DECbrouter 90 Products

Configuration and Reference Volume 1

Order Number: EK–DECB1–CG. A01

First Edition, May 1993

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

3 Using Terminals

4 Configuring the System

5 Using MOP

6 Managing and Monitoring the System

7 Configuring the Interfaces

8 Adjusting Interface Characteristics

9 Configuring Packet-Switched Software

Index

Figures

Tables

About This Guide

This section discusses the objectives, audience, organization, and conventions of this guide. It also lists related Digital publications.

Audience

This publication is intended for system administrators who will configure and maintain a DECbrouter 90.

Organization

The following chapters are contained in this guide:

- **Chapter 1, Router Product Overview**–contains an overview of the DECbrouter 90, including router capabilities, network protocols supported by the DECbrouter 90, routing protocols supported by the DECbrouter 90, and the media supported by the DECbrouter 90.
- **Chapter 2, First-Time Startup and Basic Configuration**–describes basic system startup procedures and use.
- **Chapter 3, Using Terminals**–describes use of physical and virtual terminals on the DECbrouter 90 product including, making Telnet server connections from a console attached to the router/bridge, making local changes to the terminal parameters, and using the Digital MOP terminal server.
- **Chapter 4, Configuring the System**–describes how to configure the system including, setting global system characteristics, defining the size of the system buffers, changing the system boot file specifications, storing and booting system software images with flash memory, establishing system and line passwords and system secruity, defining networking services, enabling and directing the logging of system debugging messages, and configuring the console and virtual terminal lines.
- **Chapter 5, Using MOP**–describes how to use the maintenance operating protocol (MOP).
- **Chapter 6, Managing and Monitoring the System**–describes the EXEC commands you use to monitor, manage, and troubleshoot general system processes and conditions on your DECbrouter 90.
- **Chapter 7, Configuring the Interfaces**–describes how to enable, shut down, and display messages pertinent to interface configuration and operation. This chapter also describes the procedures for configuring and maintaining the serial and Ethernet interfaces.

- **Chapter 8, Adjusting Interface Characteristics**–describes how to make adjustments to the router interfaces and how to enable loopback tests.
- **Chapter 9, Configuring Packet-Switched Software**–describes the configuration tasks for packet-switched software, including Recommendation X.25, CMNS support, Frame relay service, and SMDS.

Conventions

The following conventions are used in this guide:

Convention	Meaning
`system displays`	Terminal sessions and information the system displays are printed in monospace type.
boldface	Information you enter is in boldface type. Keywords are also in boldface type.
>	Nonprinting characters are shown in angle brackets.
[ ]	Defaults are in square brackets.
COMMANDS	Commands are in uppercase letters.
italics	Command variables, worksheet values, new terms and concepts, and titles of books and periodicals are in italics.
Ctrl/X	Indicates two keys that you must press simultaneously.
Note	Provides general information about the current topic.
Caution	Provides information to prevent damage to equipment.

Related Documentation

The most important companion to this guide is the set of Digital documentation that accompanies the product. This guide refers you to the appropriate manual for additional reference material that is beyond the scope of this guide. The following manuals are shipped with the DECbrouter 90:

- *DECbrouter 90 Products Getting Started*
- *DECbrouter 90 Products Configuration and Reference, Volume 2*
- *DECbrouter 90 Products Configuration and Reference, Volume 3*
- *DECbrouter 90T Installation and Operating Information*
- *DECbrouter 90 System Error Messages*
- *DECbrouter 90 Products Command Summary*

To order these publications or additional copies of this guide, contact your sales representative. Refer to the How to Order Additional Documentation page at the back of this document for addresses and phone numbers.

Service and Support

Call your local representative for service.

Obtaining Additional Information

This section describes how to obtain additional Digital publications and includes tips for obtaining books, standards, and other information about networks and data communications that might be helpful while using Digital products.

Ordering Additional Digital Publications

To order these publications or additional copies of the *DECbrouter 90 Products Configuration and Reference* guides, contact your local representative.

Obtaining Information

This section describes how to obtain RFCs and technical standards.

Obtaining RFCs

Information about the Internet suite of protocols is contained in documents called *Requests for Comments or RFCs*. These documents are maintained by Government Systems, Inc. (GSI). You can request copies by contacting GSI directly, or you can use the TCP/IP File Transfer Protocol (FTP) to obtain an electronic copy.

Contacting GSI

You can contact GSI through mail, by telephone, or through electronic mail:

Government Systems, Incorporated
Attn: Network Information Center
14200 Park Meadow Drive, Suite 200
Chantilly, Virginia 22021

1-800-365-3642
(703) 802-4535
(703) 802-8376 (FAX)

NIC@NIC.DDN.MIL

Network address: 192.112.36.5
Root domain server: 192.112.36.4

Obtaining an Electronic Copy

To obtain an electronic copy of an RFC through FTP, complete the following steps:

1. At your server prompt, use the **ftp** command to connect to address *nic.ddn.mil*:

   ```
   % ftp nic.ddn.mil
   ```

The following display appears, followed by a login prompt:

```
Connected to nic.ddn.mil.
220-*****Welcome to the Network Information Center*****
    *****Login with username "anonymous" and password "guest"
    *****You may change directories to the following:
      ddn-news          - DDN Management Bulletins
      domain            - Root Domain Zone Files
      ien               - Internet Engineering Notes
      iesg              - IETF Steering Group
      ietf              - Internet Engineering Task Force
      internet-drafts   - Internet Drafts
      netinfo           - NIC Information Files
      netprog           - Guest Software (ex. whois.c)
      protocols         - TCP-IP & OSI Documents
      rfc               - RFC Repository
      scc               - DDN Security Bulletins
220 And more.
```

2. At the login prompt, enter the word **anonymous** as your login name:

   ```
   Name (nic.ddn.mil:cindy): anonymous
   ```

   The NIC responds with this message:

   ```
   331 Guest login ok, send "guest" as password.
   Password:
   ```

3. Enter the word **guest** at the Password: prompt. The following message and ftp> prompt appear:

   ```
   230 Guest login ok, access restrictions apply.
   ftp>
   ```

4. Use the **cd** command to change directories. The following example illustrates how to change the RFC directory and obtain RFC 1158:

   ```
   ftp> cd rfc
   250 CWD command successful.
   ftp> get rfc1158.txt
   ```

5. To exit the FTP facility, enter the **quit** command at the ftp> prompt.

Obtaining Technical Standards

Following are additional sources for technical standards:

- Omnicom, 1-800-OMNICOM
- Global Engineering Documents, 2805 McGraw Ave., Irvine, CA 92714 1-800-854-7179
- American National Standards Institute, 1430 Broadway, New York, NY 10018 (212) 642-4932 or (212) 302-1286

1

Router Product Overview

This chapter provides a product overview of the DECbrouter 90, including:

- Router capabilities
- Network protocols supported by the DECbrouter 90
- Routing protocols supported by the DECbrouter 90
- Media supported by the DECbrouter 90

Capabilities of the Router

Complex internetworks have grown past the point where they can depend on equipment from a single vendor. Virtually all organizations connecting local area networks and creating wide area networks today have major commitments to hardware and software from many different vendors. Therefore, current and future internetworking requires products that support multiprotocol, multimedia, and multivendor networks.

The DECbrouter 90 helps LANs and WANs achieve interoperability and connectivity. This section describes Digital-supported protocols and media, as well as the capabilities of the DECbrouter 90 for routing, network management, and network security.

Support for Multiple Network Protocols

Large organizations need the flexibility that multiprotocol networks give them to communicate with diverse hardware and software from many vendors. The DECbrouter 90 supports many networking protocols, as well as several specific routing protocols for compatibility with other networks. Included are protocols based on open standards and proprietary protocols from a variety of vendors.

The DECbrouter 90 can forward packets concurrently from any combination of the following networking protocols:

- TCP/IP
- DECnet Phase IV
- Xerox Network Service (XNS)
- Novell IPX
- Ungermann-Bass
- AppleTalk (Phase 1 and Phase 2)
- ISO Connectionless Network Services (CLNS) and Connection Mode Network Services (CMNS)
- Apollo Domain
- Banyan VINES

- Xerox's Universal Protocol (PUP)
- CHAOSnet
- X.25 protocols
- DDN protocols, as used with the DDN X.25 Standard, 1822-LH/DH, and HDH (1822-J) attachments
- Frame Relay serial encapsulation
- Switched Multimegabit Data Service (SMDS) protocol
- Point-to-Point Protocol (PPP)

Dynamic Network Routing

The Interior Gateway Routing Protocol (IGRP), developed by Cisco Systems for TCP/IP and ISO CLNS (referred to as ISO-IGRP), monitors the network to determine the status of each route and selects the best route for each data packet. Network traffic, path reliability, and path speed all influence route selection.

While running IGRP or ISO-IGRP, the DECbrouter 90 can concurrently receive and understand messages from other network segments sent using different routing protocols. For example, IP routing protocols supported by the DECbrouter 90, in addition to IGRP, include:

- Open Shortest Path First (OSPF)—the link state-based interior gateway protocol defined by RFC 1247.
- Routing Information Protocol (RIP)—the interior routing protocol used by the routing process on Berkeley-derived UNIX systems.
- Exterior Gateway Protocol (EGP)—the routing protocol used by all routers attached to the Defense Data Network (DDN). This implementation maintains contact with multiple EGP-speaking routers, preserving routing information when the DDN core routers do not respond.
- Border Gateway Protocol (BGP) is a replacement protocol for EGP. BGP is defined by RFC 1163.

Similarly, the DECbrouter 90 supports concurrent operation of ISO-IGRP and the ISO-standard intermediate system-to-intermediate system (IS-IS).

The DECbrouter 90 router supports the native dynamic routing protocols used by the various supported network protocols, such as DECnet, Novell IPX, and AppleTalk. This allows compatibility with other vendor's routers. Multiple network protocols and their dynamic routing protocols operate concurrently, sharing the same router and media.

Support for Standard Media

For convenient access to existing networks, The DECbrouter 90 supports these industry-standard networking media:

- Ethernet—IEEE 802.3 and Type II
- Synchronous serial—V.10, V.11, V.28, V.35, (V.24)RS-232, RS-449, X.21, EIA-422, EIA-423, and EIA-530A.

2

First-Time Startup and Basic Configuration

This chapter describes basic system startup and use. Information in this chapter will help you with these tasks:

- First-time startup and basic configuration of the DECbrouter 90 using the SETUP command facility
- Understanding and using the system's command interpreter
- Entering configuration commands into a configuration file and saving the configuration
- Using TFTP to load configuration files and system software images over the network
- Using the flash memory to boot and store system software images
- Reloading the operating system

Using the Setup Facility for Basic Configuration

The SETUP command facility enables you to start using your network quickly and without extensive background knowledge. It does this by prompting you for the information required to perform basic configuration procedures.

Use the SETUP command facility both at initial system configuration and for basic changes at any time. In addition, use the facility as a teaching tool to become familiar with the expected command sequence as you step through the process. Because of these additional characteristics, this manual refers to SETUP as a command facility rather than simply as a command.

Refer to the *DECbrouter 90 Products Getting Started* publication for detailed information and a step-by-step description of the configuration procedure using the SETUP command facility.

Capabilities of the SETUP Command Facility

Use the SETUP command facility to do the following:

- Establish host names
- Set enable (or privileged mode) passwords
- Set virtual terminal passwords
- Enable SNMP network management
- Enable routing of protocols
- Enable transparent Ethernet bridging

Configure the following protocols with the SETUP command facility:

- IP, including IGRP and RIP dynamic routing
- DECnet (Phase IV)
- XNS
- Novell IPX
- Appletalk Phase 1 and Phase 2
- CLNS
- VINES

For more advanced applications, you need to enter a privileged configuration session, as described in the section Entering Configuration Mode later in this chapter.

Using the SETUP Command Facility

The SETUP command facility operates automatically the first time you power on your router and when you add new hardware components. To use SETUP on subsequent occasions, you must invoke it as you would any other command by entering SETUP at the EXEC prompt (described in the section Entering Configuration Mode later in this chapter).

Before you start using the SETUP command facility, you need to do the following:

1. Attach an RS-232 ASCII terminal to the system console port located at the bottom, front of the router.

   Refer to the appropriate hardware installation and maintenance publications for details about cabling considerations and establishing electrical connections.

2. Configure the terminal to operate at 9600 baud, 8 data bits, no parity, 1 stop bit.

3. Power on the router and run the *setup program*.

Note

Network connections are not required in order to effectively use the SETUP command facility.

In addition, you need to know the following before you start:

- Which protocols you plan to route.

  Note that most protocols will prompt you for specific parameters, including host name, network numbers, addresses, and subnet masks (when applicable).

- Types of interfaces installed: Ethernet or Serial.
- Whether or not you plan to use bridging.

First-Time System Startup

The SETUP command facility determines which interfaces are installed and prompts you for configuration information for each one. Once you complete one interface, the facility automatically starts over for the next, continuing until each interface has been configured.

Note

Once you start the SETUP facility, the system runs through the entire configuration process; you cannot quit out of it. If you want to make a change or correct a mistake, simply press the Return key at each prompt, then restart the command.

When you first power on your console and router, a script similar to the following will appear on the screen. The first section of the script displays the banner information, including the software version:

```
                Restricted Rights Legend

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

  Cisco Systems, Inc.
  1525 O'Brien Drive
  Menlo Park, California  94025

   3000 Software (IGS3-BFPX), Beta Version 9.14(0.17)
   Copyright (c) 1986-1993 by Cisco Systems, Inc.
   Compiled Fri 31-Jul-92 13:31
```

The next portion of the display is a list of the installed hardware. By reading the installed hardware, the system automatically presents the appropriate interfaces during the configuration process.

```
      (68020) processor with 4096K bytes of memory.
      X.25 software.
      Bridging software.
      1 Ethernet/IEEE 802.3 interface.
      1 Serial network interface.
      64 Kbytes of non-volatile memory.
```

The first two sections of the configuration script (the banner and the installed hardware) appear each time the system is started up.

At the first-time system startup, the System Configuration Dialog automatically appears, offering the prompts for which you'll provide the answers to configure your system.

```
         --- System Configuration Dialog ---

   At any point you may enter a question mark '?' for help.
   Refer to the 'Getting Started' Guide for additional help.
   Default settings are in square brackets '[]'. Continue with
   configuration dialog? [yes]:
```

At this point, you may choose not to continue with the System Configuration Dialog and exit by answering "no" to this prompt.

Answer "yes" to continue with the SETUP configuration dialog. The remainder of the script is the actual configuration process, with each prompt appearing automatically. Press the Return key to accept the default settings.

There is no default for the final prompt which asks you if you want to use this configuration; you must answer either "yes" or "no." Also note that the SETUP command only asks you to configure the protocols for each interface that you specified on a global basis. For instance, if you said "no" for XNS under the global parameters, the command does not prompt you to configure that protocol under the interface parameters.

Sample Configuration Session

Configuring global parameters:

```
      Enter host name [Router]: sandbox
      Enter enable password: shovel
      Enter virtual terminal password: pail
      Configure SNMP Network Management? [no]: yes
      Configure XRemote font servers? [no]: yes
         Enter a font server IP address or press RETURN to exit:
121.78.1.1
         Enter a font server IP address or press RETURN to exit:
      Configure IP? [yes]:
         Configure IGRP routing? [yes]:
            Your IGRP autonomous system number [1]:
      Configure DECnet? [no]:  yes
         Your area number [1]:
         Your node number [1]:
         Area (level 2) routing? [no]: yes
      Configure XNS? [no]: yes
      Configure Novell? [no]: yes
      Configure AppleTalk? [no]: yes
         Multizone networks? [no]: yes
      Configure CLNS? [yes]:
         CLNS router tag [area_1]:
         CLNS domain [49]:
         CLNS area [0001]:
         CLNS station id [0000.0C01.0D1D]:
      Configure Vines? [no]: yes
      Configure bridging? [no]: yes
      Configure LAT? [no]: yes
```

```
Configuring interface parameters:

Configuring interface Ethernet0:
   Is this interface in use? [yes]:
  Configure IP on this interface? [yes]:
      IP address for this interface: 131.108.6.67
      Number of bits in subnet field [0]:
      Class B network is 131.108.0.0, 0 subnet bits; mask is 255.255.0.0
   Configure DECnet on this interface? [yes]:
      DECnet cost [10]:
   Configure XNS on this interface? [yes]:
      XNS network number [2]:
   Configure Novell on this interface? [yes]:
      Novell network number [2]:
   Configure AppleTalk on this interface? [yes]:
      Extended AppleTalk network? [no]: yes
      AppleTalk starting cable range [4172]:
      AppleTalk ending cable range [4172]:
      AppleTalk zone name [twilight]:
      AppleTalk zone name [ozone]:
      AppleTalk zone name:
   Configure CLNS on this interface? [yes]:
   Configure Vines on this interface? [yes]:
   Configure bridging on this interface? [yes]:

Configuring interface Serial0:
   Is this interface in use? [yes]:
   Configure IP on this interface? [yes]:
      IP address for this interface: 131.108.97.67
      Number of bits in subnet field [0]:
      Class B network is 131.108.0.0, 0 subnet bits; mask is 255.255.0.0
   Configure DECnet on this interface? [yes]: no
   Configure XNS on this interface? [yes]: no
   Configure Novell on this interface? [yes]: no
   Configure AppleTalk on this interface? [yes]:
      Extended AppleTalk network? [no]:
      AppleTalk network number [1]:
      AppleTalk zone name [twilight]:
   Configure CLNS on this interface? [yes]:
   Configure Vines on this interface? [yes]:
   Configure bridging on this interface? [yes]:
```

```
The following configuration command script was created:

hostname sandbox
enable-password shovel
line vty 0 4
password pail
snmp-server community
xremote tftp host 121.78.1.1
!
ip routing
decnet routing 1.1
decnet node-type area
xns routing
novell routing
appletalk routing
clns routing
router iso-igrp area_1
net  49.0001.0000.0C01.0D1D.00
vines routing
bridge 1 protocol dec
!
!
!
!
interface Ethernet0
ip address 131.108.6.67 255.255.255.0
xns network 2
novell network 2
appletalk cable-range 4172-4172
appletalk discovery
clns router iso-igrp area_1
vines metric
bridge-group 1

!
interface Serial0
no shutdown
no ip address
appletalk address 50000.72
appletalk zone twilight
clns router iso-igrp area_1
vines metric
bridge-group 1
!
!
router igrp 109
network 131.108.0.0
!
end

Use this configuration? [yes/no]: yes
[OK]
Use the enabled mode 'configure' command to modify this configuration.


Press RETURN to get started!
```

The server displays the system name (sandbox), followed by an angle bracket (>), which is the prompt of the system's command interpreter.

Using the EXEC Command Interpreter

The command interpreter is called the EXEC. The EXEC interprets the commands you type and carries out the corresponding operations.

You can type commands when you see the system prompt, which is the system's host name ending with an angle bracket (>). Although the default system host name is *Router*>, this may have been changed during the initial configuration using the SETUP command, or with the HOSTNAME configuration command. The following sections describe how to use the EXEC.

Command Syntax

The EXEC accepts commands typed in uppercase letters, lowercase letters, or both. You may also abbreviate commands and other keywords to the number of characters that cause the command to be a unique abbreviation. For example, you can abbreviate the SHOW command to SH.

If you make a typing mistake, you can erase characters one at a time with the Delete or the Backspace key. Press either key to erase the last character typed. To erase the entire line, enter Ctrl/U. (This notation means "Hold down the Ctrl key and press the U key.") The server acts on most commands after you press the Return key.

You can list available EXEC commands by typing a question mark (?). You also can enter a question mark to obtain more information about commands. For example, enter **TERMINAL ?** to obtain a list of TERMINAL commands or **SHOW ?** to obtain a list of SHOW commands.

Certain EXEC commands produce multiple screens of output. At the end of each screen, the EXEC pauses and displays:

```
-More-
```

Type a space to continue the output; type anything else to return to the system command prompt.

EXEC Command Levels

For security purposes, the EXEC has two levels of access: user and privileged. The commands available at the user level are a subset of the commands available at the privileged level. Because many of the privileged commands set operating parameters, the privileged level should be password-protected to prevent its unauthorized use. The system prompt for the privileged level ends with a pound sign (#) instead of an angle bracket (>).

The EXEC ENABLE command allows access to the privileged level, prompting for a password if one has been set with the ENABLE-PASSWORD configuration command. (For more information, see the section Establishing Passwords and System Security in Chapter 4.)

Enter a question mark **?** to see a list of the user-level EXEC commands similar to the following:

```
Router> ?

connect <host>  Connect to host - same as typing just a host name
disconnect <cn> Break the connection specified by name or number
exit, quit      Exit from the EXEC
name-connection Give a connection a logical name
resume          Make the named connection be current
show <cmd>      Information commands, enter "show ?" for list
systat          Show terminal lines and users
telnet <host>   Connect to host using telnet protocol
terminal        Change terminal's parameters, enter "terminal ?"
where           Show open connections
<cr>            To resume connection
```

Enter **enable** and then the password to access the privileged command level. Enter the question mark **?** to see a list of privileged-level EXEC commands similar to this example:

```
Router# ?

bfe            For manual emergency modes setting, enter "bfe?" for list
clear           Reinitialization functions, enter "clear ?" for list
configure       Configure from terminal or over network
connect <host>  Connect to host - same as typing just a host name
debug           Enable debugging functions, enter "debug ?" for list
disable         Turn off privileged commands
disconnect <cn> Break the connection specified by name or number
enable          Turn on privileged commands
exit, quit      Exit from the EXEC
lat <service>   Connect to service using DEC LAT protocol
name-connection Give a connection a logical name
ping            Send echo messages
reload          Halt and reload system
resume          Make the named connection be current
send <line>|*   Send message to a terminal line or lines
setup           Initialize system configuration
show <cmd>      Information commands, enter "show ?" for list
systat          Show terminal lines and users
telnet <host>   Connect to host using telnet protocol
terminal        Change terminal's parameters, enter "terminal ?"
test            Run hardware tests, enter "test ?"
trace <address> Trace route to <address>
undebug         Disable debugging functions, enter "undebug ?" for list
where           Show open connections
which-route    Displays CLNS route table lookup.
write           Write configuration memory, enter "write ?" for list
<cr>            To resume connection
```

To return to the user-level prompt, enter **disable** at the EXEC prompt.

The EXEC command CONFIGURE begins the configuration mode, where you enter the commands to configure your router for its particular routing or bridging function. The following section describes the use of this command.

Entering Configuration Mode

Use the privileged EXEC command CONFIGURE to begin configuration of the router.

Begin by entering the privileged level of the EXEC. This is done by entering the ENABLE command at the EXEC prompt:

```
Router> enable
```

The EXEC then prompts you for a privileged-level password:

```
Password:
```

Enter the password. For security purposes, the password will not be displayed. (Also note that the password is case-sensitive.) When you enter the correct password, the system displays the privileged-mode system prompt:

```
Router#
```

To begin configuration mode, enter the CONFIGURE command at the privileged-mode prompt:

```
Router# configure
```

When you enter this command, the EXEC prompts you for the source of the configuration subcommands.

```
Configuring from terminal, memory, or network [terminal]?
```

The default is to enter the commands from the terminal console. Pressing the Return key begins this configuration method. Each configuration technique (terminal, memory, and network) is described in more detail later in this chapter.

The EXEC provides you with a simple editor for entering the configuration commands, and explains the editing functions:

```
Enter configuration commands, one per line.
Edit with DELETE, CTRL/W, and CTRL/U; end with CTRL/Z
```

Table 2–1 lists the edit key functions and their meanings.

Table 2–1 Configuration Edit Keys

Key	Meaning
Delete or Backspace	Erases one character.
Ctrl/W	Erases a word.
Ctrl/U	Erases a line.
Ctrl/R	Redisplays a line.
Return	Executes single-line commands.
Ctrl/Z	Ends configuration mode and returns to the EXEC.

Entering the Configuration Commands

The configuration subcommands are categorized by these functions:

- Global configuration commands—Define system-wide parameters.
- Interface subcommands—Define the characteristics of an interface (a serial or Ethernet interface, for example) and must be preceded by an INTERFACE command.

- Line subcommands—Define the characteristics of a serial terminal line and must be preceded by a LINE command.
- Router subcommands—Configure an IP routing protocol and must be preceded by a ROUTER command.

The command descriptions include the command types and give examples of their use.

As with EXEC commands, you can enter configuration subcommands in uppercase letters, lowercase letters, or both. You can also shorten all commands and other keywords to unique abbreviations. You can add comments by preceding the line with an exclamation point (!). Comments do not affect command processing.

The router does not display confirmation messages as it executes the commands. If the router encounters a problem, it displays an error message on the console terminal.

In most cases, you can negate a configuration subcommand or restore a default by entering **no** before the subcommand keyword. You usually can omit the arguments of the subcommand when you negate it with **no**. The command descriptions note any exceptions to these rules.

Examples of Configuration Files

The following examples of configuration files illustrate how to enter the configuration commands.

The EXEC accepts commands in uppercase and lowercase letters. Exclamation points are not parsed and serve as comment lines and delimiters between configuration commands.

Global Configuration Commands

Use global configuration commands to enable functions that affect the entire system rather than a particular line or interface, and can appear any place within the configuration file. An example of this is the global configuration command to define the host name, or the name of the router:

```
hostname router-1
```

Commands to enable a particular routing or bridging function are also global configuration commands. The following example illustrates how to enable the Xerox Network System (XNS) routing protocol:

```
xns routing 0123.4567.abcd
```

Once enabled, interface characteristics for XNS routing are specified using the INTERFACE command and XNS-specific interface subcommands. Command descriptions in the sections describing configuration will define the command type.

Interface Subcommands

Interface subcommands modify the operation of an interface such as an Ethernet or serial port. Interface subcommands always follow an INTERFACE command, which defines the interface type.

The following example illustrates how to enable XNS network 1 on interface Ethernet 0:

```
interface ethernet 0
xns network 1
```

If you forget to enter the INTERFACE command, the system displays the message "must specify a network interface."

Line Subcommands

Line subcommands modify the operation of a serial terminal line. Line subcommands always follow a LINE command, which defines the line number. If you forget to enter the LINE command, the system displays the message "must specify a line or range of lines."

The following example illustrates how to set the password on line 5:

```
line 5
password secretword
```

Router Subcommands

Router subcommands are used to configure IP routing protocol characteristics and always follow a ROUTER command. The following example illustrates how to set the maximum hop metric for the Cisco IGRP routing protocol:

```
router igrp
metric maximum-hops 150
```

If you forget to enter the ROUTER command, the system displays the message "must specify a routing protocol."

Remember to enter Ctrl/Z to end your configuration sessions and to use the DISABLE command to leave the privileged mode.

Creating the Configuration File

If you used the SETUP facility's interactive dialog prompts to start your configuration file, it was saved in nonvolatile memory when you finished the prompts. If you chose not to create your configuration file this way, there are several options you now can choose from to create the configuration file.

The router holds configuration information in two places—in *running memory* and in *nonvolatile memory*. Configuration information in running memory is temporary and will not be stored if power is shut off. Configuration information in nonvolatile memory is always available.

Use the EXEC command WRITE MEMORY to copy current (running) configuration information to nonvolatile memory. This command stores all nondefault configuration information as configuration commands in text format. The command also records a checksum for the information to protect against data corruption.

The EXEC command SHOW CONFIGURATION displays information stored in nonvolatile memory. You can use this command and the WRITE TERMINAL command to find differences between the current configuration (that in running memory) and that stored in nonvolatile memory. Use the EXEC command WRITE ERASE to clear the contents of nonvolatile memory.

WRITE commands create their output by examining the state of the system currently running. The output produced by the WRITE commands is generated by the software, and will not necessarily match the text the user entered to create the current configuration.

The router also allows you to store the configuration file on a network host. (This allows you to use an editor on the host to edit and create the configuration file.) Use the EXEC command WRITE NETWORK to copy the current configuration information to a server host on the network. Use of this command is described later in this section.

