managed object.

mib

Is (along with ***idp***) the protocol- specific instance of the managed object.

idp

Is (along with ***mib***) the protocol-specific instance of the managed object.

variable_name

Is the variable name.

13.15.1 IPX Interface-Specific Variables

The ***ipx interface-specific*** variables, in alphabetical order, are as follows:

drop

Contains the total number of IDP datagrams dropped (for whatever reason) by interface **<if>**.

rx

Contains the total number of IDP datagrams received by interface **<if>**.

tx

Contains the total number of IDP datagrams transmitted by interface **<if>**.

ulp

Contains the total number of IDP datagrams delivered by the router to an upper level protocol (for example, RIP, Echo, Error) for processing.

13.15.2 IPX Protocol Variables

The ***ipx protocol*** variables, in alphabetical order, are as follows:

forwarding

Contains an integer switch indicating the system's function within the extended IPX network. A value of 1 indicates that the system is acting as a gateway (routes and forwards datagrams); a value of 2 indicates that the system is acting as a host (does not route and forward datagrams).

forwdatagrams

Contains the number of received IDP datagrams not addressed to the IPX router. The router attempts to forward these datagrams to their ultimate destination.

inadderrors

Contains the number of IDP datagrams discarded because of invalid destination address fields.

indelivers

Contains the number of IDP datagrams delivered to IDP user protocols (including Error Protocol).

indiscards

Contains the number of valid input IDP datagrams discarded because of insufficient system resources (lack of buffer space).

inhderrors

Contains the number of IDP datagrams discarded because of errors in their IDP headers. Such errors include faulty checksums, format errors, and transport control (hop count) errors.

inreceives

Contains the total number of all IDP datagrams (including those received in error) received from all interfaces.

inunknownprotos

Contains the number of IDP datagrams discarded because of an incorrect or corrupted value in the Protocol Type field in the IDP header.

outdiscards

Contains the number of valid output IDP datagrams discarded because of insufficient system resources (lack of buffer space).

outnoroutes

Contains the number of IDP datagrams discarded because no route could be found to transmit them to their destination.

outrequests

Contains the number of IDP datagrams generated by local IDP-user protocols (including Error).

13.16 X.25 Information Base

The ***x25*** information base contains variables that describe frame-level and packet-level transmission and reception activities across each X.25 circuit; it also contains variables that describe packet-level transmission and reception activities across each X.25 point-to-point dedicated switched virtual circuit.

The system maintains ***x25 circuit information*** as a 4-level tree. The top-most level identifies the ***x25*** managed object. The next level consists of a configuration-dependent number of branches, with each branch identifying an X.25 Point-to-Point, DDN, or PDN circuit (***cct_name***). The next lower level consists of two branches identifying the variable type (***frame*** or ***pkt***). The lowest level contains the actual *x25* circuit variables.

The system maintains ***x25 point-to-point dedicated virtual circuit*** information as a 3-level tree. The top-most level identifies the ***x25*** managed object. The middle level consists of an arbitrary number of branches, with each branch identifying an X.25 virtual circuit (***svc_name***). The lowest level contains the actual ***x25*** virtual circuit variables.

Figure 13-14 shows the structure of the ***x25*** information base.