Configuring from the Console

To issue configuration commands from the console terminal, enter the EXEC command CONFIGURE at the privileged-level EXEC prompt and enter configuration mode.

The router responds with this prompt asking you to specify the terminal, a file, or nonvolatile memory as the source of configuration commands:

```
Configuring from terminal, memory, or network [terminal]?
```

To begin configuration, enter **terminal** at the prompt or press Return (since terminal is the default) to start command collection. (See the section Entering Configuration Mode in this chapter for more information.)

During command collection, the router accepts one configuration command per line. You can enter as many configuration subcommands as you want.

Press Ctrl/Z when you finish entering configuration commands. This returns you to the EXEC, where you can test your configuration or write the configuration commands to memory.

At periodic intervals, you will want to write the configuration information into nonvolatile memory or to a configuration file stored on a remote host. This will make checking, adding information to, and booting the configuration file an easier task. The procedures for writing information to nonvolatile memory are described next.

Writing the Configuration File to Nonvolatile Memory

After you enter the desired configuration information at the console terminal, use the privileged EXEC command WRITE MEMORY to make a copy of the configuration information in the nonvolatile memory. Nonvolatile memory stores the current configuration information in text format as configuration commands, recording only nondefault settings. The memory is checksummed to guard against corrupted data.

As part of its startup sequence, the router startup software always checks for configuration information in the nonvolatile memory. If the nonvolatile memory holds valid configuration commands, the router executes the commands automatically at startup. If the router detects a problem with the nonvolatile memory or the configuration information it contains, the router may enter the setup mode, prompting for configuration information. Problems can include a bad checksum for the information in the nonvolatile memory and the absence of critical information.

To display the configuration information stored in the nonvolatile memory, enter the SHOW CONFIGURATION EXEC command at the privileged-mode EXEC prompt.

To clear the contents of the nonvolatile memory, enter the WRITE ERASE EXEC command at the privileged level EXEC prompt.

To reexecute the configuration commands stored in nonvolatile memory, enter **memory** at the configure mode prompt:

```
Configuring from terminal, memory, or network [terminal]? memory
```

Writing the Configuration File to a Remote Host

To store configuration information on a remote host, enter the privileged EXEC command WRITE NETWORK. This command sends a copy of the current configuration information to a remote host. The command will prompt you for the destination host's address and a file name, as the following example illustrates.

Example

```
Tokyo# write network
Remote host [131.108.2.155]?
Name of configuration file to write [tokyo-confg]?
Write file tokyo-confg on host 131.108.2.155? [confirm] y
Writing tokyo-confg...
 [OK]
```

To retrieve and/or add to the configuration information stored on a host file on a device on your network, enter **network** at the configure mode prompt (see the section Entering Configuration Mode for more information):

```
Configuring from terminal, memory, or network [terminal]? network
```

The system will ask you to select a host or network configuration file, for the address of the host, and for a file name. The following example illustrates this process.

Example

```
Host or network configuration file [host]?
IP address of remote host [255.255.255.255]? 131.108.2.155
Name of configuration file [tokyo-confg]?
Configure using tokyo-confg from 131.108.2.155? [confirm] y
Booting tokyo-confg from 131.108.2.155: !! [OK - 874/16000 bytes]
```

Setting Up Auto Load of the Configuration File

The router may be configured to automatically load additional configuration information from a network host. You may want to keep an up-to-date version of configuration information on another host, where you can change it as necessary, and use the nonvolatile memory as a bootstrap or backup mechanism. You can instruct the router to load configuration information over the network by entering the SERVICE CONFIG subcommand and then writing the information to nonvolatile memory using the WRITE MEMORY command. Loading configuration information over the network is the default if nonvolatile memory is not installed. (The SERVICE CONFIGURATION subcommand is described in the section Tailoring Use of Network Services in Chapter 4.)

After loading configuration information from the nonvolatile memory, the router will attempt to load two configuration files from remote hosts. The first is the network configuration file, which contains commands that apply to all routers and terminal servers on a network. The second is the host configuration file, which contains commands that apply to one router in particular.

Using TFTP to Load Configuration Files or System Images

The router uses the Trivial File Transfer Protocol (TFTP) to load and save configuration files.

Note

TFTP is defined in RFC 783. The details of setting up a TFTP server process and installing system images or configuration files on the server host vary from one operating system to another. See the documentation for your host computer if you need more information about TFTP support.

Loading Configuration Files

The default name of the network configuration file is network-confg. The default name for the host configuration file is taken from the host name. The host name can be specified by the HOSTNAME configuration subcommand or can be derived from the Domain Name System (DNS); see the section Setting the Host Name in Chapter 4 for more information. To form the host configuration file name, the network server converts the host name to lowercase, stripped of any DNS information, and appends "-confg." If no host name information is available, the default host configuration file name is router-confg. Other names for these configuration files can be set using the BOOT command, which is described in the section Setting Configuration File Specifications in Chapter 4.

service config
no service config

To enable the loading of network configuration files at router reboot time, use the SERVICE CONFIG command. The NO version of this command (the default) disables the loading of these files.

If the router fails to load a configuration file during startup, it tries again every ten minutes (default setting) until a host provides the requested files. With each failed attempt, the router displays a message on the console terminal.

If the router is unable to load the file named *network-confg*, it displays the following message:

```
Booting network-confg... [timed out]
```

Note

Be aware that the system treats network and host configuration files differently when loading new parameters. When a host configuration file is loaded, all terminal line parameters are cleared before setting any new parameters. When a network configuration file is loaded, no old parameters are cleared. This means that terminal line parameters set by the network configuration file, which are generally loaded first, will be reset by the host configuration file, which is generally loaded second.

Using Flash Memory for Storing and Booting System Software

System software images can be written to flash memory for booting. Refer to the section Storing and Booting System Software Using the Flash Memory in Chapter 4 for a description of the flash commands.

Reloading the System

Use the following EXEC command to halt and restart the DECbrouter 90:

reload

If the system is set to restart on error, it reboots itself.

3
Using Terminals

This chapter describes use of physical and virtual terminals on the DECbrouter 90 product. The tasks covered include:

- Making Telnet server connections from a console attached to the router/bridge
- Making local changes to the terminal parameters
- Using the Digital MOP terminal server

This chapter concludes with an alphabetical summary of the commands it describes.

Making and Managing Terminal Connections

A TCP/IP Telnet connection is the basic way to communicate from a terminal to a host on a network. The DECbrouter 90 provides Telnet communication as defined in RFC 854 and the MIL STD 1782 specification.

Making Telnet Connections

To start a Telnet connection, enter a host name or a dotted-decimal Internet address at the EXEC prompt. You may precede the host name or Internet address with the command CONNECT or TELNET. This can be helpful if the host name you want to use conflicts with a router command name.

The router automatically numbers connections for you. Several commands use these numbers to identify connections, and you can display them using the WHERE command described later in this chapter.

If you use a host name, the router must first find the corresponding Internet address. To find this address, the router searches its host-name-to-address cache. If the name is not in the cache, the router uses a dynamic name lookup method. This method enables the router to query a set of server hosts for the address.

As an option, you can specify a decimal TCP port number after the host name or Internet address when starting a Telnet connection. Normally, the router uses the default Telnet server port, port number 23 (decimal).

After the router determines the Internet address, or if you specify the address directly, the router attempts to connect with the Telnet server port at that address. If the connection attempt fails, the router displays a message to that effect and returns to the EXEC interpreter.

If the connection attempt succeeds, you can communicate with the server host as a terminal of that host. When you log off the host, the router returns to the EXEC interpreter.

Example

To connect to a host named router-1, enter that name at the prompt, as seen in this example.

```
Router> router-1
```

This example illustrates how to connect to a router with IP address *103.81.25.2:*

```
Router> connect 103.81.25.2
```

Establishing Multiple Connections

The router provides an escape sequence with which you can leave a Telnet connection without terminating it and return to the EXEC interpreter. This allows you to have any number of concurrent Telnet connections open and to switch back and forth between them. Follow these steps to switch between connections:

1. Enter the escape sequence, which is usually the default key sequence Ctrl/^ X. This sequence is entered by pressing the Ctrl and ^ keys simultaneously, letting go, then pressing the X key.
2. At the system command prompt, enter the command to open another connection.

To make a new connection, use the procedure described in the previous section Making Telnet Connections. To return to an existing connection, enter the RESUME command. Enter the WHERE command to show your open connections.

You can change the first part of the escape sequence with the ESCAPE-CHARACTER command; see the section Setting the Escape Character in Chapter 4.

Listing Connections

Use the following command to get a listing of connections:

where

This command displays information about open connections associated with the current terminal line, as seen in the example that follows.

```
Router> where
Conn Host               Address            Byte  Idle Conn Name
   1 DREGGS             130.106.19.50         0     0 DREGGS
   2 EMBER              130.106.20.33         0     0 EMBER
*  3 CLASH              130.106.21.24         0     0 CLASH
```

The information includes the connection number, host name, address, number of characters waiting to be sent to the terminal, idle time, and connection name. An asterisk (*) indicates the current connection.

Resuming a Previous Connection

Use the EXEC RESUME command to resume a connection. This command has the following syntax:

resume [*connection*]

This command provides three ways to resume a previous connection:

- By entering the RESUME command with a connection number
- By entering only the connection number
- By pressing the Return key to return to the most recent connection.

The WHERE command provides the connection number.

The following examples demonstrate use of the RESUME command.

Examples

This command resumes connection 2:

```
Router> resume 2
```

You also can omit the command name and simply type the connection number to resume that connection. This example resumes connection 3:

```
Router> 3
```

To resume the most recent connection, press the Return key.

Naming a Connection

To name a connection, use the following command:

name-connection

This command assigns a logical name to a connection. The EXEC prompts for the connection number and name to assign when you enter this command. The WHERE command displays a list of the assigned logical connection names.

Exiting a Session

To exit a session, use one of the following commands:

exit
quit

The EXIT and QUIT commands terminate the incoming connection and all outgoing connections from the router. Enter one of these commands when you are finished with all sessions.

Disconnecting

To disconnect from a specified connection, use the following command:

disconnect [*connection*]

The optional argument *connection* is a connection name or number; the default is the current connection.

Do not use the DISCONNECT command to end a session. Instead, log off the host, thus allowing the host to initiate the disconnect. If you cannot log off the host, use the DISCONNECT command.

Resetting a Line

To reset a terminal line, use the following privileged EXEC command:

clear line *line-number*

This command aborts any connections, terminates the associated processes, and resets the data structures associated with a terminal line.

The argument *line-number* specifies the terminal line number.

Executing Special Telnet Commands

The Telnet software supports special Telnet commands in the form of Telnet sequences that map generic terminal control functions to operating system-specific functions.

To issue a special Telnet command, type the escape sequence (usually Ctrl/^) and then a command character. You can type the command character either as you hold down Ctrl or with Ctrl released, and you can type either uppercase or lowercase letters.

Table 3–1 lists the special Telnet commands.

Table 3–1 Special Commands in the Digital Systems Telnet Implementation

Command Name	Command Characters
Break	Ctrl/^, B
Interrupt Process (IP)	Ctrl/^, C
Erase Character (EC)	Ctrl/^, H
Abort Output (AO)	Ctrl/^, O
Are You There? (AYT)	Ctrl/^, T
Erase Line (EL)	Ctrl/^, U

At any time during an active Telnet session, you can list the Telnet commands by typing this command at the system prompt:

Ctrl/^ ?

This is done by entering the escape sequence followed by a question mark. This command displays an online table of the special Telnet commands, for quick reference.

A sample of this table follows (the Ctrl key is represented by the first ^ character).

```
[Special telnet escape help]
^^B  sends telnet BREAK
^^C  sends telnet IP
^^H  sends telnet EC
^^O  sends telnet AO
^^T  sends telnet AYT
^^U  sends telnet EL
```

Incoming Telnet Connections

In addition to the console terminal, each router supports up to five incoming Telnet connections. Each of these connections can start an EXEC interpreter process on the router.

The user of an incoming Telnet connection can gain access to the privileged EXEC commands through the ENABLE command, which requires a password. With access to the complete EXEC command set, the incoming connection acts as a remote console. A remote console connection provides a convenient way to monitor and adjust router operation.

You can control access to the router with access lists; see the DECbrouter 90T Products, Configuration and Reference, Volume 2, Chapter 5, "Routing IP" for more information.

The router supports the following Telnet options:

- Echo
- Binary Transmission
- Suppress Go Ahead
- Terminal Type
- Send Location

LAT EXEC Commands

The commonly used Digital commands that follow are supported on the DECbrouter 90. Use this HELP command to obtain information about LAT EXEC commands:

h

Use this LOGOUT command to exit LAT EXEC mode and free the line:

lo

Displaying TCP Connections

To show the status of a TCP connection, enter this EXEC command:

show tcp [*line-number*]

The SHOW TCP command displays the status of all TCP connections. Specify the optional argument *line-number* in octal to display the status of the TCP connections for a particular line. The following example shows the command output:

```
con0 (console terminal), connection 1 to host MATHOM
Connection state is ESTAB, I/O status: 1, unread input bytes: 1
Local host: 192.31.7.18, 33537  Foreign host: 192.31.7.17, 23
Enqueued packets for retransmit: 0, input: 0, saved: 0
Event Timers (current time is 2043535532):
Timer:       Retrans   TimeWait    AckHold    SendWnd   KeepAlive
Starts:           69          0         69          0           0
Wakeups:           5          0          1          0           0
Next:     2043536089          0          0          0           0
iss: 2043207208 snduna: 2043211083  sndnxt: 2043211483    sndwnd: 1344
irs: 3447586816 rcvnxt: 3447586900  rcvwnd:       2144 delrcvwnd:   83
RTTO: 565 ms, RTV: 233 ms, KRTT: 0 ms, minRTT: 68 ms, maxRTT: 1900 ms
ACK hold: 282 ms
Datagrams (max data segment is 536 bytes):
Rcvd: 106 (out of order: 0), with data: 71, total data bytes: 83
Sent: 96 (retransmit: 5), with data: 92, total data bytes: 4678
```

Displaying Active Sessions

To show the current active sessions, use the following command:

show sessions

The SHOW SESSIONS command provides information about open Telnet connections. This command can be run at the user-level prompt.

Displaying Information About Active Lines

The SHOW USERS and SYSTAT EXEC commands display information about the active lines of the router, including the line number, connection names, and terminal location.

show users [**all**]
systat [**all**]

Specify the optional keyword **all** to display information for both active and inactive lines. These commands enable monitoring of virtual terminal use. You can issue these commands at the user-level prompt. They are synonymous.

Changing Terminal Parameters

The following sections describe how to change the terminal parameters using the TERMINAL commands. The new settings temporarily override those made with the line configuration subcommands described in Chapter 4.

To obtain information about the terminal configuration parameter settings for the current terminal line, use the SHOW TERMINAL command. To display information about the active ports of the server, use the SHOW USERS command. The information displayed includes the line number, connection name, idle time, and terminal location.

Some TERMINAL commands use the decimal representation of an ASCII character as an argument. See the *DECbrouter 90 Configuration and Reference, Volume 3*, Appendix E, "ASCII Character Set," for ASCII-to-decimal conversion information.

To display a list of commands that you can enter to change the hardware and software parameters of the current terminal line, use the command:

terminal ?

Each command has a **no** variation that undoes the local setting.

Changing the Terminal Screen Width

To set or unset the number of characters (columns) on a single line of the current terminal screen, use the terminal width command:

terminal width *columns*
terminal no width

The login protocol uses the argument columns to set up terminal parameters on a remote host.

Example

This example sets the terminal width to 132 columns.

```
Router> terminal width 132
```

Changing the Terminal Screen Length

To set or unset the number of lines on the screen of the current terminal, use the TERMINAL LENGTH command:

terminal length *screen-length*
terminal no length

The argument *screen-length* is the desired number of lines.

The server uses this value to determine when to pause during multiple-screen output. The default length is 24 lines. A value of zero disables pausing between screens of output. The screen length specified can be learned by hosts.

Changing the Terminal Escape Character

To set or remove the escape character for the current terminal line, use the TERMINAL ESCAPE-CHARACTER command:

terminal escape-character *decimal-number*
terminal no escape-character

The argument *decimal-number* is the ASCII decimal representation of the desired escape character or an escape sequence (Ctrl/P, for example). Typing the escape character followed by the X key returns you to the EXEC when you are connected to another computer. The default escape character is Ctrl/^ X. (See *DECbrouter*

90 Configuration and Reference, Volume 3, Appendix E, "ASCII Character Set," for a list of ASCII characters.)

The operating software interprets the Break key on the console as an attempt to halt the system.

Note

Depending upon the configuration register setting, console breaks either will be ignored or will cause the server to shut down. The Break key cannot be used as the escape character on the router.

Example

This example sets Ctrl/P as the escape character:

```
Router> terminal escape-character 17
```

Displaying the Debug Messages on the Console and Terminals

To display the debug message on the console and terminals, use the TERMINAL MONITOR command:

terminal monitor
terminal no monitor

The TERMINAL MONITOR command copies DEBUG command output and system error messages to the current terminal as well as to the console terminal.

To use this command, you must first enter the ENABLE command and then the password to access the privileged command mode.

Establishing Input Notification

To establish input notification, use the TERMINAL NOTIFY command:

terminal notify
terminal no notify

When you have multiple concurrent connections, you may want to know when output is pending on a connection other than the current connection. For example, you may want to know when another connection receives mail or a message. The TERMINAL NOTIFY command causes the router to notify you of pending output. The TERMINAL NO NOTIFY command ends such notifications.

Changing the Character Padding

To set the character padding on the current terminal line, use the TERMINAL PADDING commands:

terminal padding *decimal-number count*
terminal no padding *decimal-number*

The argument *decimal-number* is the ASCII decimal representation of the character. The argument count is the number of NULL bytes sent after that character. (The *DECbrouter 90 Configuration and Reference, Volume 3*, Appendix E, "ASCII Character Set," provides a list of the ASCII characters.)

The TERMINAL NO PADDING command ends this padding.

Example

This example pads RETURNS (ASCII character 25) with 20 NULL bytes:

```
Router> terminal padding 25 20
```

Displaying Terminal Parameter Settings

To display the configuration parameter settings for the current terminal, use this EXEC command:

show terminal

This command can be issued at the user-level prompt.

Setting Widths for International Character Sets at the User Level

The number of significant bits for characters passed in EXEC and configuration modes, and for special characters, can be set. Changing these widths allows the use of international character sets.

These settings can be configured globally, by interface, and locally at the user level. See Chapter 4 for global and interface configuration information.

Use the following command to change the ASCII character widths for characters entered at the EXEC and during configuration mode:

terminal exec-character-bits {**8** | **7**}

This EXEC command overrides the default-value EXEC-CHARACTER-BITS global configuration command. Configuring the EXEC character width to 8 bits allows you to add special graphical and international characters in banners and prompts. When the user exits the system, the character width is reset to the default value established by the global configuration command. See Chapter 4 for more information.

Note

Setting the EXEC character width to eight bits can cause failures. If a user on a terminal that is sending parity enters the command HELP, an "unrecognized command" message appears because the system is reading all eight bits, although the eighth bit is not needed for the HELP command.

Example

This example temporarily configures a router to use a full 8-bit user interface for system banners and prompts. This allows the use of additional graphical and international characters.

```
gw> terminal exec-character-bits 8
```

Use the following command to temporarily change the ASCII character widths to accept special characters typed in during data connection:

terminal special-character-bits {**8** | **7**}

The TERMINAL SPECIAL-CHARACTER-BITS command temporarily allows the server to support international character sets. It overrides the default-value special-character-bits global configuration command and is used to compare character sets typed by the user with the special character available during a data connection. Special characters include software flow control and escape characters. Configuring the width to 8 allows you to use twice as many special characters as with the 7-bit setting. When the user exits the system, the command is reset to the default value established by the global configuration command. See Chapter 4 for more information.

Example

This example temporarily configures full 8-bit comparisons of flow control and interrupt characters to allow more special characters to be accepted. When you exit the system, character width will be reset to the width established by the global configuration command.

```
gw> terminal special-character-bits 8
```

Using the Digital MOP Server

All Digital internetworking products implement Digital maintenance operation protocol (MOP) for Ethernet interfaces. The MOP server supports the request ID message, periodic system ID messages, and the remote console carrier functions.

The MOP server periodically multicasts a system ID message, which is used by the DECbrouter 90 Ethernet configurator to determine what stations are present in an Ethernet network. The configurator is controlled by the network control program (NCP) command DEFINE MODULE CONFIGURATOR. For more information on this command, consult DECnet–VAX documentation.

The MOP server supports the Digital remote console function. In this capacity, a system manager on a DECnet system that does not include TCP/IP can create a virtual terminal connection to a DECbrouter 90. The NCP commands CONNECT NODE and CONNECT VIA are used to connect to the remote console. Due to the nature of the MOP server, only a single inbound connection per Ethernet interface is supported. The MOP server does not contain the mechanisms necessary to support more than one connection at a time.

MOP is not a routable protocol. To bridge the MOP console carrier and system ID functions, enable bridging for protocol type 6002. The periodic system ID messages are sent to the multicast address *AB00.0002.0000*.

The EXEC command DEBUG MOP reports events occurring within the MOP server, including reception of request ID messages, transmission of system ID messages, and reservation and release of the remote console.

EXEC Terminal Command Summary

This section alphabetically lists the EXEC commands described in this chapter.

clear line *line-number*

Aborts connections and processes and resets the line. Argument line-number specifies the line.

{**connect** | **telnet**} [*connection*]

Either of the commands connects to a remote host using the Telnet protocol. The optional argument connection specifies a host name or an IP address.

Ctrl/^ ?

Lists the special Telnet commands in the form of Telnet sequences that map generic terminal control functions to operating system-specific functions. Enter this command at the system prompt.

disconnect [*connection*]

Closes the specified connection. The optional argument connection is a connection name or number; the default is the current connection.

exit
quit

Either of these commands terminates the EXEC command processor and closes any active Telnet sessions.

name-connection

Assigns a logical name to a connection. The EXEC prompts for the connection number and name to assign when you enter this command.

resume [*connection*]

Resumes a connection. The optional argument connection is a connection name or number.

show sessions

Provides information about open Telnet connections.

show tcp[*line-number*]

Displays the status of all TCP connections. Specify the optional argument line-number in octal to display the status of the TCP connections for a particular line.

show terminal

Displays information about the terminal configuration parameter settings for the current terminal line and the active ports of the server. The optional **all** keyword requests information for both active and inactive ports.

show users [**all**]
systat [**all**]

Displays information about active lines. The optional **all** keyword provides information about inactive as well as active ports.

terminal ?

Lists commands you can enter to change hardware and software parameters of the current line.

terminal [**no**] **escape-character** *decimal-number*

Sets the escape character for the current terminal line. The argument decimal-number is either the ASCII decimal representation of the desired escape character or an escape sequence. The default escape character is Ctrl/^ X. The **no** version of the command removes the specified escape character for the current terminal line.

terminal exec-character-bits {**8** | **7**}

Changes the ASCII character widths for characters entered at the EXEC and during configuration mode.

terminal [**no**] **length** *screen-length*

Sets the length of the terminal to screen-length lines. The default length is 24 lines. The no version of the command unsets the number of lines.

terminal [**no**] **monitor**

Displays the debug message on the console and terminals. It copies DEBUG command output and system error messages to the current terminal as well as to the console terminal. To use this command, you must first enter the ENABLE command and then the password to access the privileged command mode. The **no** version of the command discontinues the display of the debug message.

terminal [**no**] **notify**

Enables notification to a terminal when output to it is generated on any of its active sessions. The no version of this command disables notification.

terminal [**no**] **padding** *decimal-number count*

Sets the character padding on the current terminal line. The argument decimal-number is the ASCII decimal representation of the character. The argument count is the number of NULL bytes sent after that character. (The *DECbrouter 90 Configuration and Reference, Volume 3*, Appendix E, "ASCII Character Set," provides a list of the ASCII characters.) The TERMINAL NO PADDING command ends this padding.

terminal special-character-bits {**8** | **7**}

Temporarily changes the ASCII character widths to accept special characters typed in during data connection.

terminal [**no**] **width** *columns*

Sets the number of characters (columns) on a single line of the current terminal screen. The login protocol uses the argument columns to set up terminal parameters on a remote host. The **no** version of the command unsets the number of characters (columns) on the single line of the current terminal screen.

where

Displays information about open connections associated with the current terminal line and provides the connection number.

4

Configuring the System

This chapter describes how to configure the system. System configuration tasks include:

- Setting global system characteristics such as the host name and console banner message
- Defining the size of the system buffers
- Changing the system boot file specifications
- Storing and booting system software images with flash memory
- Establishing system and line passwords and system security
- Defining network services such as the IP Finger protocol
- Enabling and directing the logging of system debugging messages
- Configuring the console and virtual terminal lines

The chapter concludes with alphabetical summaries of the commands it describes.

Configuring the Global System Parameters

The following sections contain procedures and command descriptions for configuring the global system characteristics: host name and passwords and configuring system security and system management functions. The global configuration commands described in the following sections are entered in configuration mode. See the section Entering Configuration Mode in Chapter 2 for the procedures used to enter into this mode.

Setting the Host Name

Enter the HOSTNAME global configuration command to specify the host name for the router. The hostname is used in prompts and default configuration file names.

hostname *name*

The argument *name* is the new host name for the router and is case-sensitive. The default host name is *Router*.

Example

This command changes the host name to sandbox.

```
hostname sandbox
```

Displaying Banner Messages

A banner is the message that the EXEC command interpreter displays whenever a user starts any EXEC process or activates a line. The general form of the BANNER global command follows.

banner {**motd** | **exec** | **incoming**} *c text c*
no banner {**motd** | **exec** | **incoming**} *c text c*

The **motd, exec,** and **incoming** keywords control when the banner message is displayed. The use of these keywords is described in the following sections.

The argument *c* specifies a delimiting character of your choice. The argument text specifies the message to be shown on the screen whenever an interface line is activated.

The default keyword is **motd** (message of the day) if none is specified.

The **no** version of these commands removes the specified banner, and the NO BANNER command removes the motd.

Follow **banner** with one or more blank spaces, then type the delimiting character, followed by one or more lines of text. Terminate the message by typing the delimiting character a second time. There is no limit to the number of characters that can be used for the banner, with the exception of buffer limits and what is appropriate for a banner.

Example

The following example uses the pound sign character as a delimiting character:

```
banner motd #
Building power will be off from 7:00 AM until 9:00 AM this coming Tuesday.
#
```

Note

You cannot use the delimiting character in the banner message.

Displaying a Message-of-the-Day Banner

To specify a general-purpose message-of-the-day banner, use the BANNER MOTD global configuration command.

banner motd *c text c*

This command displays a message-of-the-day banner whenever any type of connection is established; for example, when a line is activated, or when an incoming Telnet connection is created. Use this banner for messages that affect all users of the router (for example, for system reboots).

Note

The command BANNER is equivalent to the command BANNER MOTD except that with banner, the banner is displayed on incoming connections only.

Displaying a Banner with an EXEC Process

To display a message when an EXEC process is created, use the BANNER EXEC global configuration command.

banner exec *c text c*

This command specifies a message to be displayed when an EXEC process is created (for example, when a TTY line is activated, or an incoming connection is established to a VTY). This banner is designed for messages that affect only interactive terminal users of the router.

Displaying an Incoming Message Banner

To display an incoming message to a particular terminal line, use the BANNER INCOMING global configuration command.

banner incoming *c text c*

This command specifies a message to be displayed on incoming connections to particular terminal lines (for example, lines used for "milking machine" applications).

Note

Messages are never displayed on incoming stream-type connections, because they might interfere with printer daemons.

The EXEC banner can be suppressed on certain lines by using the NO EXEC-BANNER line subcommand (described in the section Suppressing Banner Messages later in this chapter). Lines so configured will not display the EXEC or MOTD banners when an EXEC is created.

Order of Banner Displays

Banners and messages are displayed in the following order:

1. Any banner motd message
2. Any banner incoming message. At this point, the user logs in, if required.
3. Any banner exec message

Example

This example illustrates how to display a message-of-the-day banner and a message that will be displayed when an EXEC process is created. Use the BANNER global configuration commands and NO EXEC-BANNER line subcommand to accomplish these settings.

```
! Both messages are inappropriate for the VTYs.
line vty 0 4
no exec-banner
!
banner exec /
This is training group server.
Unauthorized access prohibited.
/
!
banner motd /
The server will go down at 6pm for a software upgrade
/
```

Setting Default Widths for International Character Sets

Digital software allows you to set default widths for characters such as banners and prompts, and for special characters such as flow control, hold, escape, and disconnect characters. Modifying the character width for EXEC and special characters allows you to include international characters in banners, prompts, and special characters.

Use these global configuration commands to specify the number of significant characters for EXEC and special characters.

default-value exec-character-bits {**8** | **7**}

default-value special-character-bits {**8** | **7**}

The DEFAULT-VALUE EXEC-CHARACTER-BITS command configures the character widths of EXEC and configuration command characters. The default value is 7 bits, which results in the use of a 7-bit ASCII character set. Configuring the EXEC character width to 8 bits allows you to add special graphical and international characters in banners and prompts.

Note

Setting the EXEC character width to 8 bits can cause failures. If a user on a terminal that is sending parity enters the command HELP, an "unrecognized command" message appears because the system is reading all 8 bits, although the eighth bit is not needed for the HELP command.

The DEFAULT-VALUE SPECIAL-CHARACTER-BITS command configures the number of characters used in special characters such as software flow control, hold, escape, and disconnect characters. The default special-character width is 7. Configuring the width to 8 allows you to use twice as many special characters as with the 7-bit setting.

See the section Setting Widths for International Character Sets for the Interface later in this chapter for configuration examples and line character-width commands. See Chapter 3 for EXEC-level character width commands.

Setting the System Buffers

In normal system operation, there are several pools of different-sized buffers. These pools grow and shrink based upon demand. Some buffers are temporary and are created and destroyed as warranted. Other buffers are permanently allocated and cannot be destroyed. The BUFFERS command allows a network administrator to adjust initial buffer pool settings, as well as the limits at which temporary buffers are created and destroyed.

Note

Normally it is not necessary to adjust these parameters; do so only after consulting with Digital staff. Improper settings could adversely impact router performance.

The full syntax of the BUFFERS command follows.

buffers {**small** | **middle** | **big** | **large** | **huge**} {**permanent** | **max-free** | **min-free** | **initial**} *number*

no buffers {**small** | **middle** | **big** | **large** | **huge**} {**permanent** | **max-free** | **min-free** | **initial**} *number*

First choose the keyword that describes the size of buffers in the pool—small, big, huge, and so on. The default number of the buffers in a pool is determined by the hardware configuration and can be displayed with the EXEC SHOW BUFFERS command.

The next keyword specifies the buffer management parameter to be changed and can be one of the following:

- **permanent**—The number of permanent buffers that the system tries to allocate. Permanent buffers normally are not deallocated by the system.
- **max-free**—The maximum number of free or unallocated buffers in a buffer pool.
- **min-free**—The minimum number of free or unallocated buffers in a buffer pool.
- **initial**—The number of additional temporary buffers that should be allocated when the system is reloaded. This keyword can be used to ensure that the router has the necessary buffers immediately after reloading in a high traffic environment.

The argument *number* specifies the number of buffers to be allocated.

The NO BUFFERS command with appropriate keywords and argument restores the default buffer values.

Dynamic Buffer Sizing

An optional global configuration command for adjusting huge buffer settings is the BUFFERS HUGE SIZE command. As with the preceding command, use only after consulting with Digital staff.

buffers huge size *number*
no buffers huge size *number*

The BUFFERS HUGE SIZE command dynamically resizes all huge buffers to the value that you supply. The buffer size cannot be lowered below the default. The argument number specifies the number of buffers to be allocated.

The **no** version of the command with the keyword and argument restores the default buffer values.

Examples

In the following example, the system will try to keep at least 50 small buffers free.

```
buffers small min-free 50
```

In this example, the system will try to keep no more than 200 medium buffers free.

```
buffers medium max-free 200
```

With the following command, the system will try to create one large temporary extra buffer, just after a reload:

```
buffers large initial 1
```

In this example, the system will try to create one permanent huge buffer:

```
buffers huge permanent 1
```

In this example, the system will resize huge buffers to 20000 bytes:

```
buffers huge size 20000
```

To display statistics about the buffer pool on the router, enter the command SHOW BUFFERS. For more information, refer to the section Monitoring System Processes in Chapter 6.

Setting Configuration File Specifications

This section describes the BOOT global configuration commands used to configure boot files. The BOOT command can be used to perform the following tasks:

- Change default filenames.
- Specify a server host for netbooting configuration files.
- Specify the size buffer to configure for netbooting a host or network configuration file.

The commands to load files over the network take effect the next time the software is reloaded, provided they have been written into nonvolatile memory.

Changing the Network Configuration Filename

The network configuration file contains commands that apply to all routers and terminal servers on a network. The default name of this file is network-confg. See the section Entering Configuration Mode in Chapter 2. To change the name of this file, use the BOOT NETWORK global configuration command. The full command syntax follows:

boot network *filename [address]*
no boot network *filename [address]*

The keyword **network** changes the network configuration filename from *network-confg*. The argument *filename* is the new name for the network configuration file. If you omit the argument *address*, the router uses the default broadcast address of 255.255.255.255. If you use *address*, you can specify a specific network host or a subnet broadcast address.

Changing the Host Configuration Filename

The host configuration file contains commands that apply to one router in particular. To change the host configuration filename, use the BOOT HOST global configuration command. The full command syntax follows:

boot host *filename [address]*
no boot host *filename [address]*

The keyword **host** changes the host configuration filename to a name you specify in the *filename* argument. The router uses its name to form a host configuration filename. To form this name, the router converts its name to all lowercase letters, removes all domain information, and appends "-confg." By default, the host filename is *router-confg*.

Specifying a Boot File Buffer Size

To specify the size of the buffer to be used for netbooting a host or a network configuration file, use the BOOT BUFFERSIZE global configuration command. The full command syntax follows:

boot buffersize *bytes*
no boot buffersize *bytes*

The argument *bytes* specifies the size of the buffer to be used. By default, it is the size of your nonvolatile memory (32k). There is no minimum or maximum size that can be specified.

The EXEC commands WRITE TERMINAL and WRITE NETWORK use the information specified by the BUFFERSIZE keyword when performing their functions (see the section Entering Configuration Mode in Chapter 2 for more information about these EXEC commands).

Configuring Multiple Instances of the Boot Commands

You can configure multiple instances of the BOOT commands. When issued, each command is executed in order and thus can be used to begin a systematic search or to build a specific list. For example, you can issue multiple BOOT commands to build an ordered list of configuration file name and host address pairs. The router scans this list until it successfully loads the appropriate network or host configuration file or system boot image. In this example, the router looks first for *fred-confg* on network 192.31.7.24 and, if it cannot load that file, then looks for *wilma-confg* on network 192.31.7.19.

```
boot host /usr/local/tftpdir/fred-confg 192.31.7.24
boot host /usr/local/tftpdir/wilma-confg 192.31.7.19
```

Note

This example uses fictitious filenames; the syntax of these filenames depends on the TFTP server from which you are loading the files.

If the router cannot find either file, a background process tries at ten-minute intervals (the default) to load one file or the other.

You can issue multiple instances of all variations of the BOOT command, including the NO BOOT forms. This feature can be useful for removing configuration files. To remove a configuration file name and host address pair from the list, use the NO BOOT command syntax.

Storing and Booting System Software Using the Flash Memory

Some versions of the DECbrouter 90 feature build in flash memory, onto which system software images can be stored, booted, and rewritten as necessary. This permits fast and simple system software upgrades. Other versions of the DECbrouter 90 use one-time programmable ROM, and require that you use Digital Services for future software upgrades. DECbrouter 90 versions with one-time programmable ROM memory will report errors when you attempt to load the flash memory. The following sections apply only to routers with flash memory.

The flash memory allows you to:

- Copy the TFTP or MOP image to flash memory
- Boot a router from flash memory either automatically or manually
- Copy the flash memory image to a TFTP server

The flash memory's features include the following:

- It can be remotely loaded with multiple system software images through a TFTP or MOP transfer.
- The DECbrouter 90 system software image resides in the flash permanently.
- It allows a router to be booted manually or automatically from a system software image stored in flash memory.
- It provides write protection against accidental erasure or reprogramming of the flash memory.

Security Precautions

The flash memory provides write-protection against accidental erasure or reprogramming of the flash memories. The system image stored in the flash memory only can be changed from a privileged EXEC command session on the console terminal, which offers system-wide security as well.

Flash Memory Configuration Overview

The following is an overview of how to configure your system to boot from flash memory. This is not a step-by-step set of instructions; rather, it is an overview to the process of using the flash capability.

1. Set your system to boot from ROM software (bootstrap mode).
2. Restore the system configuration, if necessary.
3. Copy the TFTP or MOP image to flash memories.
4. Configure from the terminal to automatically boot from the desired file in flash memory.
5. Write your configuration to memory.

6. Set your system to netboot from flash (requires configuration register change).
7. Power-cycle and reboot your system to ensure that all is working as expected and that the configuration is stored in NVRAM.

The remainder of this section describes the configuration commands used with the flash feature.

Verifying Installation and Displaying Flash ROM Statistics

The SHOW FLASH command displays the total amount of flash memory present; any files that may currently exist in flash memory and their size, and the amounts of Flash memory used and remaining.

The SHOW FLASH or SHOW FLASH ALL commands will display the names of the system software images.

show flash

show flash all

The following shows sample output of the show flash command:

```
Chrysostom# show flash
4096K bytes of flash memory sized on embedded flash.
Flash: running from Flash. READ-ONLY
File  name/status
0  flash
[2104184/4194304 bytes free/total]
```

Table 4–1 and Table 4–2 explain the SHOW FLASH fields.

Table 4–1 Show Flash Field Descriptions

Field	Description
Bytes of flash memory sized on embedded flash	Total amount of flash memory present.
File	The position of the file within memory.
Name/status	The name of the file and whether it is deleted or invalidated.

The following shows sample output of the SHOW FLASH ALL command:

```
Chrysostom(boot)# show flash all
4096K bytes of flash memory sized on embedded flash.
   Chip    socket   code      bytes     name
    0       E9      89A2   0x100000   INTEL 28F008SA
    1       E20     89A2   0x100000   INTEL 28F008SA
    2       E13     89A2   0x100000   INTEL 28F008SA
    3       E25     89A2   0x100000   INTEL 28F008SA

Flash file directory:
File  name/status
        addr     length    fcksum  ccksum
 0  DEWBR
      0x3000040  2090056   0x554C 0x554C
 [2104184/4194304 bytes free/total]
```

Note

The SHOW FLASH ALL command will show abbreviated output when the DECbrouter 90 is not in the Bootstrap Mode. See *"Copying the TFTP Image to Flash Memory"* for information on entering bootstrap mode.

Table 4–2 Show Flash All Field Descriptions

Field	Description
Bytes of flash memory sized on embedded flash	Total amount of flash memory present on the DECbrouter 90.
Chip, Socket	Number and location of ROM on the circuit board.
Code	Vendor code.
Bytes	Size in hex bytes.
Name	Vendor name and part number.
Contains:	Files that may currently exist in flash memory and their size. Also the amounts of flash memory used and remaining.
[invalidated] [deleted]	Flag that appears when a file is rewritten (recopied) into flash memory, when a user aborts, when a network times out, or when there is a flash memory overflow.
Address	First byte in the raw file image.
Length	Length of the raw image in bytes.
fcksum ccksum	Image checksums.

When you see the [*invalidated*] flag, a prompt will tell you that the identical file already exists and that it will be invalidated. The first (now invalidated) copy of the file is still present within flash memory, but it is rendered unusable in favor of the newest version.

To eliminate any files from flash (invalidated or otherwise) and free up all available memory space, the entire flash memory must be erased; individual files cannot be erased from flash memory.

Both examples illustrate that the flash memory can store and display multiple, independent software images (*gsxx* and *tsyy*) for booting itself or for TFTP serving software for other products. This feature would be most useful for storing default system software as a backup. These images also can be stored in compressed format.

Entering the Bootstrap Mode

Before the flash ROM may be loaded from either the TFTP or MOP, the DECbrouter 90 must be placed in bootstrap mode.

Flash ROM comes pre-loaded with the latest software. The user loads flash only after receiving a software update. This requires a new software configuration register setting and a router reboot. To put the DECbrouter 90 into bootstrap mode, set bits 3-0 of the configuration register to 0001. This may be done with the configuration command CONFIG-REGISTER or other methods described in

the *DECbrouter 90 Getting Started* guide, the section Setting the Configuration Register.

To upgrade firmware follow the steps below:

1. Enter bootstrap mode.

   If you are using the default configuration register setting, you may use the global configuration command CONFIG-REGISTER to change the configuration register to 0x101. Otherwise refer to the *DECbrouter 90 Getting Started* guide, the section Setting the Configuration Register for instructions on reading, interpreting, and modifying the configuration register. (In effect, setting the boot field of the configuration register involves changing the rightmost hex digit of the register value to a 1.)

   Note

   After loading the flash image, return to booting from flash (exit bootstrap mode); write 0x102 to the configuration register (set the boot field to flash) and reboot.

2. Reinitialize or reboot the router to enter the bootstrap mode. In bootstrap mode the DECbrouter 90 prompts will consist of: `router name(boot)>`.
3. Put the router in enabled mode by issuing the ENABLE command. The router must be in enabled mode before the flash can be loaded.

   From this mode you may load the flash ROM or do a **show flash all** for complete flash data.
4. Use the **copy tftp flash** or **copy mop flash** commands to load the new code.
5. Set the configuration register to boot the flash code by setting the boot field to 2. For the default configuration, the configuration register value would be 0x102.
6. Reinitialize or reboot. The router will now boot the new code.

Example

The following example will configure the router for bootstrap mode and then reboot it for flash loading.

```
Chrysostom>enable

Password:
Chrysostom#configure terminal

Enter configuration commands, one per line.
Edit with DELETE, CTRL/W, and CTRL/U; end with CTRL/Z
config-register 0x101
^Z

Chrysostom#
%SYS-5-CONFIG_I: Configured from console by console ()

Chrysostom#reload
[confirm]
```

```
%SYS-5-RELOAD: Reload requested
System Bootstrap, Version 4.6
Copyright (c) 1986-1992 by cisco Systems
2000 processor with 1024 Kbytes of main memory

              Restricted Rights Legend

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

Cisco Systems, Inc.
1525 O'Brien Drive
Menlo Park, California  94025

3000 Software (IGS-RXBOOT Bootstrap), Version 9.14
Copyright (c) 1986-1992 by cisco Systems, Inc.
Compiled Tue 03-Nov-92 14:41

DECbrouter 90 (68030) processor (revision 0x00) with 1024K/1024K bytes of
memory.
Processor board serial number 00000000
DDN X.25 software.
1 Ethernet/IEEE 802.3 interface.
2 Serial network interface.
32K bytes of non-volatile configuration memory.
4096K bytes of flash memory sized on embedded flash.


Press RETURN to get started!


Chrysostom(boot)>enable

Password:
Chrysostom(boot)#
```

Example

The following example configured the router to return to booting flash after loading the flash ROM.

```
Chrysostom(boot)#configure terminal

Enter configuration commands, one per line.
Edit with DELETE, CTRL/W, and CTRL/U; end with CTRL/Z
config-register 0x102
^Z

Chrysostom#
%SYS-5-CONFIG_I: Configured from console by console ()

Chrysostom#reload
[confirm]

%SYS-5-RELOAD: Reload requested
```

Copying the TFTP Image to Flash Memory

The COPY TFTP FLASH command copies (writes) a TFTP image into the current flash configuration:

copy tftp flash

Caution

Do not enter a WRITE MEMORY command while in the bootstrap mode. The bootstrap code does not understand all the configuration commands and will corrupt your NVRAM configuration if you attempt to write it out in bootstrap mode. Make all configuration changes in flash mode.

Note

While the flash supports loading multiple files into flash memory, normal software images usually exceed 2 megabytes in size, making it impossible to load one of them in a DECbrouter 90 with only 4 megabytes of flash memory. Therefore, when loading a full image, you should request that the COPY TFTP FLASH command erase the flash before the load. If, however, there is some question about if a new image can fit without erasing the existing images, use the SHOW FLASH command and compare the size of the file you wish to copy to the amount of available flash memory shown. If the space available is less than the space required by the file you wish to copy, the copy process will continue, but the entire file will not be copied into flash. A failure message, *buffer overflow - xxxx/xxxx*, will appear, where xxxx/xxxx is the number of bytes read in /number of bytes available.

Once you issue the COPY TFTP FLASH command, the system prompts you for the IP address (or domain name) of the TFTP server. This may be another router serving ROM or flash system software images. You are then prompted for the file name of the software image and given the option to erase the existing flash memory before writing onto it only when there is free space available in Flash memory. *If no free flash memory space is available, the erase routine is required before new files can be copied*. The system will be prompt you for these conditions. The flash memory is loaded with the system software image at the factory before shipment.

Following is sample output (copying a system image named *gsxx*) of the prompt you will see under these conditions:

```
Chrysostom(boot)# copy tftp flash
IP address or name of remote host [255.255.255.255]?
131.131.101.101
Name of tftp filename to copy into flash []? gsxx
copy gsxx from 131.131.101.101 into flash memory? [confirm]
Flash is filled to capacity. (This line only appears if
Flash memory is full.)
Erasure is needed before flash may be written.
Erase flash before writing? [confirm]
```

Note

Entering n after the "Erase flash before writing?" prompt would continue the copy process. Entering a y would proceed with the erase routine. Make certain you have ample flash memory space before entering n at the erasure prompt.

When you erase the existing flash, the system clears and initializes each flash memory and displays a pound sign (#) prompt for each cleared and initialized device (total of 16). The entire copying process takes several minutes and will differ from network to network.

Note

The file name gsxx can be in either lower- or uppercase; the system will see GSxx as gsxx. If both are copied to flash, the second file copied will become the valid file.

Following is sample output from copying a system image named gsxx into the current flash configuration:

```
Chrysostom(boot)# copy tftp flash
IP address or name of remote host [255.255.255.255]? 131.131.
101.101
Name of tftp filename to copy into flash []? gsxx
copy gsxx from 131.131.101.101 into flash memory?[confirm]
xxxxxxxx bytes available for writing without erasure.
erase flash before writing? [confirm]
Clearing and initializing flash memory (please wait)01/02/03/04/05/06/07/08
/09/10/11/12/13/14/15
Loading from 101.2.13.110: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!... [OK - 324572/524212 bytes]
Verifying checksum...
vvvvvvvvvvvvvvvvvvvvvvvv
Flash verification successful.  Length = 324572, checksum = 0xE2E2
```

Note

To abort this copy process, press the Ctrl, Shift, and 6 keys simultaneously. Although the process will abort, the partial file copied before the abort was issued will remain until the entire Flash memory is erased.

The series of exclamation points (!) in the previous sample output indicates that the copying process is taking place. The series of v's indicates that a checksum verification of the flash memory is occurring as it is loaded into memory for boot. Checksum verification occurs only through data compare during programming of the flash memory. The last line in the sample configuration indicates that the copy is successful.

Having successfully copied an image onto the flash, the output of **show flash all** will provide the image name, as in the following sample output:

```
Chrysostom(boot)#sho flash all
4096K bytes of flash memory sized on embedded flash.
    Chip  Socket  Code    Bytes     Name
     0     E9     89A2  0x100000  INTEL 28F008SA
     1     E20    89A2  0x100000  INTEL 28F008SA
     2     E13    89A2  0x100000  INTEL 28F008SA
     3     E25    89A2  0x100000  INTEL 28F008SA

Flash file directory:
File  name/status
        addr     length   fcksum  ccksum
 0  f
      0x3000040  2090056  0x554c 0x554c
 [2104184/4194304 bytes free/total]
```

You are now ready to boot from flash. The following sections describe how to automatically and manually boot from flash memory.

Copying the MOP Image to Flash Memory

New releases of the system image may also be loaded from VMS using the MOP protocol. The software must be ready to load from a local MOP VMS router. The procedure is similar to loading a TFTP image. As in loading the TFTP image, you must be in bootstrap mode to load the flash. See the previous section for instructions on loading the flash.

The command for MOP loading the flash is COPY MOP FLASH. You will be prompted for the file name. As with the TFTP flash load, you are given an option to erase existing flash memory before writing onto it. The system then clears and initializes each flash memory. A pound sign (#) prompt is displayed for each cleared and initialized device (16 total). The entire copy process takes several minutes and will differ from network to network.

The following example shows sample output of copying a system image, which is named DEWBR, into the current flash configuration.

Example

```
Router(boot)#copy mop flash
File name/status
0 DEWBR
[2243008/4194304 bytes free/total]

Name of file to copy ? DEWBR
Copy DEWBR from MOP server into flash memory ? [confirm]
2242944 bytes available for writing without erasure
Erase flash before writing? [confirm]
Flash erase: erasing bank 0 ..
00/01/02/03/04/05/06/07/08/09/10/11/12/13/14/15/

Mope2flash: Loading DEWBR into flash from aa00.0400.d116 on Ethernet0:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! [OK - 1648952/4194176 bytes]
Verifying via checksum...
vvvvvvvvvvvvvvvvvvvvvvvvvvvv
Verification successful: Length = 1951232, checksum = 0xEFF1
```

The series of pound signs (#) indicate each flash device is being cleared and

initialized; one per device (16 total). The last line in the sample configuration indicates that the copy is successful.

The exclamation points indicate the copy process.

The series of Vs in the above sample output indicates that a checksum verification of the Flash is occurring as it is loaded into memory for boot. It is verified only through data compare during programming of the flash.

Note

Should you wish to abort this copy process, press the Ctrl, Shift, and 6 keys simultaneously. The process will abort; however, the data that was copied before the abort was issued, will remain until the entire flash memory is erased. Individual files cannot be erased from flash memories.

If the TFTP image is too large, the following failure message appears:

```
buffer  overflow  - xxxx/xxxx
```

The value xxxx/xxxx is the number of bytes read in/number of bytes available.

Use the output of **show flash all** to obtain the image name. See Verifying Installation and Displaying Flash ROM Statistics in this chapter for more information about this EXEC command.

Once you have configured flash memory, you are now ready to boot from flash. See Automatically Booting from Flash Memory in this chapter for more information. You can also manually boot from the ROM monitor using the B FLASH command. See Manually Booting from Flash Memory in this chapter for more information.

Automatically Booting from Flash Memory

In most situations, you will want the router configured to automatically boot the flash memory every time it reboots. (The other options are to boot the bootstrap code or enter the monitor.) This is accomplished by setting the boot field of the configuration register to boot from flash.

The section "Setting the Configuration Register" in the *DECbrouter 90 Getting Started* guide describes complete instructions on setting the processor register. The easiest way to set the configuration register is by using the CONFIG-REGISTER global configuration command.

To configure the register to automatically boot from flash, use the SHOW VERSION command first to get the current hexadecimal value of the configuration register:

```
Chrysostom#show version
3000 Software (DEWBR), Version 9.14
Copyright (c) 1986-1992 by Cisco Systems, Inc.
Compiled Tue 02-Feb-93 14:32

System Bootstrap, Version 4.6

Chrysostom uptime is 1 day, 21 hours, 24 minutes
System restarted by reload
System image file is "DEWBR", booted via flash
```

```
DECbrouter 90 router (68030) processor (revision 0x00) with 1024K/1024K
bytes of memory.
Processor board serial number 00000000
DDN X.25 software, Version 2.0.
Bridging software.
SuperLAT software (copyright 1990 by Meridian Technology Corp).
1 Ethernet/IEEE 802.3 interface.
2 Serial network interface.
32K bytes of non-volatile configuration memory.
4096K bytes of flash memory sized on embedded flash.
Configuration register is 0x101

Chrysostom#
```

The "configuration register" line shows the current hexadecimal value. The boot field is represented by the rightmost hex digit. Change this digit to the value 2 ("Boot from flash," see the descriptions of all the values in the *DECbrouter 90 Getting Started* guide, and write the new configuration register value with the CONFIG-REGISTER configuration command:

```
Chrysostom# config terminal

Enter configuration commands, one per line.
Edit with DELETE, CTRL/W, and CTRL/U; end with CTRL/Z

config-register 0x102
^Z
```

Upon reboot, the configuration register value will take effect, and the router will automatically boot from flash upon initialization.

Refer to the *DECbrouter 90 Getting Started* guide for more information on the configuration register and various boot modes.

Manually Booting from Flash Memory

Use the B FLASH command at the ROM monitor level to manually boot the system, as in the following example. Check the *DECbrouter 90 Getting Started* guide for the correct configuration register setting.

```
>b flash
F3: 1578668+35572+156084 at 0x30000060
{ROM Monitor copyrights...}
```

Copying the Flash Memory Image to a TFTP Server

To copy an image back to a TFTP server, use the COPY FLASH TFTP command. This copy of the system image can serve as a backup copy and can also be used to verify that the copy in Flash is the same as on the original file on disk.

copy flash tftp

The following is an example of the use of this system configuration command:

```
Chrysostom# copy flash tftp
IP address of remote host [255.255.255.255]? 101.2.13.110
Name of file to copy []? gsxx
writing gsxx !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Chrysostom#
```

Note

A domain name can be used in place of the IP address of the remote host.

Configuring Flash Memory as a TFTP Server

The DECbrouter 90 can be used as a trivial file transfer protocol (TFTP) file server for other routers on the network. This feature allows you to load system software into the Flash of other DECbrouter 90s from a pre-loaded DECbrouter 90. This may be useful for mass software upgrades.

For clarification in the description that follows, one router (with flash memory) will be referred to as the flash server and all other routers will be referred to as client routers. The configurations for the flash server router will be given through example configurations, with commands included as necessary. After the flash server is configured, the client router may be loaded using the "Copying TFTP image to flash memory" instructions; use the flash server address as the TFTP server.

Prerequisites

The flash server and client router must be able to reach one another before the TFTP function can be implemented. Verify this connection by using the PING command to move between the flash server and client router (in either direction).

An example use of the PING command is as follows:

```
Router# ping 131.131.101.101 <Return>
```

In this example, the internet protocol (IP) address of 131.131.101.101 belongs to the client router. Connectivity is indicated by !!!!!, while ... [timed out] or [failed] indicates none. If the connection fails, reconfigure the interface, check the physical connection between the flash server and client router, and issue PING again.

After this connection is verified, use the SHOW FLASH command to locate the file name of the boot image present in the flash memory of the flash server. You will need this file name to TFTP load the client router. The next example uses the file name gs3-bfx..91.1 for the boot image.

Once you have the file name of the boot image in flash memory, you can configure the flash server.

Configuring the Flash Server

Configure the flash server by adding both the TFTP-SERVER SYSTEM command and the ACCESS-LIST command to the configuration memory. Use the CONFIGURE-TERMINAL command to do so.

Following is sample output of these commands:

```
Server# configure terminal
Enter configuration commands, one per line.
Edit with DELETE, CRTL/W, and CRTL/U; end with CTRL/Z
tftp-server system gs3-bfx.91.1 1
access-list 1 permit 131.131.101 0.0.0.255
^Z
Server# write memory <Return>
[ok]
Server#
```

This example gives the file name of the software image in the flash server and one access list (labeled 1). The access list must include the network within which the client router resides. Thus, in the example, the network 131.131.101.0 and any client routers on it are permitted access to the flash server, file name gs3-bfx.91.1.

For more information on access lists, refer to the *DECbrouter 90T Products, Configuration and Reference, Volume 2*, Chapter 5, "IP Routing."