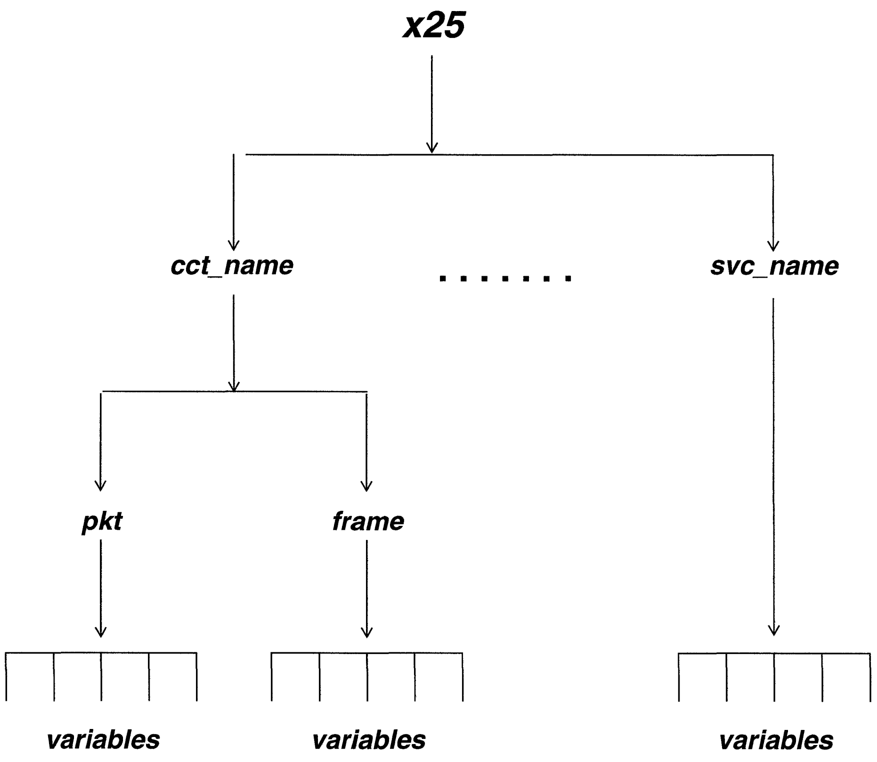


Figure 13-14 *x25* Information Base

You use the NCL `LIST` command to display all, or a portion of, the ***x25*** information base, and the NCL `GET` command to obtain the value of any variable within the information base.

You express the pathname to any ***x25 circuit*** variable in the following format:

x25.cct_name.variable_type.variable_name

where:

x25

Is the ***x25*** managed object.

cct_name

Is the X.25 circuit name.

variable_type

Is the variable type (either ***pkt*** for packet-level variables or ***frame*** for frame-level variables).

variable_name

Is the variable name.

The pathname to any ***x25 dedicated point-to-point switched virtual-circuit*** variable takes the following format:

x25.svc_name.variable_name

where:

x25

Is the ***x25*** managed object.

svc_name

Is X.25 virtual circuit name.

variable_name

Is the variable name.

13.16.1 X.25 Circuit Frame-Level Variables

The ***x25 circuit frame-level*** variables, in alphabetical order, are as follows:

bad_len_rx

Contains the number of supervisory FRMR (Frame Reject) frames received by X.25 circuit **<cct_name>** that contained W and X bits set to 1. This bit pattern indicates that the remote end has rejected a supervisory or unnumbered frame issued by X.25 circuit **<cct_name>** because the frame length was faulty.

bad_len_tx

Contains the number of supervisory FRMR (Frame Reject) frames transmitted by X.25 circuit **<cct_name>** that contained W and X bits set to 1. This bit pattern indicates that X.25 circuit **<cct_name>** has rejected a previously received supervisory or unnumbered frame because the frame length was faulty.

bad_nr_rx

Contains the number of supervisory FRMR (Frame Reject) frames received by X.25 circuit **<cct_name>** that contained a Z bit equal to 1. This bit pattern indicates that the remote end has rejected a frame issued by X.25 circuit **<cct_name>** because it contained a faulty receive sequence number.

bad_nr_tx

Contains the number of supervisory FRMR (Frame Reject) frames transmitted by X.25 circuit **<cct_name>** that contained a Z bit value equal to 1. This bit pattern indicates that X.25 circuit **<cct_name>** has rejected a previously received frame because the frame contained a faulty receive sequence number.

disc_rx

Contains the count of unnumbered DISC (Disconnect) frames received by X.25 circuit **<cct_name>**. DISC frames terminate the DCE/DTE link.

disc_tx

Contains the count of unnumbered DISC frames transmitted by X.25 circuit **<cct_name>**. DISC frames terminate the DCE/DTE link.

dm_rx

Contains the count of unnumbered DM (Disconnected Mode) frames received by X.25 circuit **<cct_name>**. DM frames indicate that the sending system is logically disconnected.

dm_tx

Contains the count of unnumbered DM (Disconnected Mode) frames transmitted by X.25 circuit **<cct_name>**. DM frames indicate that the sending system is logically disconnected.

frmr_rx

Contains the aggregate number of FRMR (Frame Reject) frames received by X.25 circuit **<cct_name>**. FRMR frames report specific error conditions.

frmr_tx

Contains the aggregate number of FRMR (Frame Reject) frames transmitted by X.25 circuit **<cct_name>**. FRMR frames report specific error conditions.

ignore_rx

Contains the aggregate number frames, received on X.25 circuit **<cct_name>**, that were not processed, generally because of insufficient length or because of a faulty address field (containing values other than 01 or 03).

info_rx

Contains the number of I (Information) frames received by X.25 circuit **<cct_name>**. I frames carry user data packets.

info_tx

Contains the number of I (Information) frames transmitted by X.25 circuit **<cct_name>**. I frames carry user data packets.