Caution

Using the no boot system command in the following example will invalidate all other boot system commands currently in the client router system configuration. Before proceeding, determine whether the system configuration stored in the router you will use as the client should first be saved (uploaded) to a TFTP file server. Refer to Chapter 2 for instructions on uploading and downloading system configuration files.

Establishing Passwords and System Security

This section describes how to configure password protection and terminal access security.

You can set passwords to control access to the privileged command level and to individual lines. The terminal access controller access control system (TACACS) protocol controls terminal use by means of a user-ID-and-password pair. The Defense Data Network (DDN) developed TACACS to control access to its TAC terminal servers; the DECbrouter 90 TACACS support is patterned after the DDN application.

These system security measures may not provide the level of protection needed for some environments; individual routing protocols and bridging support may have additional security procedures. You also may need to use access lists for additional protection. For access list configuration procedures, refer to the sections describing configuration of a particular routing protocol or bridging support.

Establishing the Privileged-Level Password

To assign a password for the privileged command level, use the ENABLE PASSWORD global configuration command.

enable password *password*

The argument *password* is case-sensitive and specifies the password prompted for in response to the EXEC command ENABLE. The *password* argument can contain any alphanumeric characters, including spaces, up to 80 characters. Password checking is also case-sensitive. The password *Secret* is different from the password *secret*, for example, and the password *two words* is an acceptable password.

To enter the privileged command level, enter the following EXEC command and then press Return:

enable

Next, enter the password for the privileged command level at the *Password:* prompt.

When you use the ENABLE command at the console terminal, the EXEC will not prompt for a password if the privileged mode password is not set. This feature allows access to the CONFIGURE command to enter configuration command collection mode, so you can set parameters—such as the password for the privileged command level—for other lines. To restrict access to the console line, set a line password as described in the section Establishing Line Passwords later in this chapter.

Example

The following example sets the password secretword for the privileged command level on all lines, including the console:

```
enable-password secretword
```

Specifying a Password

When an EXEC is started on a line with password protection, the EXEC prompts for the password. If you enter the correct password, the EXEC prints its normal nonprivileged prompt. You can try three times to enter a password before the EXEC exits and returns the terminal to the idle state.

To specify a password, use the PASSWORD line subcommand. The full command syntax follows:

password *text*
no password

The *text* argument can contain any alphanumeric character, including spaces, up to 80 characters. The password checking is also case-sensitive. The password *Secret* is different than the password *secret*, for example, and the password *two words* is an acceptable password.

To enable checking for the password specified by the PASSWORD command, use the line subcommand LOGIN:

login

Alternatively, to use the TACACS user ID and password-checking mechanism, use the following subcommand:

login tacacs

To disable all password checking, use the following command:

no login

The server prints the message-of-the-day banner before prompting for a password, so you immediately see messages such as no trespassing notifications. By default, virtual terminals require a password. If you do not set a password for a virtual terminal, it will respond to attempted connections by displaying an error message and closing the connection. Use the NO LOGIN subcommand to disable this behavior and allow connections without a password.

Example

The following example sets the password *letmein* on line 5:

```
line 5
password letmein
login
```

Recovering from a Lost Password

It is possible to lock yourself out if you enable password checking on the console terminal line and then forget the line password.

To recover from this situation, force the DECbrouter 90 into factory diagnostic mode. Set bit 7 of the processor configuration register from the monitor and reboot the DECbrouter 90. (See *DECbrouter 90 Getting Started* guide, the section Setting the Configuration Register.) Follow these steps:

1. You will be asked if you want to set the manufacturers' addresses. Respond by typing yes. You then see the following prompt:
   TEST-SYSTEM>
2. Enter the ENABLE command to get the privileged prompt, as in:
   TEST-SYSTEM# **enable**
3. Enter the SHOW CONFIGURATION command to review the system configuration and find the password.
4. To resume normal operation, set bit 7 of the configuration register to zero, and reboot the router. (You may use the CONFIG-REGISTER configuration command to do this.)
5. Log in to the router with the password that was shown in the configuration file.

The processor configuration registers are described in the *DECbrouter 90 Getting Started* guide.

When the router restarts in factory diagnostic mode, it does not read the nonvolatile memory, thus avoiding the command to set a password for the console terminal. Do not change any router configuration in the factory diagnostic mode.

Note

All debugging capabilities are turned on during diagnostic mode.

Encrypting Passwords

You can increase access security to your router by encrypting passwords. When password encryption is enabled, the encrypted forms of the passwords are displayed when a SHOW CONFIG command is entered. Use the SERVICE PASSWORD-ENCRYPTION global configuration command:

service password-encryption
no service password-encryption

The actual encryption process occurs when the current configuration is written or when a password is configured. Password encryption can be applied to both

the privileged command password and to console and virtual terminal line access passwords. See the section Establishing Line Passwords later in this chapter.

Note

It is not possible to recover a lost encrypted password.

Example

The following example causes no encryption to take place:

```
gw# config
service password-encryption
no service password-encryption
```

The next example encrypts the privileged command password secret. Additional passwords are not encrypted. The encrypted password remains in its encrypted form until it is reconfigured with a nonencrypted password:

```
gw# config
service password-encryption
enable-password secret
no service password-encryption
```

If you modify a configuration file and add a SERVICE PASSWORD-ENCRYPTION configuration command, when you write the configuration file all passwords will become encrypted.

Establishing Terminal Access Control

DECbrouter 90 provides unsupported versions of both a standard and an extended TACACS server. The servers run on most UNIX systems and are available via FTP from the host. You can use the servers to create UNIX accounting applications that monitor use of a system and user logins.

The configuration commands in the following sections tailor the behavior of the standard TACACS server.

Setting the Server Host Name

The TACACS-SERVER HOST global configuration command specifies a TACACS host. The full syntax of this command follows.

tacacs-server host *name*
no tacacs-server host *name*

The argument *name* is the name or Internet address of the host. You can use multiple TACACS-SERVER HOST subcommands to specify multiple hosts. The server will search for the hosts in the order you specify them. The NO TACACS-SERVER HOST global configuration command deletes the specified name or address.

Limiting Login Attempts

The TACACS-SERVER ATTEMPTS global configuration command controls the number of login attempts that can be made on a line set up for TACACS verification.

tacacs-server attempts *count*
no tacacs-server attempts

The argument *count* is the number of attempts. The default is three attempts. The NO TACACS-SERVER ATTEMPTS global configuration command restores the default.

Example

This command changes the login attempt to just one try:

```
!
tacacs-server attempts 1
!
```

Controlling Retries

The TACACS-SERVER RETRANSMIT global configuration command specifies the number of times the server will search the list of TACACS server hosts before giving up. The server will try all servers, allowing each one to time out before increasing the retransmit count.

tacacs-server retransmit *retries*
no tacacs-server retransmit

The argument *retries* is the retransmit count. The default is two retries.

The NO TACACS-SERVER RETRANSMIT global configuration command restores the default.

Example

This command specifies a retransmit counter value of five times:

```
!
tacacs-server retransmit 5
!
```

Setting the Timeout Intervals

The TACACS-SERVER TIMEOUT global configuration command sets the interval that the server waits for a server host to reply.

tacacs-server timeout *seconds*
no tacacs-server timeout

The argument *seconds* specifies the number of seconds. The default interval is five seconds. The NO TACACS-SERVER TIMEOUT global configuration command restores the default.

Example

This command changes the interval timer to ten seconds:

```
!
tacacs-server timeout 10
!
```

Setting the Last Resort Login Feature

If, when running the TACACS server, it does not respond, the default action is to deny the request. Use the TACACS-SERVER LAST-RESORT global configuration command to change the default.

tacacs-server last-resort {**password** | **succeed**}
no tacacs-server last-resort {**password** | **succeed**}

The command causes the router to request the privileged password as verification, or forces successful login without further input from the user, depending upon the keyword specified, as follows:

- PASSWORD—Allows the user to access the privileged-level command mode by entering the password set by the ENABLE command.
- SUCCEED—Allows the user to access the privileged-level command mode without further question.

Note

The last resort login feature can be useful when it is important to be able to ensure that login can occur. An example of such a condition is when a system administrator needs to log in to troubleshoot TACACS servers that are down.

The NO TACACS-SERVER LAST-RESORT global configuration command restores the system to the default behavior.

Establishing Privileged-Level TACACS

The following variations of the ENABLE command can be used to configure privileged-level command access using the TACACS protocol.

Enabling the Privileged Mode

The ENABLE USE-TACACS global configuration command is used for setting the TACACS protocol to determine whether a user can access the privileged command level.

enable use-tacacs
no enable use-tacacs

If you use this command, the EXEC ENABLE command will ask for both a new user name and a password. This information is then passed to the TACACS server for authentication. If you are using the extended TACACS, it will also pass any existing UNIX user identification code to the server.

Note

When used without extended TACACS, this command allows anyone with a valid user name and password to access the privileged command level, creating a potential security problem. This is because the TACACS query resulting from entering the ENABLE command is indistinguishable from an attempt to log in without extended TACACS.

Enabling the Privileged Mode Last Resort Login Feature

The ENABLE LAST-RESORT global configuration command allows you to specify what happens if the TACACS servers used by the ENABLE command do not respond.

enable last-resort {**password** | **succeed**}
no enable last-resort {**password** | **succeed**}

The default action is to fail. Use of the keyword changes the action, as follows:

- PASSWORD—Allows you to enable by entering the privileged command level password.
- SUCCEED—Allows you to enable without further question.

The NO ENABLE LAST-RESORT global configuration command restores the default.

Configuring TACACS Accounting

The following sections describe the configuration commands that tailor the behavior of the extended TACACS client.

Enabling Extended TACACS Mode

The TACACS-SERVER EXTENDED global configuration command enables an extended TACACS mode.

tacacs-server extended
no tacacs-server extended

This mode provides information about the terminal requests for use in setting up host auditing trails and accounting files for tracking use of terminal servers and routers. Information includes responses from terminal servers and routers and validation of user requests. An unsupported, extended TACACS server is available from Cisco Systems via anonymous FTP (ftp.cisco.com) for UNIX users who want to create the auditing programs.

The NO TACACS-SERVER EXTENDED command disables this mode.

Login Notification

The TACACS-SERVER NOTIFY global configuration command causes a message to be transmitted to the TACACS server, with retransmission being performed by a background process for up to five minutes. The terminal user, however, receives an immediate response allowing access to the terminal. The full syntax of this command follows.

tacacs-server notify {**connect** | **enable** | **logout**}
no tacacs-server notify {**connect** | **enable** | **logout**}

The keywords specify notification of the TACACS server whenever a user does one of the following:

- **connect**—Makes TCP connections
- **enable**—Enters the ENABLE command
- **logout**—Logs out

The NO TACACS-SERVER NOTIFY command used with the appropriate keyword disables notification.

Note

When used with extended TACACS, the command TACACS-SERVER NOTIFY ENABLE allows anyone with a valid user name and password to access the privileged-level command mode.

Login Authentication

The TACACS-SERVER AUTHENTICATE command requires a response from the network or communication server to indicate whether the user may perform the indicated action.

tacacs-server authenticate {**connect** | **enable**}
no tacacs-server authenticate {**connect** | **enable**}

Actions that require a response include the following, specified as optional keywords:

- **connect**—Makes TCP connections
- **enable**—Enters the ENABLE command

The NO TACACS-SERVER AUTHENTICATE command used with the appropriate keyword disables the action.

Optional Password Verification

You can specify that the first TACACS request to a TACACS server is made without password verification. This option is configured with the TACACS-SERVER OPTIONAL-PASSWORDS global configuration command.

tacacs-server optional-passwords

When the user enters the login name, the login request is transmitted with the name and a zero-length password. If accepted, the login procedure completes.

If the TACACS server refuses this request, the terminal server prompts for a password and tries again when the user supplies a password. The TACACS server must support authentication for users without passwords to make use of this feature. This feature supports all TACACS requests, such as login, SLIP, and enable.

Authenticating User Names

Networks that cannot support a TACACS service still may wish to use a user name-based authentication system. In addition, it may be useful to define certain user names that get special treatment (for example, an "info" user name that does not require a password, but connects the user to a general-purpose information service).

The router software supports these needs by implementing a local USERNAME configuration command. The format for the command is as follows:

username *name* [**nopassword** | **password** *encryptiontype password*]
username *name* [**accesslist** *number*]
username *name* [**autocommand** *command*]
username *name* [**noescape**] [**nohangup**]

Multiple USERNAME commands can be used to specify options for a single user.

The **nopassword** keyword means that no password is required for this user to log in. This is usually most useful in combination with the **autocommand** keyword.

The **password** keyword specifies a possibly encrypted password for this user name.

The *encryptiontype* argument is a single-digit number. Currently defined encryption types are 0, which means no encryption, and 7, which specifies a DECbrouter 90-defined encryption algorithm. Passwords entered unencrypted are written out with the DECbrouter encryption. A *password* can contain embedded spaces and must be the last option specified in the USERNAME command.

The **accesslist** keyword specifies an outgoing access list that overrides the access list specified in the **access class** line configuration subcommand. It is used for the duration of the user's session. The access list number is specified by the number argument.

The **autocommand** keyword causes the command specified by the command argument to be issued automatically after the user logs in. When the command is complete, the session is terminated. Because the command can be any length and contain embedded spaces, commands using the **autocommand** keyword must be the last option on the line.

The **noescape** keyword prevents using an escape character on the host to which the user is connected.

The **nohangup** keyword prevents the router from disconnecting the user after an automatic command (set up with the **autocommand** keyword) has completed. Instead, the user gets another login prompt.

Examples

To implement a service similar to the UNIX WHO command, which can be given at the login prompt and lists the current users of the router, the command takes the following form:

```
username who nopassword nohangup autocommand show users
```

To implement an ID that will work even if all the TACACS servers break, the command is as follows:

```
username superuser password superpassword
```

Configuring the Simple Network Management Protocol (SNMP)

The simple network management protocol (SNMP) provides a way to access and set configuration and run-time parameters for the router. The DECbrouter 90 implementation of SNMP is compatible with RFCs 1155, 1156, and 1157. The management information base (MIB) supports RFCs 1155 to 1213 and provides DECbrouter 90-specific variables.

A separate document, available in RFC 1213-type (MIB II) format, describes all the DECbrouter 90-specific SNMP variables in the Digital portion of the MIB. It also describes what is required to establish minimum configuration. Contact Digital to obtain a copy of this document, which includes instructions for accessing the variables using SNMP.

Enabling and Disabling the SNMP Server

To configure the SNMP server, you need to be in the configuration command collection mode. Enter this mode using the EXEC command CONFIGURE at the EXEC prompt. See the section Entering Configuration Mode in Chapter 2 of this manual for a description of the procedure.

Begin SNMP operation by entering the configuration commands that define the desired operation. To disable SNMP server operations on the DECbrouter 90 after it has been started, use the NO SNMP-SERVER global configuration command.

no snmp-server

Defining the SNMP Server Access List

To set up an access list that determines which hosts can send requests to the router, use the SNMP-SERVER ACCESS-LIST global configuration command. This access list applies only to the global read-only SNMP agent configured with the command SNMP-SERVER COMMUNITY. The DECbrouter 90 ignores packets from hosts that the access list denies.

The full command syntax follows.

snmp-server access-list *list*
no snmp-server access-list *list*

The argument *list* is an integer from 1 through 99 that specifies an IP access list number.

The NO SNMP-SERVER ACCESS-LIST global configuration command removes the specified access list.

Example

This command causes the router to ignore packets from hosts that do not match access list 21:

```
!
snmp-server access-list 21
!
```

Setting the Community String

To set up the community access string, use the SNMP-SERVER COMMUNITY global configuration command. The full command syntax follows.

snmp-server community *string* [**RO** | **RW**] [*list*]
no snmp-server community [*string*] [**RO** | **RW**] [*list*]

This command enables SNMP server operation on the DECbrouter 90. The argument string specifies a community string that acts like a password and permits access to the SNMP protocol.

By default, an SNMP community string permits read-only access (keyword **RO**); use the keyword **RW** to allow read-write access. The optional argument list is an integer from 1 through 99 that specifies an access list of Internet addresses that can use the community string.

The NO SNMP-SERVER COMMUNITY global configuration command removes the specified community string or access list.

Example

This command assigns the string *comaccess* to the SNMP server, allows read-only access, and specifies that addresses that match the criteria in access list 4 can use the community string. (Notice that the string is entered without quotes or any other parsing characters.)

```
snmp-server community comaccess RO 4
```

Note

Neither the console password nor the enabled-mode password can be used as the community string.

Establishing the Message Queue Length

To establish the message queue length for each TRAP host, use the SNMP-SERVER QUEUE-LENGTH global configuration command.

snmp-server queue-length *length*
no snmp-server queue-length

This command defines the length of the message queue for each TRAP host.

The argument *length* is the number of TRAP events that can be held before the queue must be emptied; the default is 10. Once a TRAP message is successfully transmitted, software will continue to empty the queue, but never faster than at a rate of four TRAP messages per second.

The NO SNMP-SERVER QUEUE-LENGTH command resets the queue length to its default value of 10.

Example

This command establishes a message queue that traps four events before it must be emptied:

```
snmp-server queue-length 4
```

Establishing Packet Filtering

To establish the packet filtering size, use the SNMP-SERVER PACKETSIZE global configuration command. The full command syntax follows.

snmp-server packetsize *bytes*
no snmp-server packetsize

This command allows control over the largest SNMP packet size permitted when the SNMP server is receiving a request or generating a reply.

The argument *bytes* is a byte count from 484 through 8192. The default is 484. The NO SNMP-SERVER PACKETSIZE command resets this default.

Example

This command establishes a packet filtering maximum size of 1024 bytes:

```
snmp-server packetsize 1024
```

Establishing the TRAP Message Recipient

To specify the recipients of TRAP messages, use the SNMP-SERVER HOST global configuration command. The full syntax follows.

snmp-server host *address community-string* [**snmp** | **tty**]
no snmp-server host *address community-string*

This command specifies which host or hosts should receive TRAP messages. You need to issue the SNMP-SERVER HOST command once for each host acting as a TRAP recipient.

The argument *address* is the name or Internet address of the host. The argument *community-string* is the password-like community string set with the SNMP-SERVER COMMUNITY command.

The optional keywords define whether the TRAPS be included, as follows:

- **snmp**—Causes all SNMP-type TRAP messages to be sent and starts the DECbrouter 90-specific RELOAD TRAP message.

- **tty**—Causes TCP connection TRAP messages to be included.

With the SNMP-SERVER HOST command, you get all the SNMP TRAP messages about TTY events by default. The NO SNMP-SERVER HOST command removes the specified host.

Examples

This command sends all possible SNMP TRAPS to 131.108.2.160, including TTY TRAPS.

```
snmp-server host 131.108.2.160
```

In order to turn these TRAP messages off, use the NO SNMP-SERVER HOST command. For example, the sequence of commands that follows sends only the seven SNMP TRAP messages to 131.108.2.160, not all the others.

```
snmp-server host 131.108.2.160
no snmp-server host 131.108.2.160 tty
```

This example causes all the SNMP-type messages to be sent to the host specified by the name gatekeeper.dec.com. The command uses the community string comaccess as the password:

```
snmp-server host gatekeeper.dec.com comaccess snmp
```

Establishing TRAP Message Authentication

To establish the TRAP message authentication, use the SNMP-SERVER TRAP-AUTHENTICATION global configuration command.

snmp-server trap-authentication
no snmp-server trap-authentication

This command enables the DECbrouter 90 to send a TRAP message when it receives a packet with an incorrect community string.

The SNMP specification requires that a TRAP message be generated for each packet with an incorrect community string. However, because this action can result in a security breach, the router by default does not return a TRAP message when it receives an incorrect community string.

Establishing the TRAP Message Timeout

To define how often to try resending TRAP messages on the retransmission queue, use these global configuration commands:

snmp-server trap-timeout *seconds*
no snmp-server trap-timeout

The argument *seconds* sets the interval for resending the messages. The default is set to 30 seconds. The NO SNMP-SERVER TRAP-TIMEOUT command restores this default.

Example

This command sets an interval of 20 seconds to try resending TRAP messages on the retransmission queue:

```
snmp-server trap-timeout 20
```

Enabling SNMP System Shutdown Feature

Using SNMP packets, a network management tool can send messages to users on virtual terminals and on the DECbrouter 90 console. The network management tool operates similarly to the SNMP SEND command; however, the SNMP request that causes the message to be issued to the users also specifies the action to be taken after the message is delivered. One possible action is a shutdown request.

Requesting a *shutdown-after-message* is similar to issuing a SEND command followed by a RELOAD command. Because the ability to cause a reload from the network is a powerful feature, it is protected by this configuration command. To use this SNMP message reload feature, the device configuration must include the SNMP-SERVER SYSTEM-SHUTDOWN global configuration command. The full command syntax follows.

snmp-server system-shutdown
no snmp-server system-shutdown

The **no snmp-server system-shutdown** option prevents a SNMP system shutdown request (from an SNMP manager) from resetting the DECbrouter 90 agent.

Configuring the Trivial File Transfer Protocol (TFTP) Server

You can configure the DECbrouter 90 to act as a limited trivial file transfer protocol (TFTP) server from which other DECbrouter 90s can load their software. As a TFTP server host, the DECbrouter 90 responds to TFTP read request messages by sending a copy of its flash ROM software to the requesting host. The TFTP read request message must use the file name that you specified in the router configuration.

To specify TFTP server operation for a communication server, use the TFTP-SERVER SYSTEM global configuration command. The full syntax follows.

tftp-server system *filename ip-access-list*
no tftp-server system *filename ip-access-list*

This command has two arguments: *filename* and *list*. The argument *filename* is the name you give the communication server flash ROM file, and the argument ip-access-list is an *IP access-list* number.

The system sends a copy of the Flash ROM software to any host that issues a TFTP read request with this filename. To learn how to specify an access list, see *DECbrouter 90T Products, Configuration and Reference, Volume 2*, Chapter 5, "Routing IP."

You can specify multiple file names by repeating the TFTP-SERVER SYSTEM command. To remove a previously defined file name, use the NO TFTP-SERVER system command and append the appropriate file name and an access list number.

Example

This command causes the router to send, via TFTP, a copy of the flash ROM software when it receives a TFTP read request for the file *goodimage*. The requesting host is checked against access list 22.

```
tftp-server system goodimage 22
```

Tailoring Use of Network Services

The SERVICE global configuration command tailors the DECbrouter 90's use of network-based services. Some SERVICE commands also configure system defaults; see decimal-tty for an example. The full command syntax follows.

service *keyword*
no service *keyword*

The argument keyword is one of the following:

- **config**—Specifies TFTP autoloading of configuration files; disabled by default on systems with nonvolatile memory.
- **decimal-tty**—Specifies that line numbers be displayed and interpreted as decimal numbers rather than octal numbers; disabled by default.
- **finger**—Allows Finger protocol requests (defined in RFC 742) to be made of the router; enabled by default. This service is equivalent to issuing a remote SHOW USERS command.
- **tcp-keepalives-**{**in** | **out**}—Generates keepalive packets on idle network connections. The **in** keyword generates them on incoming connections (initiated by remote host); the **out** keyword generates them on outgoing connections (initiated by a user). There is a column in the EXEC **show tcp** display showing the keepalive statistics. The wakeups row shows how many keepalives have been transmitted without receiving any response (this is reset to 0 when a response is received).

The NO SERVICE command disables the specified service or function.

Example

The following command enables TFTP autoloading of configuration files:

service config

Redirecting System Error Messages

By default, the DECbrouter 90 sends the output from the EXEC command **debug** and system error messages to the console terminal.

To redirect these messages, as well as output from asynchronous events such as interface transition, to other destinations, use the LOGGING configuration command options.

These destinations include the console terminal, virtual terminals, and UNIX hosts running a syslog server; the syslog format is compatible with 4.3 BSD UNIX.

To configure message logging, you need to be in the configuration command collection mode. To enter this mode, use the EXEC command CONFIGURE at the EXEC prompt (see the section Entering Configuration Mode in Chapter 2 for the procedure). The following sections describe how to implement these redirection options.

Enabling Message Logging

To enable or disable message logging, use the following global configuration commands:

logging on
no logging on

The LOGGING ON command enables message logging to all supported destinations other than the console. This behavior is the default.

The NO LOGGING on command enables logging to the console terminal only.

Logging Messages to an Internal Buffer

The default logging device is the console; all messages are displayed on the console unless otherwise specified.

To log messages to an internal buffer, use the logging buffered global configuration command. The full command syntax follows.

logging buffered
no logging buffered

The LOGGING BUFFERED command copies logging messages to an internal buffer instead of writing them to the console terminal. The buffer is circular in nature, so newer messages overwrite older messages. To display the messages that are logged in the buffer, use the EXEC command SHOW LOGGING. The first message displayed is the oldest message in the buffer.

The NO LOGGING BUFFERED command cancels the use of the buffer and writes messages to the console terminal; this is the default.

Logging Messages to the Console

To limit how many messages are logged to the console, use the LOGGING CONSOLE global configuration command. The full syntax of this command follows.

logging console *level*
no logging console

The LOGGING CONSOLE command limits the logging messages displayed on the console terminal to messages with a level number at or below the specified severity level, which is specified by the *level* argument.

The argument *level* can be one of the keywords listed in Table 4–3. They are listed in order from the most severe to the least severe level.

Table 4–3 Logging Message Keywords and Levels

Level	Keyword	Description	Syslog Definition
0	**emergencies**	System is unusable	LOG_EMERG
1	**alerts**	Immediate action is needed	LOG_ALERT
2	**critical**	Critical conditions exist	LOG_CRIT
3	**errors**	Error conditions exist	LOG_ERR
4	**warnings**	Warning conditions exist	LOG_WARNING
5	**notification**	Normal, but significant, conditions exist	LOG_NOTICE
6	**informational**	Informational messages	LOG_INFO
7	**debugging**	Debugging messages	LOG_DEBUG

The default is to log messages to the console at the **warnings** level.

The NO LOGGING CONSOLE command disables logging to the console terminal.

Example

This command sets console logging of messages at the debug level:

```
!
logging console debug
!
```

Logging Messages to Another Monitor

To limit the level of messages to log to the terminal lines (monitors), use the LOGGING MONITOR command. The full syntax of this command follows.

logging monitor *level*
no logging monitor

The LOGGING MONITOR command limits the logging messages displayed on terminal lines other than the console line to messages with a level at or above level. The argument *level* is one of the keywords described for the LOGGING CONSOLE command in the previous section, "Logging Messages to the Console." To display logging messages on a terminal, use the privileged EXEC command TERMINAL MONITOR.

The NO LOGGING MONITOR command disables logging to terminal lines other than the console line.

This command sets the level of messages displayed on monitors other than the console to notifications:

```
!
logging monitor notifications
!
```

Logging Messages to a UNIX Syslog Server

To log messages to the syslog server host, use the LOGGING global configuration command. The full syntax is as follows:

logging *internet-address*
no logging *internet-address*

The LOGGING command identifies a syslog server host to receive logging messages. The argument *internet-address* is the Internet address of the host. By issuing this command more than once, you build a list of syslog servers that receive logging messages. The NO LOGGING command deletes the syslog server with the specified address from the list of syslogs.

Limiting Messages to a Syslog Server

To limit how many messages are sent to the syslog servers, use the LOGGING TRAP global configuration command. Its full syntax follows.

logging trap *level*
no logging trap

The LOGGING TRAP command limits the logging messages sent to syslog servers to messages with a level at or above *level*. The argument *level* is one of the keywords described for the LOGGING CONSOLE command in Table 4-3.

To send logging messages to a syslog server, specify its host address with the LOGGING command.

The default trap level is **informational**.

The NO LOGGING TRAP command disables logging to syslog servers.

The current software generates four categories of the syslog messages:

- Error messages about software or hardware malfunctions, displayed at the **errors** level.
- Output from the DEBUG commands, displayed at the debugging level.
- Interface up/down transitions and system restart messages, displayed at the **notifications** level.
- Reload requests and low-process stack messages, displayed at the **informational** level.