lev2_down

Contains the number of times that X.25 circuit **<cct_name>** went down in accordance with X.25 frame-level protocol.

lev2_up

Contains the number of times that X.25 circuit **<cct_name>** came up in accordance with X.25 frame-level protocol.

rej_rx

Contains the number of REJ (Reject) frames received by X.25 circuit **<cct_name>**. A REJ frame is a negative acknowledgment calling for the retransmission of specified I frames.

rej_tx

Contains the number of REJ (Reject) frames transmitted by X.25 circuit **<cct_name>**. A REJ frame is a negative acknowledgment calling for the retransmission of specified I frames.

rnr_rx

Contains the number of RNR (Receiver Not Ready) frames received by X.25 circuit **<cct_name>**. An RNR frame denotes a busy condition, indicating a temporary inability to accept I frames from the remote end of the circuit.

rnr_tx

Contains the number of RNR (Receiver Not Ready) frames transmitted by X.25 circuit **<cct_name>**. An RNR frame denotes a busy condition, indicating a temporary inability to accept I frames from the remote end of the circuit.

rr_rx

Contains the number of RR (Receiver Ready) frames received by X.25 circuit **<cct_name>**. An RR frame either indicates the readiness to receive I frames, or acknowledges the receipt of I frames.

rr_tx

Contains the number of RR (Receiver Ready) frames transmitted by X.25 circuit **<cct_name>**. An RR frame either indicates the readiness to receive I frames, or acknowledges the receipt of I frames.

sabm_rx

Contains the number of SABM (Set Asynchronous Balanced Mode) frames received by X.25 circuit **<cct_name>**. SABM frames initiate the establishment of the DCE/DTE link.

sabm_tx

Contains the number of SABM (Set Asynchronous Balanced Mode) frames transmitted by X.25 circuit **<cct_name>**. SABM frames initiate the establishment of the DCE/DTE link.

ua_rx

Contains the number of UA (Unnumbered Acknowledgment) frames received by X.25 circuit **<cct_name>**. UA frames acknowledge receipt of SABM or DISC frames, and complete the establishment or termination of the DCE/DTE link.

ua_tx

Contains the number of UA (Unnumbered Acknowledgment) frames transmitted by X.25 circuit **<cct_name>**. UA frames acknowledge receipt of SABM or DISC frames, and complete the establishment or termination of the DCE/DTE link.

unknown_rx

Contains the number of FRMR (Frame Reject) frames received by X.25 circuit **<cct_name>** that contained a W bit set to 1. This bit setting indicates that the remote end has rejected a frame issued by **<cct_name>** because the frame contained an invalid, or unimplemented, command or response.

unknown_tx

Contains the number of FRMR (Frame Reject) frames transmitted by X.25 circuit **<cct_name>** that contained a W bit set to 1. This bit setting indicates that **<cct_name>** has rejected a previously-received frame because the frame contained an invalid, or unimplemented, command or response.

13.16.2 X.25 Circuit Packet-Level Variables

The ***x25 circuit packet-level*** variables, in alphabetical order, are as follows:

call_cfm_rx

Contains the number of CALL CONNECTED packets received by X.25 circuit **<cct_name>**. A CALL CONNECTED packet completes the call-setup procedure.

call_cfm_tx

Contains the number of CALL ACCEPTED packets transmitted by X.25 circuit **<cct_name>**. A CALL ACCEPTED packet indicates readiness to accept an incoming call, and generates a CALL CONNECTED packet at the remote end of the circuit.

call_rx

Contains the number of INCOMING CALL packets received by X.25 circuit **<cct_name>**. An INCOMING CALL packet indicates that a remote system is attempting to establish a connection.

call_tx

Contains the number of CALL REQUEST packets transmitted by X.25 circuit **<cct_name>**. A CALL REQUEST packet initiates the call-setup procedure, and generates an INCOMING CALL packet at the remote end of the circuit.

clear_cfm_rx

Contains the number of CLEAR CONFIRMATION packets received by X.25 circuit **<cct_name>**. A CLEAR CONFIRMATION packet acknowledges that the previously requested clear action has been implemented.

clear_cfm_tx

Contains the number of CLEAR CONFIRMATION packets transmitted by X.25 circuit **<cct_name>**. A CLEAR CONFIRMATION packet acknowledges that the previously-requested clear action has been implemented.

clear_rx

Contains the number of CLEAR INDICATION packets received by X.25 circuit **<cct_name>**. A CLEAR INDICATION packet acknowledges the receipt of a CLEAR REQUEST packet.