The EXEC command SHOW LOGGING displays the addresses and levels associated with the current logging setup. The command output also includes ancillary statistics.

Example

To set up the syslog daemon on a 4.3 BSD UNIX system, include a line such as the following in the file */etc/syslog.conf*:

```
local7.debug /usr/adm/logs/tiplog
```

The `local7` keyword specifies the logging facility to be used.

The `debug` argument specifies the syslog level. See the previous *level* arguments list for other arguments that can be listed.

The UNIX system sends messages at or below this level to the file specified in the next field. The file must already exist, and the syslog daemon must have permission to write to it.

Configuring Console and Virtual Terminal Lines

To configure your console and virtual terminal lines you need to be in the configuration command collection mode. To enter this mode, use the EXEC command CONFIGURE at the EXEC prompt (see the section Entering Configuration Mode in Chapter 2 for the procedure).

Starting Line Configuration

To start configuring a terminal line, use the LINE command. This command identifies a specific line for configuration and starts line configuration command collection.

The LINE command has the following syntax:

line *[type-keyword] first-line [last-line]*

This command can take up to three arguments: a keyword, a line number, or a range of line numbers.

The optional argument *type-keyword* specifies the type of line to be configured; it is one of the following keywords:

- **console**—Console terminal line
- **vty**—Virtual terminal for remote console access

When the line type is specified, the argument *first-line* is the relative number of the terminal line (or the first line in a contiguous group) you want to configure. Numbering begins with zero.

The optional argument *last-line* then is the relative number of the last line in a contiguous group you want to configure.

If you omit *type*, then *first-line* and *last-line* are absolute rather than relative line numbers. To display absolute line numbers, use the EXEC command SHOW USERS ALL.

The router displays an error message if you do not specify a line number.

Note

Line numbers, by default, are octal on the routers.

The LINE command enables you to easily configure a large group of lines all at once. After you set the defaults for the group, you can use additional LINE commands and subcommands to set special characteristics, such as location, for individual terminal lines.

Example

The following command starts configuration for the first five virtual terminal lines:

```
line vty 0 4
```

Establishing Line Passwords

When you start an EXEC on a line with password protection, the EXEC prompts for the password. If you enter the correct password, the EXEC prints its normal prompt. You can try to enter a password three times before the EXEC exits and returns the terminal to the idle state.

To specify a password, use the PASSWORD line subcommand. Its full syntax follows.

password *text*
no password

The *text* argument can contain any alphanumeric characters, including spaces, up to 80 characters. The password checking is case-sensitive. The password *Secret* is different from the password *secret*, for example, and the password *two words* is an acceptable password.

Example

This command sets the words "Big Easy" as the password on line 1:

```
!
line 1
password Big Easy
!
```

Setting Widths for International Character Sets for the Interface

These commands allow you to use graphical and international characters in banners and prompts, and to add special characters such as software flow control. Use these commands to set the character widths for a specific line to values other than the default values defined by the global commands described in the earlier section, Configuring the Global System Parameters. The decision to use the global configuration commands or line configuration subcommands depends on the types and numbers of terminals connected to the router, as follows:

- If a large number of connected terminals support nondefault ASCII bit settings, use the global configuration commands. (See the section Configuring the Global System Parameters earlier in the chapter for default settings.)
- If only a few of the connected terminals support nondefault ASCII bit settings, use the following line configuration subcommands or the EXEC **terminal exec-character-bits, terminal data-character-bits,** or TERMINAL SPECIAL-CHARACTER-BITS commands described in Chapter 3 to selectively configure the lines.

Use the following line subcommands to configure character widths on a per-line basis.

exec-character-bits {**8** | **7**}
special-character-bits {**8** | **7**}

The EXEC-CHARACTER-BITS command configures the character widths of EXEC and configuration command characters. The default value is 7 bits, which results in the use of a 7-bit ASCII character set. Configuring the EXEC character width to 8 allows you to add special graphical and international characters in

banners and prompts. Setting this value to 8 allows additional international and graphical characters in banner messages and prompts.

The SPECIAL-CHARACTER-BITS command configures the number of characters used in special characters such as software flow control and escape characters. The default special-character width is 7. Configuring the width to 8 bits allows you to use twice as many special characters as with the 7-bit setting.

Note

Setting the EXEC character width to 8 bits can cause failures. If a user on a terminal that is sending parity enters the command HELP, an "unrecognized command" message appears because the system is reading all 8 bits, although the eighth bit is not needed for the HELP command.

See the section Setting Default Widths for International Character Sets earlier in the chapter for global character-width commands.

Example

This example allows full 8-bit international character sets by default, except for the console, which is a dumb ASCII terminal. It illustrates use of the global configuration command and the line configuration subcommands.

```
default-value exec-character-bits 8
!
line 0
exec-character-bits 7
```

Establishing Connection Restrictions

To establish connection restrictions on the lines to some Internet addresses, use the ACCESS-CLASS line subcommand. The full command syntax follows.

access-class *list* {**in** | **out**}
no access-class *list* {**in** | **out**}

The ACCESS-CLASS subcommand restricts connections on a line or group of lines to certain Internet addresses. The argument *list* is an integer from 1 through 99 that identifies a specific access list of Internet addresses. The keyword **in** applies to incoming connections, such as virtual terminals. The keyword **out** applies to outgoing Telnet connections. The NO ACCESS-CLASS command removes access restrictions on the line for the specified connections.

Example

This example subcommand sets restrictions on access list 1 for outgoing Telnet connections:

```
access-class 1 out
```

See the *DECbrouter 90T Products, Configuration and Reference, Volume 2*, Chapter 5, "IP Routing" for information about configuring access lists.

Suppressing Banner Messages

By default, messages defined by the BANNER MOTD and BANNER EXEC commands are always displayed. This condition is defined by the EXEC-BANNER line subcommand. Its full syntax follows.

exec-banner
no exec-banner

To suppress display of a banner, enter the NO EXEC-BANNER command.

Example

These commands suppress the banner on virtual terminal lines 0 through 4:

```
line vty 0 4
no exec-banner
```

Turning On and Off the Vacant Banner

The router will display a message on the console when there is no active EXEC. This message, called the vacant message, is different from the banner message displayed when an EXEC process is activated.

To turn the vacant message banner on or off, use the VACANT-MESSAGE line configuration subcommands. The VACANT-MESSAGE command enables the banner to be displayed on the screen of an idle terminal. The full syntax of this command follows.

vacant-message *[c message c]*
no vacant-message

The VACANT-MESSAGE subcommand without any arguments causes the default message to be displayed. If you desire a banner, follow **vacant-message** with one or more blank spaces and a delimiting character *(c)* that you choose. Then type one or more lines of text *(message)*, terminating the text with the second occurrence of the delimiting character.

The NO VACANT-MESSAGE line configuration subcommand suppresses a banner message.

Example

This example will turn on the system banner and display a message:

```
line 0
vacant-message #
               Welcome to Digital Equipment Corp.
        This is the console terminal of the router Dross.
#
```

Note

You cannot use the delimiting character in the banner message.

Setting the Escape Character

The ESCAPE-CHARACTER line subcommand defines the escape character. The full syntax of this command is as follows:

escape-character *decimal-number*
no escape-character

The argument *decimal-number* is the ASCII decimal representation of the desired escape character or an escape character (Ctrl/P, for example). Typing the escape character followed by the X key returns you to the EXEC when you are connected to another computer. The default escape character is Ctrl/^. (See *DECbrouter 90 Products, Configuration and Reference, Volume 3,* Appendix E, "ASCII Character Set," for a list of ASCII characters.)

The operating software interprets break on the console as an attempt to halt the system.

Note

Depending upon the configuration register setting, console breaks either will be ignored or will cause the server to shut down. The Break key *cannot* be used as the escape character on the router.

The NO ESCAPE-CHARACTER line configuration subcommand reinstates the default escape character.

Example

This command changes the escape characters to Ctrl/P (ASCII character 17):

```
!
line 5
escape-character 17
!
```

Setting the Terminal Location

To set the location of the terminal, use the LOCATION line subcommand. The full syntax of this command follows.

location *text*
no location

This subcommand is for informational purposes only; it is not used by any aspects of the system software. The argument *text* is the desired description. The description appears in the output of the EXEC command SYSTAT. A maximum of 80 characters can be entered.

The NO LOCATION subcommand removes the information.

Example

This command describes the location of the terminal on line 2 as being Andrea's terminal:

```
!
line 2
location Andrea's terminal
!
```

Setting the EXEC Timeout Intervals

The EXEC command interpreter waits for a specified interval of time until the user starts input. If no input is detected, the EXEC resumes the current connection. If no connections exist, the EXEC returns the terminal to the idle state and disconnects the incoming session.

To set this interval, use the EXEC-TIMEOUT line configuration subcommand. The full syntax of the command follows.

exec-timeout *minutes [seconds]*
no exec-timeout

The argument *minutes* is the number of minutes, and the optional argument *seconds* specifies additional interval time in seconds. The default interval is ten minutes; an interval of zero specifies no timeouts.

The NO EXEC-TIMEOUT subcommand removes the timeout definition. It is the same as entering EXEC-TIMEOUT 0.

Examples

This command sets an interval of 2 minutes, 30 seconds:

```
exec-timeout 2 30
```

This command sets an interval of 10 seconds:

```
exec-timeout 0 10
```

Setting the Screen Length

To set the terminal screen length, use the LENGTH line configuration subcommand. The full syntax of the command follows.

length *screen-length*
no length

The argument *screen-length* is the number of lines on the screen. The router uses this value to determine when to pause during multiple-screen output. The default length is 24 lines. A value of zero disables pausing between screens of output.

The NO LENGTH command is the same as entering the command LENGTH 0.

Note

Not all commands pay attention to the configured screen length. For example, the SHOW TERMINAL command assumes a screen length of 24 lines or more.

Setting Notification

To enable the terminal to notify the user about pending output, use the NOTIFY line subcommand. The full syntax of the command follows.

notify
no notify

The NOTIFY subcommand sets a line to inform a user who has multiple, concurrent Telnet connections when output is pending on a connection other than the current connection.

The NO NOTIFY line configuration subcommand ends notification and is the default.

Setting Character Padding

To set the padding on characters, use the PADDING line configuration subcommand. The full syntax of the command follows.

padding *decimal-number count*
no padding *decimal-number*

The PADDING subcommand sets padding for a specified output character. The argument *decimal-number* is the ASCII decimal representation of the character, and the argument *count* is the number of NUL bytes sent after that character.

The NO PADDING line configuration subcommand removes padding for the specified output character.

The following command pads Return (ASCII character 13) with 25 NUL bytes:

```
padding 13 25
```

Global System Configuration Command Summary

This section lists all of the global system configuration commands in alphabetical order.

[**no**] **banner** {**motd** | **exec** | **incoming**} *c text c*

Displays the message that the EXEC command interpreter displays whenever a user starts any EXEC process or activates a line. The **motd, exec,** and **incoming** keywords control when the banner message is displayed. The argument c specifies a delimiting character of your choice. The argument *text* specifies the message to be shown on the screen whenever an interface line is activated.

[**no**] **boot buffersize** *bytes*

Specifies the size of the buffer to be used for netbooting a host or a network configuration file. The argument *bytes* by default is the size of your nonvolatile memory (32 kilobytes). There is no minimum or maximum size that can be specified.

[**no**] **boot host** *filename [address]*

Specifies the host configuration file name. The argument *filename* is the new name for the host configuration file. If you omit the argument *address*, the DECbrouter 90 uses the default broadcast address of 255.255.255.255. The optional argument *address* allows you to specify a specific network host or a subnet broadcast address.

[**no**] **boot network** *filename [address]*

Specifies the network configuration file name. The argument *filename* is the new name for the network configuration file. If you omit the optional argument *address*, the DECbrouter 90 uses the default broadcast address of 255.255.255.255. The optional argument *address* allows you to specify a specific network host or a subnet broadcast address.

[**no**] **buffers** {**small** | **middle** | **big** | **large** | **huge**} {**permanent** | **max-free** | **min-free** | **initial**} *number*

Allows a network administrator to adjust initial buffer pool settings and set limits at which temporary buffers are created and destroyed. The first keyword denotes the size of buffers in the pool; the default number of the buffers in a pool is determined by the hardware configuration. The second keyword specifies the buffer management parameter to be changed, as follows:

- **permanent**—The number of permanent buffers that the system tries to allocate.
- **max-free**—The maximum number of free or unallocated buffers in a buffer pool.
- **min-free**—The minimum number of free or unallocated buffers in a buffer pool.
- **initial**—The number of additional temporary buffers that should be allocated when the system is reloaded.

The argument *number* specifies the number of buffers to be allocated.

The NO BUFFERS command with appropriate keywords and arguments restores the default buffer values.

[**no**] **buffers huge size** *number*

Dynamically resizes all huge buffers to the value that you supply. The buffer size cannot be lowered below the default. The argument *number* specifies the number of buffers to be allocated. The **no** version of the command with the keyword and argument restores the default buffer values.

copy flash tftp

Copies a flash TFTP image back to a TFTP server.

copy tftp flash

Copies a TFTP image into the current flash configuration.

copy mop flash

Copies a MOP image into the current flash configuration.

default-value exec-character-bits {**8** | **7**}

Configures the character widths of EXEC and configuration command characters. The default value is 7 bits, which results in the use of a 7-bit ASCII character set. Configuring the EXEC character width to 8 bits allows you to add special graphical and international characters in banners and prompts.

default-value special-character-bits {**8** | **7**}

Configures the number of characters used in special characters such as software flow control, hold, escape, and disconnect characters. The default special-character width is 7. Configuring the width to 8 allows you to use twice as many special characters as with the 7-bit setting.

enable

Allows you to enter the privileged command level.

enable password *password*

Assigns a password for the privileged command level. The argument *password* is case-sensitive and specifies the password prompted for in response to the EXEC command ENABLE.

[**no**] **enable last-resort** {**succeed** | **password**}

Allows you to specify what happens if the TACACS servers used by the ENABLE command do not respond. The default action is to fail. The keywords change this default action:

- **succeed**—Allows you to enable without further question.

- **password**—Allows you to enable by entering the privileged command level.

The **no** version of the command restores the default.

[**no**] **enable use-tacacs**

Enables or disables use of TACACS to check the user ID and password supplied to the EXEC ENABLE command.

hostname *name*

Specifies the name for the router. The default is *Router*.

[**no**] **logging** *internet-address*

Identifies a syslog server host to receive logging messages. The argument *internet-address* is the Internet address of the host.The NO LOGGING command deletes the syslog server with the specified address from the list of syslogs.

[**no**] **logging buffered**

Copies logging messages to an internal buffer instead of writing them to the console. The **no** version of the command cancels this behavior and writes messages to the console terminal; this is the default.

[**no**] **logging console** *level*

Limits the logging of messages displayed on the console terminal to messages with a level at or above the specified severity, which is specified by the *level* argument. The argument *level* can be one of the following keywords, listed here in order from the most severe to the least severe level.

- **emergencies**—System unusable
- **alerts**—Immediate action needed
- **critical**—Critical conditions
- **errors**—Error conditions
- **warnings**—Warning conditions (default)
- **notification**—Normal but significant conditions
- **informational**—Informational messages only
- **debugging**—Debugging messages

The NO LOGGING CONSOLE command disables logging to the console terminal.

[**no**] **logging monitor** *level*

Limits the logging messages displayed on terminal lines other than the console line to messages with a level at or above *level*. The argument *level* is one of the keywords described for the LOGGING CONSOLE command. The NO LOGGING MONITOR command disables logging to terminal lines other than the console line.

[**no**] **logging on**

Enables or disables message logging to all supported destinations except the console. Enabled message logging is the default.

[**no**] **logging trap** *level*

Limits the logging messages sent to syslog servers to messages with a level at or above *level*. The argument *level* is one of the keywords described for the LOGGING CONSOLE command.

[**no**] **service** *keyword*

Tailors the router's use of network-based services. The argument *keyword* is one of the following:

- **config**—Specifies TFTP autoloading of configuration files; disabled by default on system with nonvolatile memory.
- **decimal-tty**—Specifies that line numbers be displayed and interpreted as decimal numbers rather than octal numbers; disabled by default.
- **finger**—Allows finger protocol requests (defined in RFC 742) to be made of the router; enabled by default.
- **password-encryption**—Displays passwords in encrypted form when a SHOW CONFIG command is issued.
- **tcp-keepalives-**{**in** | **out**}—Generates keepalive packets on idle network connections. The **in** keyword generates them on incoming connections; the **out** keyword generates them on outgoing connections.

The NO SERVICE command disables the specified service or function.

no snmp-server

Disables the SNMP operations.

[**no**] **snmp-server access-list** *list*

Sets up an access list that determines which hosts can send requests to the DECbrouter 90. The argument list is an integer from 1 through 99 that specifies an IP access list.

[**no**] **snmp-server community** *string* [**RO** | **RW**] [*list*]

Enables or disables SNMP server operation on the DECbrouter 90. The argument *string* specifies a community string that acts like a password and permits access to the SNMP protocol.

[**no**] **snmp-server host** *address community-string* [**snmp | tty**]

Specifies which host or hosts should receive TRAP messages. Issue the SNMP-SERVER HOST command once for each host acting as a TRAP recipient. The argument *address* is the name or Internet address of the host. The argument *community-string* is the password-like community string set with the SNMP-SERVER COMMUNITY command. These optional keywords direct the TRAP:

- **snmp**—Causes all SNMP-type TRAP messages to be sent and starts the DECbrouter 90-specific RELOAD TRAP message.
- **tty**—Causes TCP connection TRAP messages to be included.

[**no**] **snmp-server packetsize** *bytes*

Sets or removes control over the largest SNMP packet size permitted when the SNMP server is receiving a request or generating a reply. The argument bytes is a byte count from 484 through 8192. The default is 484.

[**no**] **snmp-server queue-length** *length*

Defines the length of the message queue for each TRAP host. The argument *length* is the number of TRAP events that can be held before the queue must be emptied; the default is ten. The **no** version of the command resets the queue length to the default.

[**no**] **snmp-server system-shutdown**

Allows or restricts use of the SNMP message reload feature and prevents an SNMP system shutdown request from resetting the DECbrouter 90 agent.

[**no**] **snmp-server trap-authentication**

Allows or restricts the DECbrouter 90 from sending a TRAP message when it receives a packet with an incorrect community string.

[**no**] **snmp-server trap-timeout** *seconds*

Defines how often to try resending TRAP messages on the retransmission queue. The argument *seconds* sets the interval for resending the messages. The default is set to 30 seconds. The **no** form of the command restores the default.

[**no**] **tacacs-server attempts** *count*

Controls the number of login attempts that can be made on a line set up for TACACS verification. The argument *count* is the number of attempts. The default is three attempts.

[**no**] **tacacs-server authenticate** {**connect** | **enable**}

Specifies that a response is required from the network or communication server to indicate whether the user can perform the indicated action. Actions that require a response include the following:

- **connect**—User makes TCP connections.
- **enable**—User enters the ENABLE command.

The **no** form of the command disables the action.

[**no**] **tacacs-server extended**

Enables or disables an extended TACACS mode. This mode provides information about the terminal requests for use in setting up UNIX auditing trails and accounting files for tracking use of terminal servers and routers.

[**no**] **tacacs-server host** *name*

Specifies a TACACS host. The argument name is the name or Internet address of the host. You can use multiple commands to specify multiple hosts.

[**no**] **tacacs-server last-resort** {**password** | **succeed**}

Causes the DECbrouter 90 to request the privileged password as verification, or forces successful login without further input from the user, depending upon the keyword specified.

- **password**—Allows the user to access the privileged-level command mode by entering the password set by the ENABLE command.
- **succeed**—Allows the user to access the privileged-level command mode without further question.

The **no** form of the command disables the action.

[**no**] **tacacs-server notify** {**connect** | **enable** | **logout**}

Causes a message to be transmitted to the TACACS server, with retransmission being performed by a background process for up to five minutes. The keywords specify notification of the TACACS server whenever a user does one of the following:

- **connect**—Makes TCP connections.
- **enable**—Enters the ENABLE command.
- **logout**—Logs out.

The **no** form of the command disables the messages.

tacacs-server optional-passwords

Specifies that the first TACACS request to a TACACS server is made without password verification. Supports all TACACS requests—login, SLIP and enable.

[**no**] **tacacs-server retransmit** *retries*

Specifies the number of times the server will search the list of TACACS server hosts before giving up. The argument *retries* is the retransmit count. The default is two retries. The **no** form of the command restores the default.

[**no**] **tacacs-server timeout** *seconds*

Sets the interval the server waits for a server host to reply. The argument *seconds* specifies the number of seconds. The default interval is five seconds. The **no** form of the command restores the default.

[**no**] **tftp-server system** *filename ip-access-list*

Specifies or removes TFTP server operation for the DECbrouter 90. The argument *filename* is the name given to the router flash ROM file, and the argument *access-list* is an IP access list number.

username *name*[**nopassword** | **password** *encryptiontype password*]
username *name* [**accesslist** *number*]
username *name* [**autocommand** *command*]
username *name* [**noescape**] [**nohangup**]

Creates special-case user name-based authentications. Multiple USERNAME commands can be used to specify options for a single user.

The **nopassword** keyword means that no password is required for this user to log in. This is usually most useful in combination with the **autocommand** keyword.

The **password** keyword specifies a possibly encrypted password for this user name.

The *encryptiontype* argument is a single-digit number. Currently defined encryption types are 0, which means no encryption, and 7, which specifies a Digital-defined encryption algorithm. Passwords entered unencrypted are written out with the Digital encryption. Passwords can contain embedded spaces and must be the last option specified in the USERNAME command.

The **accesslist** keyword specifies an outgoing access list that overrides the access list specified in the ACCESS CLASS line configuration subcommand. It is used for the duration of the user's session. The access list number is specified by the *number* argument.

The **autocommand** keyword causes the command specified by the *command* argument to be issued automatically after the user logs in. When the command is complete, the session is terminated. As the command can be any length and contain embedded spaces, commands using the **autocommand** keyword must be the last option on the line.

The **noescape** keyword prevents a user from using an escape character on the host to which he is connected.

The **nohangup** keyword prevents the DECbrouter 90 from disconnecting the user after an automatic command (set up with the **autocommand** keyword) has completed. Another login prompt is provided to the user.

Line Configuration Subcommand Summary

This section contains a summary of all the line configuration subcommands in alphabetical order.

[**no**] **access-class** *list* {**in | out**}

Restricts or permits connections on a line or group of lines to certain Internet addresses. The argument list is an integer from 1 through 99 that identifies a specific access list of Internet addresses. The keyword **in** applies to incoming connections; the keyword **out** applies to outgoing Telnet connections.

[**no**] **escape-character** *decimal-number*

Sets or removes the escape character on the specified line. The argument *decimal-number* is either the ASCII decimal representation of the character or a control sequence (Ctrl/E, for example). The default escape character is Ctrl/^.

[**no**] **exec-banner**

Enables or disables a banner. By default, a banner is displayed on the console. To suppress display of a banner, enter the **no** variation of the command.

exec-character-bits {**8** | **7**}

Configures the character widths of EXEC and configuration command characters. The default value is 7 bits, which results in the use of a 7-bit ASCII character set. Configuring the EXEC character width to 8 allows you to add special graphical and international characters in banners and prompts.

[**no**] **exec-timeout** *minutes [seconds]*

Sets the interval the EXEC waits for user input before resuming the current connection, or if no connections exist, before returning the terminal to the idle state and disconnecting the incoming session. The argument *minutes* is the number of minutes, and the optional argument *seconds* specifies additional interval time in seconds. The default interval is ten minutes; an interval of zero specifies no timeouts. The **no** form of the command restores the default.

[**no**] **length** *screen-length*

Sets the terminal screen length. The argument *screen-length* is the number of lines on the screen. The DECbrouter 90 uses this value to determine when to pause during multiple-screen output. The default length is 24 lines. A value of 0 (zero) or the **no** form of the command disables pausing between screens of output.

line [*type-keyword*] *first-line* [*last-line*]

Identifies a specific line for configuration and starts line configuration command collection. The optional argument *type-keyword* specifies the type of line to be configured; it can be **console, aux,** or **vty**. When the line type is specified, the argument first-line is the relative number of the terminal line (or the first line in a contiguous group) you want to configure. The optional argument *last-line* is the relative number of the last line in a contiguous group you want to configure. If you omit *type*, then *first-line* and *last-line* are absolute rather than relative line numbers.

line aux *port-address*

This variation of the LINE command configures the auxiliary port. When configuring the auxiliary port, address it as line 0.

[**no**] **location** *text*

Enters or removes information-only data about the terminal location and/or status. The argument *text* is the desired description.

[**no**] **login**

Enables or disables password-checking for the password specified by the PASSWORD command.

[**no**] **login tacacs**

Causes the TACACS user ID and password checking mechanism to be used instead of the regular password checking. The **no** form of the command disables this mechanism.

[**no**] **notify**

Enables or disables line notification of a user who has multiple, concurrent Telnet connections when output is pending on a connection other than the current line.

[**no**] **padding** *decimal-number count*

Sets or unsets padding for a specified output character. The argument *decimal-number* is the ASCII decimal representation of the character; and the argument *count* is the number of NUL bytes sent after that character.

[**no**] **password** *text*

Specifies a password. The text argument specifies a password. It can contain any alphanumeric characters, including spaces, up to 80 characters. The **no** version of the command removes the specified password.

[**no**] **service password-encryption**

Controls whether privileged command and line passwords are encrypted. The **no** version of the command turns off password encryption.

special-character-bits {**8** | **7**}

Configures the number of characters used in special characters such as software flow control and escape characters. The default special-character width is 7. Configuring the width to 8 bits allows you to use twice as many special characters as with the 7-bit setting.

[**no**] **vacant-message** [*c message c*]

Controls whether or not a banner is displayed on the screen of an idle terminal. The command without any arguments causes the default message to be displayed. The NO VACANT-MESSAGE command suppresses a banner message. To display a banner, follow **vacant-message** with one or more blank spaces and a delimiting character (*c*) that you choose, then type one

or more lines of text (*message*), and terminate the text with the second occurrence of the delimiting character.

5
Using MOP

This chapter provides information about how to use Digital Equipment Corporation's Maintenance Operation Protocol (MOP). It includes the following topics:

- Overview of MOP and configuring MOP on your router
- Using MOP to copy an image into Flash memory and to boot from a MOP boot server
- Network configurator support and remote system management
- Debugging MOP

There is a command summary at the end of the chapter.

Note

The device referred to as a "MOP boot server" in this chapter can be any device running a Digital operating system that supports MOP. This includes a range of devices, from a VMS timesharing system to an Infoserver 150.

MOP Overview

Digital Equipment Corporation's MOP is supported on Ethernet and serial interfaces. The DECbrouter 90 supports the following MOP features:

- Periodic ID messages (MOP SYSID) The router periodically broadcasts "hello" messages that are used by network management tools to produce network maps.
- System software loading Loads system software from a MOP boot server over the network.
- Flash memory loading Copies system software updates to Flash memory over the network.
- Remote router management Establishes a connection from a MOP boot server to a virtual port on the router to re- configure or examine the state of the router.

MOP was designed to be used for basic management of network components. It is a special-purpose protocol that does not support many of the features supported by other protocols.

MOP can only be used between systems connected to the same local area network or between systems connected by transparent bridging. It cannot be routed, because it does not have a network layer. MOP does not store connection context information, so only one MOP function can be active on an interface. For example, only one MOP virtual terminal session can be supported at one time.

MOP references a system by its MAC address, which can be displayed by using the SHOW INTERFACE command. When referring to a remote system, you may have to specify both an interface and a MAC address.

Note

If you bridge MOP and use filters, note that all MOP file-loading operations use the protocol type 6001 and the multicast address AB00.0001.0000. All other MOP opera- tions use the protocol type 6002 and the multicast address AB00.0002.0000.

Configuring MOP

This section describes how to configure MOP using global and interface configuration commands.