clear_tx

Contains the number of CLEAR REQUEST packets transmitted by X.25 circuit **<cct_name>**. A CLEAR REQUEST packet disconnects the virtual circuit.

diagnostic_rx

Contains the number of DIAGNOSTIC packets received by X.25 circuit **<cct_name>**. The DCE generates DIAGNOSTIC packets for fault isolation purposes.

data_rx

Contains the number of DATA packets received by X.25 circuit **<cct_name>**. DATA packets contain user data.

data_tx

Contains the number of DATA packets transmitted by X.25 circuit **<cct_name>**. DATA packets contain user data.

dropped_tx

Contains the count of IP datagrams dropped by the circuit because of X.25 failures or queue clipping.

error_rx

Contains the number of erroneous packets (for example, packets with a bad length, REGISTRATION packets if registration is not enabled, and RESTART packets not directed to LCN0) received by X.25 circuit **<cct_name>**.

reset_cfm_rx

Contains the number of RESET CONFIRMATION packets received by X.25 circuit **<cct_name>**. A RESET

CONFIRMATION packet acknowledges that the previously requested reset action has been implemented.

reset_cfm_tx

Contains the number of RESET CONFIRMATION packets transmitted by X.25 circuit **<cct_name>**. A RESET CONFIRMATION packet acknowledges that the previously requested reset action has been implemented.

reset_rx

Contains the number of RESET INDICATION packets received by X.25 circuit **<cct_name>**. A RESET INDICATION packet informs the recipient that the remote system has reset the send and receive packet sequences to 0.

reset_tx

Contains the number of RESET REQUEST packets transmitted by X.25 circuit **<cct_name>**. A RESET REQUEST packet sets the send and receive packet sequences to 0, and generates a RESET INDICATION packet at the remote end of the circuit.

restart_cfm_rx

Contains the number of RESTART CONFIRMATION packets received by X.25 circuit **<cct_name>**. A RESTART CONFIRMATION packet acknowledges that the previously requested restart action has been implemented.

restart_cfm_tx

Contains the number of RESTART CONFIRMATION packets transmitted by X.25 circuit **<cct_name>**. A RESTART CONFIRMATION packet acknowledges that the previously requested restart action has been implemented.

restart_rx

Contains the number of RESTART INDICATION packets received by X.25 circuit **<cct_name>**. A RESTART INDICATION packet informs the recipient that the remote system has cleared all switched virtual circuits.

restart_tx

Contains the number of RESTART REQUEST packets transmitted by X.25 circuit **<cct_name>**. A RESTART REQUEST packet clears all switched virtual circuits, and generates a RESTART INDICATION packet at the remote end of the circuit.

rnr_rx

Contains the number of RNR (Receiver Not Ready) packets received by X.25 circuit **<cct_name>**. An RNR packet denotes a busy condition, indicating a temporary inability to accept DATA packets from the remote end of the circuit.

rnr_tx

Contains the number of RNR (Receiver Not Ready) packets transmitted by X.25 circuit **<cct_name>**. An RNR packet denotes a busy condition, indicating a temporary inability to accept DATA packets from the remote end of the circuit.

rr_rx

Contains the number of RR (Receiver Ready) packets received by X.25 circuit **<cct_name>**. An RR packet either indicates the readiness to receive DATA packets, or acknowledges the receipt of DATA packets.

rr_tx

Contains the number of RR (Receiver Ready) packets transmitted by X.25 circuit **<cct_name>**. An RR packet either indicates the readiness to receive DATA packets, or acknowledges the receipt of DATA packets.

t20_tmout

Contains the number of T20 timer expirations. The T20 timer starts when a RESTART REQUEST packet is issued, and terminates when a RESTART CONFIRMATION packet is received. If a RESTART CONFIRMATION is not received within T20 seconds (typically 180 seconds), the RESTART REQUEST is reissued.

t21_tmout

Contains the number of T21 timer expirations. The T21 timer starts when a CALL REQUEST packet is issued, and terminates when a CALL CONNECTED, CLEAR INDICATION, or INCOMING CALL packet is received. If such a packet is not received within T21 seconds (typically 200 seconds), a CLEAR REQUEST is issued.

t22_tmout

Contains the number of T22 timer expirations. The T22 timer starts when a RESET REQUEST packet is issued, and terminates when a RESET CONFIRMATION or RESET INDICATION packet is received. If such a packet is not received within T22 seconds (typically 180 seconds), the RESET REQUEST is reissued.

t23_tmout

Contains the number of T23 timer expirations. The T23 timer starts when a CLEAR REQUEST packet is issued, and terminates when a CLEAR CONFIRMATION or CLEAR INDICATION packet is received. If such a packet is not received within T23 seconds (typically 180 seconds), the CLEAR REQUEST is reissued.

13.16.3 X.25 Point-to-Point Virtual Circuit Variables

The ***x25 dedicated point-to-point virtual-circuit packet-level*** variables, in alphabetical order, are as follows:

data_rx

Contains the number of DATA packets received by X.25 virtual circuit **<svc_name>**. DATA packets contain user data.

data_tx

Contains the number of DATA packets transmitted by X.25 virtual circuit **<svc_name>**. DATA packets contain user data.

dropped_tx

Contains the number of packets dropped by X.25 virtual circuit **<svc_name>** because of resource limitations (no buffer space available, or the virtual circuit was down).

reset_cfm_rx

Contains the number of RESET CONFIRMATION packets received by X.25 virtual circuit **<svc_name>**. A RESET CONFIRMATION packet acknowledges that the previously requested-reset action has been implemented.

reset_cfm_tx

Contains the number of RESET CONFIRMATION packets transmitted by X.25 virtual circuit **<svc_name>**. A RESET CONFIRMATION packet acknowledges that the previously-requested reset action has been implemented.

reset_rx

Contains the number of RESET INDICATION packets received by X.25 virtual circuit **<svc_name>**. A RESET INDICATION packet informs the recipient that the remote system has reset the send and receive packet sequences to 0.

reset_tx

Contains the number of RESET REQUEST packets transmitted by X.25 virtual circuit **<svc_name>**. A RESET REQUEST packet sets the send and receive packet sequences to 0, and generates a RESET INDICATION packet at the remote end of the circuit.

rnr_rx

Contains the number of RNR (Receiver Not Ready) packets received by X.25 virtual circuit **<svc_name>**. An RNR packet denotes a busy condition, indicating a temporary inability to accept DATA packets from the remote end of the circuit.

rnr_tx

Contains the number of RNR (Receiver Not Ready) packets transmitted by X.25 virtual circuit **<svc_name>**. An RNR packet denotes a busy condition, indicating a temporary inability to accept DATA packets from the remote end of the circuit.

rr_rx

Contains the number of RR (Receiver Ready) packets received by X.25 virtual circuit **<svc_name>**. An RR packet either indicates the readiness to receive DATA packets, or acknowledges the receipt of DATA packets.

5rr_tx

Contains the number of RR (Receiver Ready) packets transmitted by X.25 virtual circuit **<svc_name>**. An RR packet either indicates the readiness to receive DATA packets, or acknowledges the receipt of DATA packets.

13.16.4 X.25 DDN and PDN Virtual Circuit Variables

X.25 DDN and PDN service maintain a set of variables identical to that described in A.28.3. Such DDN or PDN virtual circuit variables, however, are ephemeral. These variables can be accessed only for currently-established DDN or PDN virtual circuits.

You access the event log to identify currently-established virtual circuits. Established calls are indicated by event log entries which take the following format:

call: <cct_name>.ip_addr.#

where:

cct_name

Is the name of the X.25 DDN or PDN circuit.

ip_addr

Is the dotted-decimal IP address of the remote device.

#

Is the logical connection number.

After verifying that a call has been established, scan the log to ensure that the call (and switched virtual circuit) is still active (has not been cleared). Cleared calls are indicated by an event log entry which takes the following format:

clr: <cct_name>.ip_addr.# (C=nn) (D=nn)

where:

cct_name

Is the name of the X.25 DDN or PDN circuit.

ip_addr

Is the dotted-decimal IP address of the remote device.

#

Is the logical connection number.

(C=nn)

Is the encoded clearing cause.

(D=nn)

Is encoded diagnostic information.

To display all virtual circuit variables for an established DDN or PDN virtual circuit, enter the following at the NCL prompt:

x25.cct_name.ip_addr.#.*ø

where:

x25

Is the X.25 managed object.

cct_name.ip_addr.#

Is taken from the event log.

****

Is the wildcard character.

To display a single virtual circuit variable for an established DDN or PDN virtual circuit, enter the following at the NCL prompt:

x25.cct_name.ip_addr.#.codeø

where:

x25

Is the X.25 managed object.

cct_name.ip_addr.#

Is taken from the event log.

code

Is the pathname code for the variable.

Index

WELLFLEET communications

15 Crosby Drive, Bedford, MA 01730-2204 Tel: (617) 275-2400 Fax: (617) 275-8421