Enabling MOP for an Interface

To enable MOP on an interface, use the following interface configuration subcommand:

mop enabled
no mop enabled

This command enables all MOP operations on an interface. MOP is enabled by default on Ethernet interfaces and disabled on all other interfaces. Use the NO MOP ENABLED command to disable MOP.

Example

The following example enables MOP on interface serial 0:

```
interface serial 0
mop enabled
```

Sending MOP System ID Messages

If MOP is enabled on an interface, it will transmit a system ID message every 8 to 12 minutes by default. This message is used by various network management tools. Use the following interface subcommand to configure your router to send system ID messages:

mop sysid
no mop sysid

If you do not use the network management tools and are concerned about network traffic levels (for example, over slow serial lines), use the NO MOP SYSID command to disable system ID messages. With system ID messages disabled, you can use MOP but the router will not generate system ID messages.

Example

The following configuration disables MOP system ID messages on interface serial 0:

```
interface serial 0
mop enabled
no mop sysid
```

Network Configurator Support

MOP includes network management software called the network configurator. MOP maintains protocol messages that the network configurator uses to generate a network map, which is a list of the stations operating on a LAN. Every 8 to 12 minutes, a system that supports MOP sends out a system ID message informing management stations of its existence.

Within the system ID message, there is a *communications device code* that informs the management station of the type of device on the network. The DECbrouter 90T uses a MOP communications device code of 121 (decimal). (This differs from other Digital products.)

Setting Retransmit Timer Interval

By default, when the router transmits a request that requires a response from a MOP boot server and the server does not receive a response, the message will be retransmitted after four seconds. If the MOP boot server and router are separated by a slow serial link, it may take longer than four seconds for the router to receive a response to its message. Therefore, you may want to configure the router to wait longer than four seconds before retransmitting the message if you are using such a line.

Use the following command to change the length of time the router waits before retransmitting a message:

mop retransmit-timer *seconds*
no mop retransmit-timer *seconds*

The argument *seconds* is a number from 1 through 20. The default value is 4.

Example

In the following example, if the MOP boot server does not respond within 10 seconds after the router sends a message, the server will retransmit the message.

```
interface serial 0
mop enabled
mop retransmit-timer 10
```

Setting the Maximum Number of Retransmissions

By default, the router will retransmit a request up to eight times before declaring a failure. If you have noisy or congested lines with very high drop rates, you may want to configure the router to retransmit messages more than eight times.

Use the following command to alter the number of times a router will retransmit a message:

mop retries *count*
no mop retries *count*

The argument *count* must be a number from 3 through 24. The default value is 8.

Example

In the following example, the router will attempt to retransmit a message to an unresponsive host 11 times before declaring a failure.

```
interface serial 0
mop enabled
mop retries 11
```

Copying a MOP Image to Flash Memory

Your DECbrouter 90T comes from the factory pre-loaded with the latest router software. However, when you receive updates, you will need to load the software into the Flash memory of your router. Use the following command to use MOP to copy a system image into Flash memory (note that MOP must be enabled on the relevant interfaces):

copy mop flash

When you enter the COPY MOP FLASH command, the system prompts you for the filename of the software image. You are then asked for a confirmation. You can terminate the command at this point by entering **n**. If you confirm, and the Flash memory has enough available space, you have the option of erasing any existing files before writing into Flash memory. If no free space is available, or if files have never been written to Flash memory, you must erase Flash memory before copying the MOP image.

You do not need to specify the address of a MOP server; the system automatically solicits a MOP boot server for the specified file by sending a multicast file request message.

Note

The router must be in bootstrap mode before you may execute the COPY MOP FLASH command. In bootstrap mode, the router runs out of conventional ROM, permitting the Flash to be erased. The router is in bootstrap mode when the EXEC prompt ends in (boot)> (for example, Router(boot)>). For instructions on placing the router in bootstrap mode, please see page 4-10 in the *DECbrouter 90 Configuration and Reference, Volume 1* or the *DECbrouter 90 Getting Started* guide, the section Setting the Configuration Register.

Examples

The following example shows the sample output of the prompts you will see when you enter the COPY MOP FLASH command. In this example, the system image xx-k is being copied to Flash memory, another file of the same name is already in Flash and there is enough memory to copy it without erasing any existing files.

```
George(boot)# copy mop flash
Flash file directory:
File  name/status

0  xx-k

[2264000/4194304 bytes free/total]

Name of file to copy ? xx-k
File xx-k already exists; it will be invalidated!
Copy xx-k from MOP server into flash memory ?  [confirm]
2263936 bytes available for writing without erasure.

This line appears only if the same file already exists in Flash memory
```

In the following example, the Flash memory has no space available for the MOP file.

```
George(boot)# copy mop flash
Flash is filled to capacity.
Erasure is needed before flash may be written.
Erase flash before writing? [confirm]
```

If Flash is filled to capacity, the command will terminate. If you enter a y, the erase process will start. If Flash is not filled to capacity, and you enter **n** after the "Erase flash before writing?" prompt, the copy process will continue.

Note

Make sure you have enough Flash memory space before entering **n** at the "Erase flash before writing?" prompt. The Flash directory information displayed after you enter the COPY MOP FLASH command and before the prompts indicates the amount of free and total space in Flash memory.

Erasing Flash Memory

When you enter commands to erase Flash memory, the system erases every bank of Flash memory. For each bank of Flash memory, an e is displayed at periodic intervals to indicate that the erase process is in progress. The entire erase process takes several tens of seconds, depending on the total amount of Flash memory.

The filename can be in either lowercase or uppercase. If the same file is copied twice to Flash without an intermediate erase, and there is enough available Flash memory to hold the two files, the first file will be invalidated and the second file will become the valid file.

Example

The following example shows sample commands and output for copying a system image named XX-K into a Flash memory that already contains a file named XX-K. In this example, Flash memory is erased before copying the file XX-K into Flash memory.

```
George(boot)# copy mop flash

Flash file directory:
File  name/status
 0  xx-k
 [2264000/4194304 bytes free/total]

Name of file to copy ? XX-K
File XX-K already exists; it will be invalidated!
Copy XX-K from MOP server into flash memory ?  [confirm]
2263936 bytes available for writing without erasure.
Erase flash before writing? [confirm]
Flash erase: erasing bank 0 .. eeeeeeeeeeeeeee

Mop2flash: Loading XX-K into flash from aa00.0400.9005 on Ethernet0 :
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!
Verify
checksum...vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
vvvvvvvvvvvvvvvvv
Verification successful:Length = 1930240, checksum = 0x3461
```

The following example shows sample commands and output for copying a system image named xx-k into Flash memory that already contains an image named XX-K. In this example, Flash memory is not erased before loading the file:

```
George(boot)# copy mop flash
Flash file directory:
File  name/status
 0  XX-K

Name of file to copy ? xx-k
File xx-k already exists; it will be invalidated!
Copy xx-k from MOP server into flash memory ?  [confirm]
2263936 bytes available for writing without erasure.
Erase flash before writing? [confirm]n
Mop2flash: Loading xx-k into flash from aa00.0400.9005 on Ethernet0 :
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!
Verify
checksum...vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
vvvvvvvvvvvvvvvvvv

Verification successful:Length = 1930240, checksum = 0x3461
```

Note that the MOP boot server address (aa00.0400.9005) was obtained automatically. The series of ! marks indicates that the copying process is active over the network. The series of v marks indicate that a checksum verification of the file written to Flash memory is taking place.

The following example shows sample commands and output for copying a system image named xx-kf into Flash memory that already contains two images.

```
George(boot)# copy mop flash
Flash file directory:
File  name/status
 0  XX-K [deleted]
 1  xx-k
 [333696/4194304 bytes free/total]

Name of file to copy ? xx-kf
Copy xx-kf from MOP server into flash memory ?  [confirm]
333632 bytes available for writing without erasure.
Erase flash before writing? [confirm]n
Mop2flash: Loading xx-kf into flash from aa00.0400.9005 on Ethernet0 :
!!!!!!!!!!!!!!Flash: final address 0x3400240 beyond flash range of
0x3400000.

Error programming flash memory.
```

The following example shows sample commands and output for copying a system image named xx-kf into the Flash memory that is already filled to capacity. In this example the image is not copied into Flash memory.

```
George(boot)# copy mop flash
Flash file directory:
File  name/status
 0  XX-K [deleted]
 1  xx-k
 2  xx-kf [deleted]
 [0/4194304 bytes free/total]

Name of file to copy ? xx-kf
Copy xx-kf from MOP server into flash memory ?  [confirm]
Flash is filled to capacity.
Erasure is needed before flash may be written.
Erase flash before writing? [confirm]n
George#
```

Aborting the Copy Process

To abort the copy process, press the Ctrl, Shift, and 6 keys simultaneously. The process will stop, but the partial file that was copied before the abort command was issued will remain in Flash memory until it is erased.

When an image has been successfully copied into Flash, the commands SHOW FLASH and SHOW FLASH ALL will provide a list of files present in Flash memory.

For information about booting from Flash memory, see the section Automatically Booting from Flash Memory in Chapter 4.

Remote Router Management

Using MOP on your device, you can create a virtual terminal connection to a router. This allows you to execute all router commands as if your device were connected to the router console.

To support remote management, set a password on the incoming virtual terminal (VTY) lines or enable TACACS on the line. For more information, see Establishing Passwords and System Security in Chapter 4.

To initiate a MOP session, you must the use services of the operating system running on your boot server. For the VMS operating system, use the CONNECT NODE or CONNECT VIA commands under the NCP program. Consult your operating system documentation for more information.

Note

MOP is not designed to be used as a general-purpose, remote terminal protocol. It places considerable load on both the host and router and should only be used for remote system management.

Example

The following example shows a connection being established from a VMS system to a router.

```
$ RUN SYS$SYSTEM:NCP
NCP> CONN VIA SVA-0 PHYSICAL ADDRESS 00-00-0C-04-05-06
Console connected (press CTRL/D when finished)
User Access Verification
Password:
chaos> show hardware
DECbrouter 90 Software (XX-K), Version 9.14(1)
Copyright (c) 1986-1992 by cisco Systems, Inc.
Compiled Fri 06-Apr-93 17:09

System Bootstrap, Version 4.6(1), SOFTWARE

pisa uptime is 1 day, 16 hours, 57 minutes
System restarted by reload
System image file is "dit/xx-k.91s", booted via flash

DECbrouter 90 (68030) processor (revision 0xA0) with 1024K/1024K bytes
of memory.
Processor board serial number 00000076
DDN X.25 software.
Bridging software.
1 Ethernet/IEEE 802.3 interface.
1 Serial network interface.
32K bytes of non-volatile configuration memory.
4096K bytes of flash memory sized on embedded flash.
Configuration register is 0x0
chaos> ^d
NCP> EXIT
```

Debugging MOP

The EXEC command DEBUG MOP reports events occurring on the MOP server, including request ID messages received, system ID messages transmitted, and the reservation and release of the remote console. This command can be used to debug problems encountered when attempting to connect to a router using MOP.

Example

The following example shows debug output while a user is connecting to a router using MOP.

```
muddy-river# debug mop
MOP event debugging is on
muddy-river#
MOP: Reserving console for aa00.0400.9005
MOP(Ethernet0): Got request_id message from aa00.0400.9005
MOP(Ethernet0): Sending sysid message to aa00.0400.9005
MOP: Console released by aa00.0400.9005
```

Command Summary

This section lists the commands described in this chapter. Unless otherwise specified, these are configuration commands.

b mop *filename* [*MAC-address*] [*interface*]
ROM monitor command that boots a system image using MOP. The argument *filename* is the name of the file image that is being loaded. The optional argument *MAC-address* is the MAC address of the host from which you are loading the software. The optional argument *interface* selects the interface from which to load the software.

copy mop flash
Copies a system image that has been stored on a Digital MOP server into Flash memory. Note: The DECbrouter 90T *must* be in bootstrap mode in order to load the flash.
This is an EXEC command.

debug mop
Enables the output of MOP debugging messages. Used for troubleshooting.
This is an EXEC command.

[**no**] **mop enabled**
Enables MOP on an interface. MOP is enabled by default on Ethernet interfaces and is not enabled on other interfaces. Use the NO MOP ENABLED command to disable MOP. Once enabled, the MOP server periodically multicasts a system ID message.

[**no**] **mop retransmit-timer** *seconds*
Sets the time that the router will wait for a response before retransmitting a message that has not yet been acknowledged. The argument *seconds* is the number of seconds to wait and is a number from 1 through 20. The default value is 4.

[**no**] **mop retries** *count*
Sets the number of times the router will attempt to retransmit an unacknowledged message before declaring a failure. The argument *count* must be a number from 3 through 24 (inclusive). The default value is 8.

[**no**] **mop sysid**
Configures an interface for periodic MOP system ID messages. Use the NO MOP SYSID command to disable MOP from sending the system ID messages. This lets you use MOP, but does not generate messages used by the MOP network configurator.

6

Managing and Monitoring the System

This chapter describes the EXEC commands you use to monitor, manage, and troubleshoot general system processes and conditions on your DECbrouter 90. These processes and conditions include:

- System memory allocator and buffer pools
- System configuration and processes
- Stack utilization
- System error message logs

This chapter also provides a general overview of these system tasks:

- Using the DEBUG commands to troubleshoot network problems
- Testing connectivity
- Tracing routes
- Testing the system

See the chapters containing information about the interfaces and the protocols supported by the DECbrouter 90 for descriptions of the interface-specific and protocol-specific debugging and monitoring commands.

Most of the network management commands are executed at the privileged-level prompt, although there is a subset of monitoring commands that you can enter at the user-level prompt.

At the end of the chapter is an alphabetically arranged summary of the commands it describes.

Monitoring System Processes

Use the EXEC show commands to display data structures, configuration parameters, and usage statistics for the router. To list all the available SHOW command options, enter this command at the EXEC prompt.

```
show ?
```

Two different lists will be displayed, one at the user-level prompt, and one at the enabled, privileged-level prompt. The lists include a summary of the command function for easy reference. See Chapter 2 for the procedure to enter the user and privileged levels.

Displaying Buffer Pool Statistics

The DECbrouter 90 has one pool of queuing elements and five pools of packet buffers of different sizes. For each pool, the router keeps counts of the number of buffers outstanding, the number of buffers in the free list, and the maximum number of buffers allowed in the free list. To display statistics for the buffer pools on the router, use the SHOW BUFFERS command. Enter this command at the EXEC prompt:

show buffers [*interface*]

The optional argument *interface* causes a search of all buffers that have been associated with that interface for longer than one minute. The contents of these buffers are printed to the screen. This option is useful in diagnosing problems where the input queue count on an interface is consistently nonzero.

Following is sample output without the optional interface argument. Table 6–1 describes the fields.

```
Buffer elements:
     250 in free list (250 max allowed)
     10816 hits, 0 misses, 0 created
Small buffers, 104 bytes (total 120, permanent 120):
     120 in free list (0 min, 250 max allowed)
     26665 hits, 0 misses, 0 trims, 0 created
Middle buffers, 600 bytes (total 90, permanent 90):
     90 in free list (0 min, 200 max allowed)
     5468 hits, 0 misses, 0 trims, 0 created
Big buffers, 1524 bytes (total 90, permanent 90):
     90 in free list (0 min, 300 max allowed)
     1447 hits, 0 misses, 0 trims, 0 created
Large buffers, 5024 bytes (total 0, permanent 0):
     0 in free list (0 min, 100 max allowed)
     0 hits, 0 misses, 0 trims, 0 created
Huge buffers, 12024 bytes (total 0, permanent 0):
     0 in free list (0 min, 30 max allowed)
    0 hits, 0 misses, 0 trims, 0 created

0 failures (0 no memory)
```

Table 6–1 Show Buffers Field Descriptions

Field	Description
Buffer elements	Blocks of memory used in internal operating system queues
Small buffers	
Middle buffers	
Big buffers	Blocks of memory used to hold network packets
Large buffers	
Huge buffers	
Hits	Count of successful attempts to allocate a buffer when needed
Misses	Count of allocation attempts that failed for lack of a free buffer in the pool
Created	Count of new buffers created
Trims	Count of buffers destroyed
In free list	Number of buffers of a given type that are not currently allocated and are available for use
Max allowed	Maximum number of buffers of a given type allowed in the system
Failures	Total number of allocation requests that have failed for lack of a free buffer
No memory	Number of failures due to a lack of memory to create a new buffer

Displaying System Memory Statistics

To show statistics about the memory, use the SHOW MEMORY command. Enter this command at the EXEC prompt:

show memory

This command displays memory-free pool statistics. These statistics include summary information about the activities of the system memory allocator and a block-by-block listing of memory use. Sample output follows. Table 6–2 describes the fields; Table 6–3 lists the characteristics of each block of memory in the system.

```
            Head   Free Start   Total Bytes     Used Bytes      Free Bytes
Processor  148B8C     1D66B0       2847860         561252         2286608

Address   Bytes Prev.   Next    Free? PrevF   NextF   Alloc PC  What
148B8C      916 0       148F20                        7B8E      *Init*
14B27C      100 14AA94  14B2E0                        3F2FE     Logger
14B2E0      152 14B27C  14B378                        1DBC      Router Init
14B378      480 14B2E0  14B558    y   1CE270  1CA938  5E85A     IGRP Router
14B558      100 14B378  14B5BC                        57A8E     DECnet Input
14B5F8       72 14B5BC  14B640                        7B830     XNS Router
14B6BC       88 14B688  14B714                        C724C     Virtual Exec
```

Table 6–2 Show Memory Field Descriptions

Field	Description
Head	Hexadecimal address of the head of the memory allocation chain
Free start	Hexadecimal address of the base of the free list
Total bytes	Total amount of system memory
Used bytes	Amount of memory in use
Free bytes	Amount of memory not in use

Table 6–3 Characteristics of Each Block of Memory

Field	Description
Address	Hexadecimal address of block
Bytes	Size of block in bytes
Prev	Address of previous block (should match address on previous line)
Next	Address of next block (should match address on next line)
Free?	Tells whether the block is free
Alloc PC	Address of the system call that allocated the block
What	Name of process that owns the block

Displaying Active System Processes

To see information about the active processes, use the SHOW PROCESSES command. Enter this command at the EXEC prompt:

show processes

Following is a partial display of the command output. Table 6–4 describes the fields.

CPU utilization for five seconds: 19%; one minute: 12%; five minutes: 12%

```
PID Q T     PC Runtime (ms)     Invoked   uSecs  Stacks     TTY Process
  1 M E  17518        40072         830   48279  606/800    0 Net Background
  2 M E   5040          932          11   84727  486/800    0 Logger
  32 M E  4C390        73012        6141   11889  480/800    0 IGRP Router
  4 M E  22984          172         252     682  662/800    0 BOOTP Server
  5 H E   5040       100324       66619    1505  606/900    0 IP Input
  6 M E  21278        12188       22451     542  508/800    0 IP Protocols
  7 M E  32F32           32       10926       2  592/800    0 TCP Timer
  8 L E  33C1E          508          28   18142  576/800    0 TCP Protocols
  9 L E   5040         1104         935    1180  666/800    0 ARP Input
 10 L E   5040          352         458     768  674/800    0 Probe Input
 11 H E   5040         2636        9077     290  710/800    0 Net Input
 12 M T   2CF2        36976       49175     751  602/800    0 TTY Background
 13 H E   5040            0           2       0  852/900    0 DECnet Input
 14 M E  44AE4        21964       18029    1218  742/900    0 DECnet Routing
```

Table 6–4 Show Processes Field Descriptions

Field	Description
CPU utilization	Ratio of current idle time over the longest idle time. Gives the user a general idea of how busy the processor is.
PID	Process ID
Q	Process queue priority (high, medium, low)
T	Scheduler test (Event, Time, Suspended)
PC	Current program counter
Runtime (ms)	CPU time the process has used, in milliseconds
Invoked	Number of times the process has been invoked
uSecs	Microseconds of CPU time for each invocation
Stacks	Low water mark/Total stack space available
TTY	Terminal that controls the process
Process	Name of process

Note

Because the router has a 4-millisecond clock resolution, run times are considered reliable only after a large number of invocations or after a reasonable, measured run time.

Displaying Memory Utilization

To show memory utilization, use the SHOW PROCESSES MEMORY command. Enter this command at the EXEC prompt:

show processes memory

The SHOW PROCESSES MEMORY command monitors the memory utilization of processes. The following is a display of sample output. Table 6–5 describes the fields.

```
Total: 2416588, Used: 530908, Free: 1885680
   PID  TTY  Allocated      Freed    Holding Process
     0    0     462708       2048     460660 *Init*
     0    0         76       4328 -     4252 *Sched*
     0    0      82732      33696      49036 *Dead*
     1    0       2616          0       2616 Net Background
     2    0          0          0          0 Logger
    21    0      20156         40      20116 IGRP Router
     4    0        104          0        104 BOOTP Server
     5    0          0          0          0 IP Input
     6    0          0          0          0 TCP Timer
     7    0        360          0        360 TCP Protocols
     8    0          0          0          0 ARP Input
     9    0          0          0          0 Probe Input
    10    0          0          0          0 MOP Protocols
    11    0          0          0          0 Timers
    12    0          0          0          0 Net Input
                                          530936 Total
```

Table 6–5 Show Processes Memory Field Descriptions

Field	Description
PID	Process ID
TTY	Terminal that controls the process
Allocated	Sum of all memory that process has requested from the system
Freed	Amount of memory a process has returned to the system
Holding	Allocated memory minus freed memory; can be negative when it has freed more than it was allocated
Process	Process name
Init	System initialization
Sched	The scheduler
Dead	Processes as a group that are now dead
Total	Total amount of memory held

Displaying Stack Utilization

To show stack utilization, use the SHOW STACKS command. Enter this command at the EXEC prompt:

```
show stacks
```

The SHOW STACKS command monitors the stack utilization of processes and interrupt routines. Its display includes the reason for the last system reboot. If the system was reloaded because of a system failure, a saved system stack trace is displayed. This information can be useful for analyzing crashes in the field.

Displaying the System Configuration

To display the contents of the nonvolatile memory, if present and valid, use the SHOW CONFIGURATION command. Enter this command at the EXEC prompt:

```
show configuration
```

The nonvolatile memory stores the configuration information in the router in text form as configuration commands.

Displaying the Error Logging Conditions

To show the state of logging (syslog), use the SHOW LOGGING command. Enter this command at the EXEC prompt:

```
show logging
```

This command displays the state of syslog error and event logging, including host addresses, and whether console logging is enabled. This command also displays simple network management protocol (SNMP) configuration parameters and protocol activity. See the section Redirecting System Error Messages in Chapter 4 for an explanation of how to configure message logging. Following is a sample output. Table 6–6 describes the fields.

```
Syslog logging: enabled
    Console logging: disabled
    Monitor logging: level debugging, 266 messages logged.
    Trap logging: level informational, 266 messages logged.
    Logging to 131.108.2.238

SNMP logging: disabled, retransmission after 30 seconds
    0 messages logged
```

Table 6–6 Show Logging Field Descriptions

Field	Description
Syslog logging	When enabled, system logging messages are sent to a UNIX host that acts as a syslog server; that is, it captures and saves the messages.
Console logging	If enabled, states the level; otherwise this field displays disabled.
Monitor logging	The minimum level of severity required for a log message to be sent to a monitor terminal (not the console).
Trap logging	The minimum level of severity required for a log message to be sent to syslog server.
SNMP logging	Shows whether SNMP logging is enabled, the number of messages logged, and the retransmission interval.

Displaying Protocol Information

To display the configured protocols, use the SHOW PROTOCOLS command. Enter this command at the EXEC prompt:

show protocols

The command shows the global and interface-specific status of any configured Level 3 protocol; for example, IP, DECnet, Novell and AppleTalk. The following is sample output:

```
Global values:
  Internet Protocol routing is enabled
  DECNET routing is enabled
  XNS routing is enabled
  Appletalk routing is enabled
  X.25 routing is enabled
Ethernet 0 is up, line protocol is up
  Internet address is 131.108.1.1, subnet mask is 255.255.255.0
  Decnet cost is 5
  XNS address is 2001.AA00.0400.06CC
  AppleTalk address is 4.129, zone Twilight
Serial 0 is up, line protocol is up
  Internet address is 192.31.7.49, subnet mask is 255.255.255.240
Serial 1 is down, line protocol is down
  Internet address is 192.31.7.177, subnet mask is 255.255.255.240
  AppleTalk address is 999.1, zone Magnolia Estates
```

Troubleshooting Network Operations

The DECbrouter 90 includes software to aid in tracking down problems with the router or with other hosts on the network. The privileged EXEC command DEBUG enables the display of several classes of network events on the console terminal. The privileged UNDEBUG command turns off the display of these classes. The EXEC command SHOW DEBUGGING displays the state of each debugging option.

show debugging

See the section Redirecting System Error Messages in Chapter 4 for an explanation of how to configure message logging.

Note

The system gives high priority to debugging output. For this reason, debugging commands should be turned on only for troubleshooting specific problems or during troubleshooting sessions with Digital staff. Excessive debugging output can render the system unusable.

To list and briefly describe all the DEBUG command options, enter the DEBUG ? command at the privileged-level EXEC prompt.

debug ?

This section provides an overview of how to use the debugging commands. See the interface and protocol-specific chapters of this manual for the DEBUG command descriptions.

To turn on all system diagnostics, enter this command at the EXEC prompt:

debug all

Its converse, the UNDEBUG ALL command, turns off all diagnostic output.

Real-time Debugging Support

Time-stamping enhances real-time debugging by providing the relative timing of logged events. This information is especially useful when you send debugging output to Digital for assistance.

To configure the system for time-stamping, use the global command:

[**no**] **service timestamps**

Note

Unsolicited error messages are not time-stamped; this features applies only to messages resulting from the DEBUG command.

When debugging is enabled, the resulting messages are preceded by a time-stamp when they are sent to monitor terminals. The time-stamp is the format HHHH:MM:SS and indicates the time since the router was last rebooted.

Example

This example shows the command to enter configuration mode and the configuration command that implements the time-stamping of debugging events.

```
gw# configure
Configuring from terminal, memory, or network [terminal]?
Enter configuration commands, one per line.
Edit with DELETE, CTRL/W, and CTRL/U; end with CTRL/Z
service timestamps
```

Example

In this example, debugging modem events is activated, events are logged, and time-stamping occurs, indicating the relative time the event is logged based on the time since the router was last rebooted. The TTY4 asserting DTR event occurred 3 minutes and 48 seconds after the router was last rebooted

```
gw# debug modem
RS-232 modem signal debugging is on
cs# cs 4004
Trying RUBBLE.DEC.COM (16.108.61.34, 4004)....Open

TTY4: asserting DTR
[Connection to cs closed by foreign host]
cs#
TTY4: timed out waiting for CTS
TTY4: dropping DTR, hanging up
cs# configure
Configuring from terminal, memory, or network [terminal]?
Enter configuration commands, one per line.
Edit with DELETE, CTRL/W, and CTRL/U; end with CTRL/Z
service timestamps
%SYS-5-CONFIG_I: Configured from console by console ()

cs# cs 4004
Trying RUBBLE.DEC.COM (16.108.61.34, 4004)...Open
0:03:48: TTY4 asserting DTR
[Connection to RUBBLE closed by foreign host]
cs#
```

Example

In the following example, the ARP events occurred 52 minutes and 51 seconds after reload.

```
gw# debug arp
ARP packet debugging is on
IP ARP: sent req src 131.108.61.37 0000.0c01.f0fe, dst 131.108.19.40
0000.0000.0000
IP ARP: rcvd rep src 131.108.19.40 0000.0c01.0e3b, dst 131.108.61.37
ARP: (merge) Creating entry for IP addr: 131.108.19.40, hw: 0000.0c01.0e3b
cs# configure
Enter configuration commands, one per line.
Edit with DELETE, CTRL/W, and CTRL/U; end with CTRL/Z
service timestamp
^Z
cs#
%SYS-5-CONFIG_I: Configured from console by console ()
0:52:51:  IP ARP: sent req src 131.108.61.37 0000.0c01.f0fe,
dst 131.108.19.33 0000.0000.0000
0:52:51:  IP ARP: rcvd rep src 131.108.19.33 0000.0c01.0e3b,
dst 131.108.61.37
0:52:51:  ARP: (merge) Creating entry for IP addr: 131.108.19.33,
hw: 0000.0c01.0e3b
```

Testing Connectivity with the PING Command

As an aid to diagnosing basic network connectivity, many network protocols support the *packet internet groper* (ping) program, which sends an echo request packet to an address, then awaits a reply. Results from this echo protocol can help in evaluating the path-to-host reliability, delays over the path, and whether the host can be reached or is functioning.

Note

Not all hosts support pings for all protocols.

To implement this program, use the privileged EXEC command PING. When the PING command is entered, the system issues a prompt for one of the following protocol keywords— **appletalk, clns, ip, novell, pup,** or **xns**.

The default protocol is IP. After determining the protocol type, the PING command prompts for an address or host name, repeat count (default is 5), datagram size (default is 100 bytes), time-out interval (default is 2 seconds), and extended commands (default is none). The precise dialog varies from protocol to protocol.

If a host name or address is typed on the same line as the EXEC PING command, the default action is taken as appropriate for the protocol type of that name or address.

The PING command uses the exclamation point (!) and period (.) in its display. Each exclamation point indicates receipt of a reply. A period (.) indicates that the router timed out while waiting for a reply. Other characters may appear in the ping output display, depending on the protocol type. The output concludes with the success rate and minimum, average, and maximum round-trip times.

To abort a ping session, type the escape sequence (by default, Ctrl/^ X, which is done by simultaneously pressing the Ctrl, shift, and 6 keys, letting go, then pressing the X key).

Sample displays and tips for using these protocols are included in the chapters describing the protocols supported by the DECbrouter 90 PING command.

Checking Routes with the TRACE Command

The TRACE command is a useful debugging command that allows the network administrator to discover the routes that packets actually will take when traveling to their destinations. The TRACE command supports **ip, clns,** and **vines** route tracing.

trace [*destination*]

To invoke a simple trace test, enter the destination address or host name on the command line. The default parameters for the appropriate protocol are assumed, and the tracing action begins.

To use nondefault parameters and invoke an extended trace test, enter the command without a destination argument. You will be stepped through a dialog to select the desired parameters.

Typing the escape sequence (by default, Ctrl/^ X, which is done by simultaneously pressing the Ctrl, shift, and 6 keys, letting go, then pressing the X key) terminates a TRACE command. See the *DECbrouter 90 Products, Configuration and Reference, Volume 2* and the *DECbrouter 90 Products, Configuration and Reference, Volume 3* respectively, for more information about using this command.

Writing System Configuration Information

This section describes the privileged WRITE commands used to manage the system configuration information.

To erase the configuration information, use the following EXEC command. This command erases the configuration information in the nonvolatile memory. This command does not affect the configuration in use.

write erase

To copy the configuration to memory, use the following EXEC command. This command copies the current configuration information to the nonvolatile memory.

write memory

To copy the configuration to the network, use the following EXEC command. This command sends a copy of the current configuration information to a server host. You are prompted for a destination host and a file name.

write network

To write the configuration on the terminal, use the following EXEC command. This command displays the current configuration information on the terminal.

write terminal

Testing the System

Included as part of the EXEC command set are commands that allow testing of system interfaces and memory.

Note

Digital does not recommend using these commands; they are intended to aid Digital manufacturing staff in checking out system functionality.

The following EXEC command is used to test the system interfaces:

test interfaces

The EXEC TEST INTERFACES command is intended for the factory checkout of network interfaces. It is not intended for diagnosing problems with an operational router. The TEST INTERFACES output does not report correct results if the router is attached to a "live" network. For each network interface that has an IP address that can be tested in loopback (all serial interfaces), the TEST INTERFACE command sends a series of ICMP echoes. Error counters are examined to determine the operational status of the interface.

EXEC System Management Command Summary

This section lists all the EXEC system management and user commands in alphabetical order.

debug ?

Lists and briefly describes all the DEBUG command options.

[un]debug all

Enables all diagnostic output. Its converse, the UNDEBUG ALL command, turns off all diagnostic output.

ping

Provides a diagnostic tool for testing connectivity. Results help evaluate path-to-host reliability, delays over the path, and whether the host is functioning.

[no] service timestamps

Enhances real-time debugging by providing the relative timing of logged events. This information is especially useful when you send debugging output to Digital technical support personnel for assistance.

show ?

Lists all the SHOW command options. Two lists can be displayed: one at the user-level prompt and one at the enabled, privileged-level prompt.

show buffers [*interface*]

Displays statistics for the buffer pools on the DECbrouter 90. The router has one pool of queuing elements and five pools of packet buffers of different sizes. For each pool, the router keeps counts of the number of buffers outstanding, the number of buffers in the free list, and the maximum number of buffers allowed in the free list. With the optional argument *interface*, this command searches all buffers that have been associated with the interface for longer than one minute.

show configuration

Displays the contents of the nonvolatile memory, if present and valid. The nonvolatile memory stores the configuration information in the DECbrouter 90 in text form as configuration commands.

show debugging

Displays the current settings of the debug command options.

show flash

Displays the total amount of flash memory present, any files that may currently exist in flash memory, and the amounts of flash memory used and remaining.

show flash all

Displays all the output of **show flash**; also shows information about each Flash memory device. Must be in the bootstrap mode to see the full output.

show logging

Displays the state of syslog error and event logging, including host addresses and whether console logging is enabled. This command also displays SNMP configuration parameters and protocol activity.

show memory

Displays memory-free pool statistics. These statistics include summary information about the activities of the system memory allocator and a block-by-block listing of memory use.

show processes

Displays information about all active processes.

show process memory

Displays information about memory utilization.

show protocols

Displays the global and interface-specific status of any configured level 3 protocol.

show stacks

Monitors the stack utilization of processes and interrupt routines. Its display includes the reason for the last system reboot. If the system was reloaded because of a system failure, a saved system stack trace is displayed. This information can be useful to support staff for analyzing crashes in the field.

test interfaces

Intended for the factory checkout of network interfaces. It is not intended for diagnosing problems with an operational router.

trace [*destination*]

Allows the network administrator to discover the routes that packets actually will take when traveling to their destinations. The TRACE command supports IP, CLNS, and VINES route tracing. Typing the escape sequence terminates TRACE.

write erase

Erases the configuration information in the nonvolatile memory. This command does not affect the configuration in use.

write memory

Copies the current configuration information to the nonvolatile memory.

write network

Sends a copy of the current configuration information to a server host. You are prompted for a destination host and a file name.

write terminal

Displays the current configuration information.

7

Configuring the Interfaces

This chapter contains information on enabling, shutting down, and displaying messages pertinent to interface configuration and operation. Also discussed are the procedures and command descriptions for configuring and maintaining the following router interface components:

- Serial interfaces
- Ethernet interfaces

You also will find information about configuring the serial Dial Backup feature and learn how to configure a null interface and the point-to-point protocol (PPP) and HDLC.

To enable an interface, you must be in the configuration command collection mode. To enter this mode, enter the EXEC command CONFIGURE at the EXEC prompt. Once in the command collection mode, start configuring the interface by entering the INTERFACE command. When the interface is configured, you can check its status by entering EXEC SHOW commands at the EXEC prompt.

This chapter provides software configuration information only. For hardware technical descriptions, and for information about installing these interfaces, refer to the appropriate hardware installation and maintenance publication.

Summaries of the interface configuration commands and EXEC monitoring commands described in this chapter are included at the end of the chapter.

Specifying an Interface

Enter the INTERFACE global configuration command in configuration mode to identify a specific network interface (for example, serial port or Ethernet port). By entering this command you begin the configuration *subcommand* collection mode for the specified interface.

The INTERFACE command has the following syntax:

interface *type unit*

The argument *type* identifies the interface type; the argument *unit* identifies the connector number. Unit numbers are assigned at the factory at the time of installation or when added to a system and can be displayed with the SHOW INTERFACES command.

Example:

This example begins interface configuration subcommand collection mode for serial connector 0 (interface serial 0):

```
interface serial 0
```

Use the EXEC command SHOW INTERFACES (described later in this chapter) to determine the interface type and unit numbers.

In the interface configuration subcommand collection mode, you enter the interface subcommands for your particular routing or bridging protocol. The interface configuration subcommand collection mode ends when you enter a command that is not an interface subcommand or when you type the Ctrl/Z sequence.

Note

All DECbrouter 90 models have one Ethernet interface (Ethernet 0). The DECbrouter 90T1 has one serial interface (serial 0); the DECbrouter 90T2 and the DECbrouter 90T2A have two serial interfaces (serial 0 and serial 1).

Adding a Descriptive Name to an Interface

To add a descriptive name to an interface, use the DESCRIPTION interface subcommand.

description *name-string*
no description

The argument *name-string* is descriptive text to help you remember what is attached to this interface. The DESCRIPTION command is meant solely as a comment to be put in the configuration to help you remember what certain interfaces are for. The description will appear in the output of the following commands: SHOW CONFIGURATION, WRITE TERMINAL and SHOW INTERFACES. The **no** version removes the *name-string*.

Example

This example describes a 3174 controller on serial 0:

```
interface serial 0
description 3174 Controller for test lab
```

Shutting Down and Restarting an Interface

To disable an interface, use the SHUTDOWN interface subcommand. The full syntax for this command follows.

shutdown
no shutdown

The SHUTDOWN command disables all functions on the specified interface and prevents all packet transmission. The command also marks the interface as unavailable; this information is communicated to other routers through all

dynamic routing protocols. The interface will not be mentioned in any routing updates. On serial interfaces, this command causes the DTR signal to be dropped.

To restart a disabled interface, use the NO SHUTDOWN interface subcommand.

To check whether an interface is disabled, use the EXEC command SHOW INTERFACES as described in the next section. An interface that has been shut down is shown as administratively down in this command's display:

Examples

This example turns off the interface Ethernet 0:

```
interface ethernet 0
shutdown
```

This example turns the interface back on:

```
interface ethernet 0
no shutdown
```

Clearing Interface Counters

To clear the interface counters shown with the SHOW INTERFACES command, enter the following command at the EXEC prompt:

clear counters [*type unit*]

The command clears all the current interface counters from the interface unless the optional arguments *type* and *unit* are specified to clear only a specific interface type (serial and Ethernet) from a specific unit or card number.

Note

This command will not clear counters retrieved using SNMP, but only those seen with the EXEC SHOW INTERFACES command.

Displaying Information About an Interface

The software contains commands that you can enter at the EXEC prompt to display information about the interface, including the software and hardware versions, the controller status, and some statistics about the interfaces. These commands begin with the word "show." (The full list of these commands can be displayed by entering the command SHOW ? at the EXEC prompt.) A description of interface-specific SHOW commands follows.

Displaying the System Configuration

The SHOW VERSION command displays the configuration of the system hardware, the software version, the names and sources of configuration files, and the boot images. Enter this command at the EXEC prompt:

show version

The following shows sample output from this command:

```
DECbrouter 90 Software (DEWBR), Version 9.14(1)
Copyright (c) 1986-1992 by cisco Systems, Inc.
Compiled Thu 17-Dec-92 13:43

System Bootstrap, Version 4.6

Chrysostom uptime is 2 days, 10 hours, 0 minutes
System restarted by power-on
System image file is "DEWBR", booted through flash

DECbrouter 90 router (68030) processor (revision 0x00) with 1024K/1024K bytes
of memory.
Processor board serial number 00000000
DDN X.25 software.
Bridging software.
SuperLAT software (copyright 1990 by Meridian Technology Corp).
1 Ethernet/IEEE 802.3 interface.
2 Serial network interface.
32K bytes of non-volatile configuration memory.
4096K bytes of flash memory sized on embedded flash.
Configuration register is 0x102
```

In the output, the first block of text lists information about the system software, which includes the software release version number. Always specify the complete version number when reporting a possible software problem. In the sample output, the version number is 9.14, initial release.

The second block of text lists the system bootstrap version, in this case 4.3.

The third block of text includes the system name and *uptime*, or the amount of time the system has been up and running. Also displayed is a log of how the system was last booted, both as a result of normal system startup and of system error. For example, information can be displayed to indicate a bus error that is generally the result of an attempt to access a nonexistent address.

```
System restarted by bus error at PC0XC4CA address 0X210C0C0
```

The name of the flash file booted is shown.

The remaining output shows the hardware configuration and any nonstandard software options. The configuration register contents are displayed in hexadecimal notation.

Note

The output of the SHOW VERSION EXEC command also can provide certain messages, such as bus error messages. If such error messages appear, report the complete text of this message to your Digital technical support specialist.

Displaying Controller Status

The SHOW CONTROLLERS command displays current internal status information for different interfaces. Enter this command at the EXEC prompt:

show controllers {**serial** | **ether**}

Use the following keywords to display the information about each:

- **serial**—For the serial interfaces
- **ether**—For the Ethernet interface (You may follow the interface command with a specific interface number.)

Sample output for the Ethernet interface follows.

```
LANCE unit 0, idb 0x4D764, ds 0x4EDD0, regaddr = 0x2130000, reset_mask 0x4
IB at 0x10BFEC: mode=0x0000, mcfilter 0001/0100/0008/4000
station address 0800.2ba0.b51c  default station address 0800.2ba0.b51c
buffer size 1524
RX ring with 16 entries at 0x10C028
Rxhead = 0x10C050 (5), Rxp = 0x4EDFC (5)
00 pak=0x12704C ds=0x127156 status=0x80 max_size=1524 pak_size=87
01 pak=0x12366C ds=0x123776 status=0x80 max_size=1524 pak_size=85
02 pak=0x106F28 ds=0x107032 status=0x80 max_size=1524 pak_size=132
03 pak=0x10C184 ds=0x10C28E status=0x80 max_size=1524 pak_size=132
04 pak=0x1261D4 ds=0x1262DE status=0x80 max_size=1524 pak_size=132
05 pak=0x10ECEC ds=0x10EDF6 status=0x80 max_size=1524 pak_size=132
06 pak=0x125A98 ds=0x125BA2 status=0x80 max_size=1524 pak_size=139
07 pak=0x1067EC ds=0x1068F6 status=0x80 max_size=1524 pak_size=83
08 pak=0x10D738 ds=0x10D842 status=0x80 max_size=1524 pak_size=144
09 pak=0x124C20 ds=0x124D2A status=0x80 max_size=1524 pak_size=132
10 pak=0x11EE14 ds=0x11EF1E status=0x80 max_size=1524 pak_size=132
11 pak=0x12535C ds=0x125466 status=0x80 max_size=1524 pak_size=132
12 pak=0x126910 ds=0x126A1A status=0x80 max_size=1524 pak_size=96
13 pak=0x123DA8 ds=0x123EB2 status=0x80 max_size=1524 pak_size=106
14 pak=0x10DE74 ds=0x10DF7E status=0x80 max_size=1524 pak_size=74
15 pak=0x1060B0 ds=0x1061BA status=0x80 max_size=1524 pak_size=122
TX ring with 4 entries at 0x10C0D8, tx_count = 0
tx_head = 0x10C0E0 (1), head_txp = 0x4EE40 (1)
tx_tail = 0x10C0E0 (1), tail_txp = 0x4EE40 (1)
00 pak=0x000000 ds=0x10BD96 status=0x03 status2=0x0000 pak_size=118
01 pak=0x000000 ds=0x10BD96 status=0x03 status2=0x0000 pak_size=118
02 pak=0x000000 ds=0x10BD96 status=0x03 status2=0x0000 pak_size=118
03 pak=0x000000 ds=0x10BD96 status=0x03 status2=0x0000 pak_size=118
0 missed datagrams, 0 overruns, 0 late collisions, 0 lost carrier events
0 transmitter underruns, 0 excessive collisions, 0 babbles
0 memory errors, 0 spurious initialization done interrupts
Lance csr0 = 0x73
MK5 unit 0, idb 0x4F6AC, ds 0x50D18, regaddr 0x2130040, reset_mask 0x2
IB at 0x117D9C: mode=0x0108, local_addr=0, remote_addr=0
N1=1524, N2=1, scaler=100, T1=1000, T3=2000, TP=1
buffer size 1524
No serial cable attached
RX ring with 16 entries at 0x117DF8 : RLEN=4, Rxhead 0
00 pak=0x117660 ds=0x117774 status=80 max_size=1524 pak_size=0
01 pak=0x116F24 ds=0x117038 status=80 max_size=1524 pak_size=0
02 pak=0x1167E8 ds=0x1168FC status=80 max_size=1524 pak_size=0
03 pak=0x1160AC ds=0x1161C0 status=80 max_size=1524 pak_size=0
04 pak=0x115970 ds=0x115A84 status=80 max_size=1524 pak_size=0
05 pak=0x115234 ds=0x115348 status=80 max_size=1524 pak_size=0
06 pak=0x114AF8 ds=0x114C0C status=80 max_size=1524 pak_size=0
07 pak=0x1143BC ds=0x1144D0 status=80 max_size=1524 pak_size=0
08 pak=0x113C80 ds=0x113D94 status=80 max_size=1524 pak_size=0
09 pak=0x113544 ds=0x113658 status=80 max_size=1524 pak_size=0
10 pak=0x112E08 ds=0x112F1C status=80 max_size=1524 pak_size=0
11 pak=0x1126CC ds=0x1127E0 status=80 max_size=1524 pak_size=0
12 pak=0x111F90 ds=0x1120A4 status=80 max_size=1524 pak_size=0
13 pak=0x111854 ds=0x111968 status=80 max_size=1524 pak_size=0
14 pak=0x111118 ds=0x11122C status=80 max_size=1524 pak_size=0
15 pak=0x1109DC ds=0x110AF0 status=80 max_size=1524 pak_size=0
TX ring with 4 entries at 0x117EA8 : TLEN=2, TWD=7
```

```
tx_count = 0, tx_head = 0, tx_tail = 0
00 pak=0x000000 ds=0x000000 status=0x38 max_size=1524 pak_size=0
01 pak=0x000000 ds=0x000000 status=0x38 max_size=1524 pak_size=0
02 pak=0x000000 ds=0x000000 status=0x38 max_size=1524 pak_size=0
03 pak=0x000000 ds=0x000000 status=0x38 max_size=1524 pak_size=0
XID/Test TX desc at 0xFFFFFF, status=0x30, max_buffer_size=0, packet_size=0
XID/Test RX desc at 0xFFFFFF, status=0x0, max_buffer_size=0, packet_size=0
Status Buffer at 0x117F58: rcv=0, tcv=0, local_state=0, remote_state=0
phase=0, tac=0, currd=0x00000, curxd=0x00000
bad_frames=0, frmrs=0, T1_timeouts=0, rej_rxs=0, runts=0
0 missed datagrams, 0 overruns, 0 bad frame addresses
0 bad datagram encapsulations, 0 user primitive errors
0 provider primitives lost, 0 unexpected provider primitives
0 spurious primitive interrupts, 0 memory errors, 0 transmitter underruns
mk5025 registers: csr0 = 0x4000, csr1 = 0x0002, csr2 = 0x0511
                  csr3 = 0x7D9C, csr4 = 0x0000, csr5 = 0x0003
```

Note

The interface type is only queried at startup. If the hardware changes *subsequent* to initial startup, the wrong type is reported. This has *no* adverse effect on the operation of the software. For instance, if a DCE cable is connected to a dual-mode V.35 interface after the unit has been booted, the display presented for **show interfaces** incorrectly reports attachment to a DTE device, although the software recognizes the DCE interface and behaves accordingly.

Displaying Interface Statistics

The software also provides the SHOW INTERFACES command, which displays statistics for the network interfaces on the router. Enter the following command at the EXEC prompt:

show interfaces [*type unit*] [**accounting**]

Specify the optional arguments *type* and *unit* to display statistics for a particular network interface. The argument *type* can be one of the following: **ethernet** or **serial**. Use the argument *unit* to specify the interface unit number.

The optional keyword **accounting** displays the number of packets of each protocol type that has been sent through the interface. You can show these numbers for all interfaces, or you can specify a specific type and unit.

Except for protocols that are encapsulated inside other protocols, such as IP over X.25, the accounting option also shows the total of all bytes sent and received, including the MAC header. For example, it totals the size of the Ethernet packet or the size of a packet that includes HDLC encapsulation.

Table 7–1 lists the protocols for which per-packet accounting information is kept.

Table 7–1 Per-Packet Counted Protocols

Protocol	Notes
IP	n/a
DECnet	n/a
XNS	n/a
CLNS	n/a
AppleTalk	n/a
Novell	n/a
Apollo	n/a
VINES	n/a
Trans. Bridge	n/a
SR Bridge	n/a
Chaos	n/a
Xerox PUP	n/a
Digital MOP	The routers use MOP packets to advertise their existence to Digital machines that use the MOP protocol. A router periodically broadcasts MOP packets to identify itself as a MOP host. This results in MOP packets being counted, even when DECnet is not being actively used.
LAN Manager	LAN Network Manager and IBM Network Manager
Spanning Tree	n/a
ARP	For IP, Chaos, Apollo, Frame relay, SMDS
HP Probe	n/a
Serial Tunnel	SDLC

The SHOW INTERFACE command is used frequently while configuring and monitoring modules.

For further explanations and examples about specific interfaces, refer to the following sections in this chapter: Monitoring the Serial Interface and Monitoring the Ethernet Interface.

Serial Interface Support

This section describes how to specify, maintain, monitor, and debug the serial interface. This section also describes methods of serial encapsulation.

Specifying a Serial Interface

To specify a serial interface, use this configuration command:

interface serial *unit*

Specify the serial interface connector number with the argument *unit*.

Follow this command with the routing or bridging interface subcommands for your particular protocol or application, as described later in this manual.

The serial interfaces can query the cables to determine their types. However, they do so only at system startup, so the cables must be attached when the system is started. Issue a SHOW CONTROLLERS SERIAL command to determine how

the serial card has identified them. The command also will show the capabilities of the serial card and report controller-related failures.

Example

This command begins configuration on interface serial 0:

```
interface serial 0
```

Serial Encapsulation Methods

The serial interfaces support the following kinds of serial encapsulations:

- High-level data link control (HDLC)
- HDLC distant host (HDH)
- Frame relay
- Point-to-point protocol (PPP)
- Synchronous data link control (SDLC)
- Switched multimegabit data services (SMDS)
- Cisco serial tunnel (STUN)
- X.25-based encapsulations

With the exception of the PPP and HDLC encapsulations, these serial encapsulation methods are described in Chapter 9. The HDLC serial encapsulation method is described in this section. PPP serial encapsulation is described later in the section Configuring the Point-to-Point Protocol.

Change the encapsulation method by using the interface configuration subcommand ENCAPSULATION, followed by a keyword that defines the encapsulation method.

encapsulation *encapsulation-type*

The *encapsulation-type* argument is a keyword that identifies one of the following serial encapsulation types that the software supports:

- **bfex25**—Blacker front end encryption X.25 operation
- **ddnx25-dce**—DDN X.25 DCE operation
- **ddnx25**—DDN X.25 DTE operation
- **frame-relay**—Frame relay
- **hdh**—HDH protocol
- **hdlc**—HDLC protocol
- **lapb-dce**—X.25 LAPB DCE operation
- **lapb**—X.25 LAPB DTE operation
- **multi-lapb-dce**—X.25 LAPB multiprotocol DCE operation
- **multi-lapb**—X.25 LAPB multiprotocol DTE operation
- **ppp**—Point-to-point protocol (PPP)
- **sdlc-primary**—IBM serial SNA

- **sdlc-secondary**—IBM serial SNA
- **smds**—SMDS service
- **stun**—STUN protocol functions
- **x25-dce**—X.25 DCE operation
- **x25**—X.25 DTE operation

HDLC Serial Encapsulation Method

The DECbrouter 90 provides HDLC serial encapsulation for serial lines. This encapsulation method provides the synchronous framing and error detection functions of HDLC without windowing or retransmission. Although HDLC is the default serial encapsulation method, it can be reinstalled using the **hdlc** keyword with the ENCAPSULATION command as follows:

encapsulation hdlc

Maintaining the Serial Interface

Use the command CLEAR INTERFACE to reset the hardware logic on an interface. Enter this command at the EXEC prompt:

clear interface *type unit*

The arguments *type* and *unit* specify a particular interface type and its unit number. In this case, the argument *type* is **serial**.

Note

Under normal circumstances, you do not need to clear the hardware logic on interfaces.

Monitoring the Serial Interface

In general, use the command SHOW INTERFACES to display information about the serial interface. Enter this command at the EXEC prompt:

show interfaces [*type unit*] [**accounting**]

The argument *type* is specified as **serial** and *unit* is the interface unit number. If you do not provide values for the parameters *type* and *unit*, the command will display statistics for all the network interfaces. The optional keyword **accounting** displays the number of packets of each protocol type that have been sent through the interface.

Sample outputs of this command for HDLC synchronous serial interfaces follows.

Table 7–2 describes the fields.

```
Chrysostom> show interfaces serial 0
Serial 0 is up, line protocol is up
  Hardware is MK5025
  Internet address is 150.136.190.203, subnet mask is 255.255.255.0
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation HDLC, loopback not set, keepalive set (10 sec)
  Last input 0:00:07, output 0:00:00, output hang never
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 0 bits/sec, 0 packets/sec
     16263 packets input, 1347238 bytes, 0 no buffer
     Received 13983 broadcasts, 0 runts, 0 giants
     2 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 2 abort
     22146 packets output, 2383680 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets, 0 restarts
     1 carrier transitions
```

When you use the accounting option, only the accounting statistics are displayed.

```
Chrysostom> show interfaces serial 0 accounting
Serial0
       Protocol    Pkts In   Chars In   Pkts Out  Chars Out
             IP       7344    4787842       1803    1535774
      Appletalk      33345    4797459      12781    1089695
        DEC MOP          0          0        127       9779
            ARP          7        420         39       2340
```

Table 7–2 Show Interfaces Serial Field Descriptions

Field	Description
Serial ... is {up | down}...is administratively down	Gives the interface unit number and whether the interface hardware is currently active (whether carrier detect is present) or if it has been taken down by an administrator
Line protocol is {up | down | administratively down}	Tells whether the software processes that handle the line protocol consider the line usable (are keepalives successful?) or if it has been taken down by an administrator.
Hardware is	Specifies the hardware type.
Internet address is	Specifies the Internet address and subnet mask
MTU	Maximum transmission unit of the interface.
BW	Bandwidth of the interface in kilobits per second.
DLY	Delay of the interface in microseconds.
Rely	Reliability of the interface as a fraction of 255 (255/255 is 100% reliability), calculated as an exponential average over five minutes.
Load	Load on the interface as a fraction of 255 (255 /255 is completely saturated), calculated as an exponential average over five minutes.
Encapsulation	Encapsulation method assigned to interface.
Loopback	Tells whether loopback is set.
Keepalive	Tells whether keepalives are set.
Last input	The number of hours, minutes, and seconds since the last packet was successfully received by an interface. Useful for knowing when a dead interface failed.
Output hang	The number of hours, minutes, and seconds (or never) since the interface was last reset because of a transmission that took too long. When the number of hours in any of the "last" fields exceeds 24 hours, the number of days and hours is printed. If that field overflows, asterisks are printed.
Output queue, input queue, drops	Number of packets in output and input queues. Each number is followed by a slash, the maximum size of the queue, and the number of packets dropped due to a full queue.
Five-minute input rate; five-minute output rate	The average number of bits and packets transmitted per second in the last five minutes.
packets input	The total number of error-free packets received by the system.
Received... broadcasts	The total number of broadcast or multicast packets received by the interface.

(continued on next page)

Table 7–2 (Cont.) Show Interfaces Serial Field Descriptions

Field	Description
runts	The number of packets that are discarded because they are smaller than the medium's minimum packet size.
giants	The number of packets that are discarded because they exceed the medium's maximum packet size.
input error	The total number of no buffer, runts, giants, CRCs, frame, overrun, ignored, and abort counts. Other input-related errors also can increment the count, so this sum may not balance with the other counts.
CRC	The cyclic redundancy checksum generated by the originating station or far-end device does not match the checksum calculated from the data received. On a serial link, CRCs usually indicate noise, gain hits, or other transmission problems on the data link.
frame	The number of packets received incorrectly having a CRC error and a noninteger number of octets. On a serial line, this is usually the result of noise or other transmission problems.
overrun	The number of times the serial receiver hardware was unable to hand received data to a hardware buffer because the input rate exceeded the receiver's ability to handle the data.
ignored	The number of received packets ignored by the interface because the interface hardware ran low on internal buffers. Broadcast storms and bursts of noise can cause the ignored count to be increased.
abort	An illegal sequence of one bits on a serial interface. This usually indicates a clocking problem between the serial interface and the data link equipment.
packets output	Total number of messages transmitted by the system.
bytes output	Total number of bytes, including data and MAC encapsulation, transmitted by the system.
Underruns	Number of times that the transmitter has been running faster than the router can handle. This may never be reported on some interfaces.

(continued on next page)

Table 7–2 (Cont.) Show Interfaces Serial Field Descriptions

Field	Description
Output errors	The sum of all errors that prevented the final transmission of datagrams out of the interface being examined. Note that this may not balance with the sum of the enumerated output errors, as some datagrams may have more than one error, and others may have errors that do not fall into any of the specifically tabulated categories.
interface resets	The number of times an interface has been completely reset. This can happen if packets queued for transmission were not sent within several seconds' time. On a serial line, this can be caused by a malfunctioning modem that is not supplying the transmit clock signal or by a cable problem. If the system notices that the carrier detect line of a serial interface is up but the line protocol is down, it periodically resets the interface in an effort to restart it. Interface resets also can occur when an interface is looped back or shut down.
restarts	The number of times the controller was restarted because of errors.
carrier transitions	The number of times the carrier detect signal of a serial interface has changed state. Indicates modem or line problems if the carrier detect line is changing state often.
Protocol	The protocol that is operating on the interface.
Pkts In	The number of packets received for that protocol.
Chars In	The number of characters received for that protocol.
Pkts Out	The number of packets transmitted for that protocol.
Chars Out	The number of characters transmitted for that protocol.

Debugging the Serial Interface

Use the commands DEBUG SERIAL-INTERFACE and DEBUG SERIAL-PACKET to debug serial interface events. The EXEC commands are as follows:

debug serial-interface
debug serial-packet

Use DEBUG SERIAL-PACKET for detailed debugging information and DEBUG SERIAL-INTERFACE for more general information.

Use UNDEBUG SERIAL-INTERFACE and UNDEBUG SERIAL-PACKET to turn off messaging from these DEBUG commands.

Ethernet Interface Support

The DECbrouter 90 has one built-in Ethernet interface compatible with Ethernet versions 1 and 2 and the IEEE 802.3 protocol.

Specifying an Ethernet Interface

To specify an Ethernet interface, use this configuration command:

interface ethernet *unit*

Because the DECbrouter 90 has only one Ethernet interface, use a unit number of 0.

Follow this command with the routing or bridging interface subcommands for your particular protocol or application as described later in this manual.

Example

This command begins configuration on interface Ethernet 0.

```
interface ethernet 0
```

Ethernet Encapsulation Methods

The Ethernet interface supports a number of encapsulation methods. These methods are assigned by using the interface subcommand ENCAPSULATION followed by a keyword that defines the encapsulation method. The encapsulation method used depends on the protocol. Currently, there are three common Ethernet encapsulation methods:

- The standard Ethernet Version 2.0 encapsulation that uses a 16-bit protocol type code
- The IEEE 802.3 encapsulation, in which the type code becomes the frame length for the IEEE 802.2 LLC encapsulation (destination and source Service Access Points and a control byte)
- The SNAP method, as specified in RFC 1042, which allows Ethernet protocols to run on IEEE 802.2 media

The syntax of the ENCAPSULATION command follows.

encapsulation *encapsulation-type*

The *encapsulation-type* is one of the following three keywords:

- **arpa**—Standard Ethernet Version 2.0 encapsulation
- **iso1**—IEEE 802.3 encapsulation
- **snap**—IEEE 802.2 Ethernet media

Example

These commands enable standard Ethernet Version 2.0 encapsulation on interface Ethernet 0:

```
interface ethernet 0
encapsulation arpa
```

Maintaining the Ethernet Interface

Use the command CLEAR INTERFACE to reset the hardware logic on an interface. Enter this command at the EXEC prompt:

clear interface *type unit*

The arguments *type* and *unit* specify a particular interface type and its unit number. In this case, the argument *type* is **ethernet**.

Note

Under normal circumstances, you do not need to clear the hardware logic on interfaces.

Monitoring the Ethernet Interface

Use the command SHOW INTERFACES to display information about the Ethernet interface. Enter this command at the EXEC prompt:

show interfaces [*type unit*] [**accounting**]

Where *type* is the ethernet keyword and *unit* is the interface unit number. If you do not provide values for the parameters *type* and *unit*, the command will display statistics for all the network interfaces. The optional keyword **accounting** displays the number of packets of each protocol type that have been sent through the interface.

Sample output of this command follows. Table 7–3 describes the fields.

```
Chrysostom> show interfaces ethernet 0

Ethernet 0 is up, line protocol is up
  Hardware is Lance, address is aa00.0400.0134 (bia 0000.0c00.4369)
  Internet address is 131.108.1.1, subnet mask is 255.255.255.0
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, PROBE, ARP Timeout 4:00:00
  Last input 0:00:00, output 0:00:00, output hang never
  Output queue 0/40, 0 drops; input queue 0/75, 2 drops
  Five minute input rate 61000 bits/sec, 4 packets/sec
  Five minute output rate 1000 bits/sec, 2 packets/sec
     2295197 packets input, 305539992 bytes, 0 no buffer
     Received 1925500 broadcasts, 0 runts, 0 giants
     3 input errors, 3 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     3594664 packets output, 436549843 bytes, 0 underruns
     8 output errors, 1790 collisions, 10 interface resets, 0 restarts
```

When you use the accounting option, only the accounting statistics are displayed.

```
Chrysostom> show interfaces ethernet 0 accounting
Ethernet0
       Protocol    Pkts In   Chars In   Pkts Out  Chars Out
             IP       7344    4787842       1803    1535774
      Appletalk      33345    4797459      12781    1089695
        DEC MOP          0          0        127       9779
            ARP          7        420         39       2340
```

Table 7–3 Show Interfaces Ethernet Field Descriptions

Field	Description
Ethernet ... is up ...is administratively down	Tells whether the interface hardware is currently active or if it has been taken down by an administrator.
line protocol is {up | down | administratively down	Tells whether the software process that handles the line protocol believes the interface is usable (are the keepalives successful?)
Hardware	Specifies the hardware type (for example, Lance) and address.
Internet address	Lists the Internet address followed by subnet mask
MTU	Maximum transmission unit of the interface.
BW	Bandwidth of the interface in kilobits per second.
DLY	Delay of the interface in microseconds.
rely	Reliability of the interface as a fraction of 255 (255/255 is 100% reliability), calculated as an exponential average over five minutes.
load	Load on the interface as a fraction of 255 (255 /255 is completely saturated), calculated as an exponential average over five minutes.
Encapsulation	Encapsulation method assigned to interface.
loopback	Tells whether loopback is set.
keepalive	Tells whether keepalives are set.
ARP type:	Type of address resolution protocol assigned.
Last input	The number of hours, minutes, and seconds since the last packet was successfully received by an interface. Useful for knowing when a dead interface failed.
output	Number of hours, minutes, and seconds since the last packet was successfully transmitted by the interface. Useful for knowing when a dead interface failed.
output hang	The number of hours, minutes, and seconds (or never) since the interface was last reset because of a transmission that took too long. When the number of hours in any of the "last" fields exceeds 24 hours, the number of days and hours is printed. If that field overflows, asterisks are printed.

(continued on next page)

Table 7–3 (Cont.) Show Interfaces Ethernet Field Descriptions

Field	Description
Last clearing	The time at which the counters that measure cumulative statistics (such as number of bytes transmitted and received) shown in this report were last reset to zero. Note that variables that might affect routing (for example, load and reliability) are not cleared when the counters are cleared.
Output queue, input queue, drops	Number of packets in output and input queues. Each number is followed by a slash, the maximum size of the queue, and the number of packets dropped due to a full queue.
Five-minute input rate; five-minute output rate	The average number of bits and packets transmitted per second in the last five minutes.
Packets input	The total number of error-free packets received by the system.
Received ... broadcasts	The total number of broadcast or multicast packets received by the interface.
Runts	The number of packets that are discarded because they are smaller than the medium's minimum packet size. For instance, any Ethernet packet that is less than 64 bytes is considered a runt.
Giants	The number of packets that are discarded because they exceed the medium's maximum packet size. For example, any Ethernet packet that is greater than 1518 bytes is considered a giant.
Input error	Includes runts, giants, no buffer, CRC, frame, overrun, and ignored counts. Other input-related errors also can cause the input errors count to be increased, and some datagrams may have more than one error; therefore, this sum may not balance with the sum of enumerated input error counts.
CRC	The cyclic redundancy checksum generated by the originating LAN station or far-end device does not match the checksum calculated from the data received. On a LAN, this usually indicates noise or transmission problems on the LAN interface or the LAN bus itself. A high number of CRCs is usually the result of collisions or a station transmitting bad data.
Frame	The number of packets received incorrectly having a CRC error and a noninteger number of octets. On a LAN, this is usually the result of collisions or a malfunctioning Ethernet device.

(continued on next page)

Table 7–3 (Cont.) Show Interfaces Ethernet Field Descriptions

Field	Description
Overrun	The number of times the receiver hardware was unable to hand received data to a hardware buffer because the input rate exceeded the receiver's ability to handle the data.
Ignored	The number of received packets ignored by the interface because the interface hardware ran low on internal buffers. These buffers are different than the system buffers mentioned previously in the buffer description. Broadcast storms and bursts of noise can cause the ignored count to be increased.
Packets output	Total number of messages transmitted by the system.
Bytes	Total number of bytes, including data and MAC encapsulation, transmitted by the system.
Underruns	Number of times that the transmitter has been running faster than the router can handle. This may never be reported on some interfaces.
Output errors	The sum of all errors that prevented the final transmission of datagrams out of the interface being examined. Note that this may not balance with the sum of the enumerated output errors, because some datagrams may have more than one error, and others may have errors that do not fall into any of the specifically tabulated categories.
Collisions	The number of messages retransmitted due to an Ethernet collision. This is usually the result of an overextended LAN (Ethernet or transceiver cable too long, more than two repeaters between stations, or too many cascaded multiport transceivers). A packet that collides is counted only once in output packets.
Interface resets	The number of times an interface has been completely reset. This can happen if packets queued for transmission were not sent within several seconds. On a serial line, this can be caused by a malfunctioning modem that is not supplying the transmit clock signal, or by a cable problem. If the system notices that the carrier detect line of a serial interface is up, but the line protocol is down, it periodically resets the interface in an effort to restart it. Interface resets also can occur when an interface is looped back or shut down. restarts The number of times a type 2 Ethernet controller was restarted because of errors.
Protocol	The protocol that is operating on the interface.

(continued on next page)

Table 7–3 (Cont.) Show Interfaces Ethernet Field Descriptions

Field	Description
Pkts In	The number of packets received for that protocol.
Chars In	The number of characters received for that protocol.
Pkts Out	The number of packets transmitted for that protocol.
Chars Out	The number of characters transmitted for that protocol.

Debugging the Ethernet Interface

Use the command DEBUG BROADCAST to debug MAC broadcast packets. Enter this command at the EXEC prompt:

```
debug broadcast
undebug broadcast
```

Use the UNDEBUG BROADCAST command to turn off messaging.

Use the command DEBUG PACKET to enable a log of packets that the network is unable to classify. Examples of this are packets with unknown link type or IP packets with an unrecognized protocol field. Enter this command at the EXEC prompt:

```
debug packet
undebug packet
```

Use the UNDEBUG PACKET command to turn off messaging.

Configuring Dial Backup Service

The dial backup service provides protection against WAN downtime by allowing you to configure a backup serial line through a circuit-switched connection.

To configure dial backup, associate a secondary serial interface as a backup to a primary serial interface. This feature requires that an external modem or CSU /DSU device be connected on the secondary serial interface. The external device must be capable of responding to a DTR signal (DTR active) by auto-dialing a connection to a preconfigured remote site.

The dial backup software keeps the secondary line inactive (DTR inactive) until one of the following conditions is met:

- The primary line goes down.
- The transmitted traffic load on the primary line exceeds a defined limit.

These conditions are defined using the interface subcommands described later in this section.

When the software detects a lost carrier detect signal from the primary line device or finds that the line protocol is down, it activates DTR on the secondary line. At that time, the modem or CSU/DSU must be set to dial the remote site. When that connection is made, the routing protocol defined for the serial line will continue the job of transmitting traffic over the dialup line.

You also can configure the dial backup feature to activate the secondary line based upon traffic load on the primary line.

The software monitors the traffic load and computes a five-minute moving average. If this average exceeds the value you set for the line, the secondary line is activated, and depending upon how the line is configured, some or all of the traffic will flow onto the secondary dialup line.

You also can specify a value that defines when the secondary line should be disabled and the amount of time the secondary line can take going up or down.

Configuring the Dial Backup Line

Use the BACKUP INTERFACE interface subcommands to configure the serial interface as a secondary, or dial backup, line. The commands have this syntax:

backup interface *interface-name*
no backup interface *interface-name*

The argument *interface-name* specifies the serial port to be set as the secondary interface line.

Use the NO BACKUP interface command with the appropriate serial port designation to turn this feature off.

Example

These commands set serial 1 as the backup line to serial 0:

```
interface serial 0
backup interface serial 1
```

Defining the Traffic Load Threshold

Use the BACKUP LOAD interface subcommands to set the traffic load thresholds. The commands have this syntax:

backup load {*enable-threshold* | **never**} {*disable-load* | **never**}
no backup load {*enable-threshold* | **never**} {*disable-load* | **never**}

Enter the arguments *enable-threshold* and *disable-load* using percentage numbers representing the load as a percentage of the primary line's available bandwidth.

When the transmitted or received load on the primary line is greater than the value assigned to the *enable-threshold* argument, the secondary line is enabled.

When the transmitted load on the primary line plus the transmitted load on the secondary line is less than the value entered for the *disable-load* argument, and the received load on the primary line plus the received load on the secondary line is less than the value entered for the *disable-load* argument, the secondary line is disabled.

If the **never** keyword is used instead of a *enable-threshold* value, the secondary line is never activated due to load. If the **never** keyword is used instead of an *disable-load* value, the secondary line is never deactivated due to load.

By default, no backup loads are defined.

Example

This example sets the traffic load threshold to 60 percent on the primary line. When that load is exceeded, the secondary line is activated, and will not be deactivated until the combined load is less than 5 percent of the primary bandwidth.

```
interface serial 0
backup load 60 5
```

Defining the Backup Line Delay

Use the BACKUP DELAY interface subcommands to define how much time should elapse before a secondary line is set up or taken down (after a primary line transitions). The commands have the following syntax:

backup delay {*enable-delay* | **never**} {*disable-delay* | **never**}
no backup delay {*enable-delay* | **never**} {*disable-delay* | **never**}

Note

The interval configured with the BACKUP DELAY subcommand does not affect the operation of the BACKUP LOAD subcommand.

The argument *enable-delay* is the delay in seconds after the primary line goes down before the secondary line is activated.

The argument *disable-delay* is the delay in seconds after the primary line goes up before the secondary line is deactivated.

When the primary line goes down, the router delays the amount of seconds defined by the *enable-delay* argument before enabling the secondary line. If, after the delay period, the primary line is still down, the secondary line is activated.

When the primary line comes back up, the router will delay the amount of seconds defined by the *disable-delay* argument. If the *disable-load* condition described previously can be satisfied, or if the secondary is never activated due to load, then the secondary line also is deactivated.

Note

In cases where there are spurious signal disruptions that may appear as intermittent lost carrier signals, it is recommended that some delay be enabled before activating and deactivating a secondary.

Use the **never** keyword to prevent the secondary line from being activated or deactivated.

The **never** version is the default value for **backup delay**. In order for the secondary line to be activated when the primary line fails, a **backup delay** value must be specified.

Example

This example sets a ten-second delay on deactivating the secondary line; however, the line is activated immediately.

```
interface serial 0
backup delay 0 10
```

Dial Backup Configuration Examples

This section contains three examples of different dial-backup configurations.

Example 1

The following example configures serial 1 as a secondary line that activates only when the primary line (serial 0) goes down. The secondary line will not be activated due to load of the primary.

```
interface serial 0
backup interface serial 1
backup delay 30 60
```

The secondary line is configured to activate 30 seconds after the primary line goes down and to remain on for 60 seconds after the primary line is reactivated.

Example 2

The following example configures the secondary line (serial 1) to be activated only when the load of the primary line reaches a certain threshold.

```
interface serial 0
backup interface serial 1
backup load 75 5
```

In this case, the secondary line will not be activated when the primary goes down. The secondary line will be activated when the load on the primary line is greater than 75 percent of the primary's bandwidth. The secondary line will then be brought down when the aggregate load between the primary and secondary lines fits within 5 percent of the primary bandwidth.

Example 3

This example configures the secondary line to activate once the traffic threshold on the primary line exceeds 25 percent.

```
interface serial 0
backup interface serial 1
backup load 25 5
backup delay 10 60
```

Once the aggregate load of the primary and the secondary lines return to within 5 percent of the primary bandwidth, the secondary line is deactivated. The secondary line waits 10 seconds after the primary goes down before activating, and remains active for 60 seconds after the primary returns and becomes active again.

Configuring Dial-on-Demand Routing

Dial-on-demand routing (DDR) provides network connections in an environment using the public switched telephone network (PSTN). Traditionally, networks have been interconnected using dedicated lines for WAN connections. When used with modems, DDR provides low-volume, periodic network connections over a PSTN. This section details the commands required to interconnect the DECbrouter 90 and the PSTN using data communications equipment (DCE) devices and dial-on-demand routing.

DECbrouter 90 DDR Implementation

DDR allows attachment to modems that support V.25bis dialing. V.25bis is a CCITT recommendation for initiating calls using an automatic calling unit (ACU). V.25bis is useful for establishing calls when transmitting synchronous data.

DDR supports large internetworks with many routers through its support of public circuit-switched networks such as telephone lines or switched services provided by the telephone companies.

The V.25bis specification describes two modes of establishing or receiving calls: the direct call mode and the addressed call mode. The DECbrouter 90 supports connections using the addressed call mode and synchronous, bit-oriented operation. The addressed call mode allows the use of control signals and commands sent over the DCE data interface to establish and terminate calls. These commands are packaged in synchronous data frames (HDLC type) and are sent when the interface is not in use for data transfer (that is, when a call does not exist on the line).

The DECbrouter 90 supports connections from synchronous serial interfaces on the router to any DCE device that supports V.25bis. Typically, these devices are V.32 (9.6 kbps) or V.32 bis modems (14.4 kbps).

The DDR call-initiating process is driven by certain characteristics of bridged or IP-routed packets. A router activates the dial-on-demand feature when it receives a bridged or routed IP packet destined for a location on the other side of the dial-up line. After the router dials the destination phone number and establishes the connection, packets of any protocol that the DECbrouter 90 supports can be transmitted. When the transmission is complete, the line is automatically disconnected.

In order for the router to use a device for dialing out, in addition to V.25 bis, the device must support certain hardware signals. When the router drops DTR, the device must disconnect any calls that are currently connected. When the device connects to the remote side, the device must have DCD become active.

Note

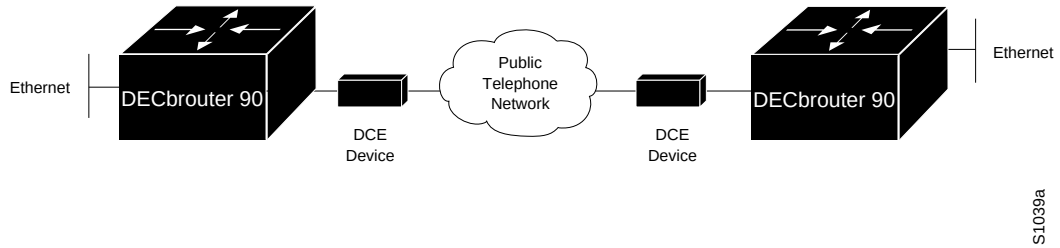
For many V.25bis devices, the requirement for DCD to be raised in turn requires a special cable to swap DCD and DSRs, because the V.25bis specification requires that DSR be raised when a connection is established.

IS-IS, BGP, and OSPF cannot be used as routing protocols with DDR. This is because they require an acknowledgment for routing updates, and DDR does not necessarily establish a connection when a routing update is to be transmitted.

For bridged traffic, packets of interest are based on *packet type* as specified with the hexadecimal value in a bridge access list. Refer to the *DECbrouter 90 Products, Configuration and Reference, Volume 3*, Appendix B, for more information about bridge access lists.

For IP traffic, packets of interest are based on the *destination address*; with a routed IP packet, the destination address used is the *next hop address*. Packets of interest are defined with IP access lists. Refer to the *DECbrouter 90 Products, Configuration and Reference, Volume 2* for more information concerning IP access lists. Figure 7–1 illustrates a typical interconnection featuring dial-on-demand routing.

Figure 7–1 Dial-on-Demand Interconnection



Specifying Dialer Type

Use the DIALER IN-BAND interface subcommand to specify that dial-on-demand routing is to be supported on a synchronous serial line. The syntax for this command is as follows:

dialer in-band
no dialer in-band

This command specifies that V.25bis dialing is to be used on the specified interface. It is used only with serial interfaces.

Note

If an interface is going to only accept calls and never place them, the DIALER IN-BAND interface subcommand is the only command needed to configure it. If an interface is configured in this manner, with no dialer groups (described in the next section), then it is always active, and the idle timer never disconnects the line. It is up to the remote end (the end that placed the call) to disconnect the line based on idle time.

The NO DIALER IN-BAND command disables dial-on-demand routing for the interface.

Example

The following example specifies dial-on-demand routing for serial 0:

```
interface serial 0
dialer in-band
```

When **dialer in-band** is specified, CRN is added to the front of the strings that are sent to the V.25bis DCE. CRN is a V.25bis command that specifies that the string that follows is a number to be called. This number is specified using the DIALER STRING or DIALER MAP commands described later in this chapter.

Controlling Dialer Idle Time

Use the optional DIALER IDLE-TIMEOUT interface subcommand to specify the idle time before the line is disconnected. The syntax for this command is as follows:

dialer idle-timeout *number-of-seconds*
no dialer idle-timeout

The argument *number-of-seconds* specifies in seconds how much idle time must occur on an interface before the line is disconnected. Acceptable values are positive, nonzero integers. If no value is entered, the default is 120 seconds.

The NO DIALER IDLE-TIMEOUT command resets the idle timeout to the default.

Example

The following example specifies of an idle timeout of 3 minutes (180 seconds) on interface serial 1:

```
interface serial 1
dialer idle-timeout 180
```

Setting Idle Time for Busy Interfaces

Use the optional dialer FAST-IDLE interface subcommand to specify the idle time before the line is disconnected for interfaces for which there is an unusually high level of contention. The syntax for this command is as follows:

dialer fast-idle *number-of-seconds*
no dialer fast-idle

The argument *number-of-seconds* specifies in seconds how much idle time must occur on an interface before the line is disconnected. Acceptable values are positive, nonzero integers. If no value is entered, the default is 20 seconds.

This timer is used when an interface is connected and a packet is received for a different destination (phone number). If the duration specified for the DIALER FAST-IDLE command is exceeded and no traffic has been detected on the connection during the fast idle period, the link is disconnected. This makes the port available for making connections to an alternate destination (after the **dialer enable-timeout** period expires) and reduces idle time on an interface.

This command only applies if multiple destinations have been specified for a given interface using the DIALER MAP interface subcommand.

The **no dialer fast-idle** resets the timeout period to the default.

Example

The following example specifies a fast idle timeout of 35 seconds on interface serial 1:

```
interface serial 1
dialer fast-idle 35
```

Specifying Dialer Enable Time

Use the optional DIALER ENABLE-TIMEOUT interface subcommand to set the length of time an interface stays down before it is available to dial again. The command syntax is as follows:

dialer enable-timeout *number-of-seconds*
no dialer enable-timeout

Acceptable values for *number-of-seconds* are positive, nonzero integers. If no value is entered, the default is 15 seconds. When a call terminates (hangs up or is busy) this is the amount of time that the router waits before the next call can occur on the specific interface.

The NO DIALER ENABLE-TIMEOUT command resets the enable timeout value to the default.

Example

The following example specifies a waiting period of 30 seconds on interface serial 1:

```
interface serial 1
dialer enable-timeout 30
```

Specifying Carrier Wait Time

Use the optional DIALER WAIT-FOR-CARRIER-TIME interface subcommand to specify how long to wait for a carrier. The command syntax is as follows:

dialer wait-for-carrier-time *number-of-seconds*
no dialer wait-for-carrier-time

The argument *number-of-seconds* specifies in seconds how long to wait for a carrier to come up when a call is placed. Acceptable values are positive, nonzero integers. If no value is entered, the default is 30 seconds. If a carrier signal is not detected in this amount of time, the interface is disabled until the enable timeout occurs.

The wait-for-carrier time is set to a default of 30 seconds, because this is the estimated time required to make a typical connection over telephone lines.

The NO DIALER WAIT-FOR-CARRIER-TIME command resets the carrier wait time value to the default.

Example

The following example specifies a carrier wait time of 45 seconds on interface serial 1:

```
interface serial 1
dialer wait-for-carrier-timeout 45
```

Specifying a Single DDR Telephone Number

Use the DIALER STRING interface subcommand to specify the string (telephone number) to be called. The command syntax is:

dialer string *dial-string*
no dialer string

The argument *dial-string* specifies the character string to be called. Dialers configured as **in-band**, pass the string to the external DCE. One DIALER STRING command is specified per interface.

To specify multiple strings, use the DIALER MAP command, discussed next. However, you must use the DIALER STRING command if you intend to bridge traffic over the serial link or transmit broadcast traffic. In general, you include a DIALER STRING or DIALER MAP command if you intend to use a specific interface to initiate a dial-on-demand call.

Note

If a DIALER STRING command is specified for the interface, and no DIALER-GROUP subcommand is assigned or no access lists are defined, dialing never will be initiated. An error message will be displayed indicating that dialing never will occur if DEBUG SERIAL-INTERFACE is enabled.

The *dial-string* number specified in the DIALER STRING command is the default number used under the following conditions:

- A DIALER MAP command is not included in the interface configuration
- The *next-hop-address* specified in a packet is not included in any of the DIALER MAP interface subcommands recorded—assuming that the destination address passes any access lists specified for dial-on-demand through the DIALER-LIST command (discussed later in this section)
- When bridging.

CCITT V.25bis Options

Depending on the type of modem you are using, CCITT V.25bis options are supported as dial-string parameters of the DIALER STRING command. The functions of the parameters are nation-specific, and they may have different implementations in your country. These options apply only if you have enabled DDR with the DIALER IN-BAND command. Refer to your manual for the modem you are using for a list of supported options.

Table 7–4 CCITT V.25bis Options

Option	Description
:	wait tone
<	Pause Usage and duration of this parameter vary by country.
=	Separator 3 For national use.
>	Separator 4 For national use.
P	Dialing to be continued in pulse mode. Optionally accepted parameter.
T	Tone (Dialing to be continued in DTMF mode) Optionally accepted parameter.
&	Flash The flash duration varies by country. Optionally accepted parameter.

The NO DIALER STRING command deletes the dialer string specified for the interface.

Examples

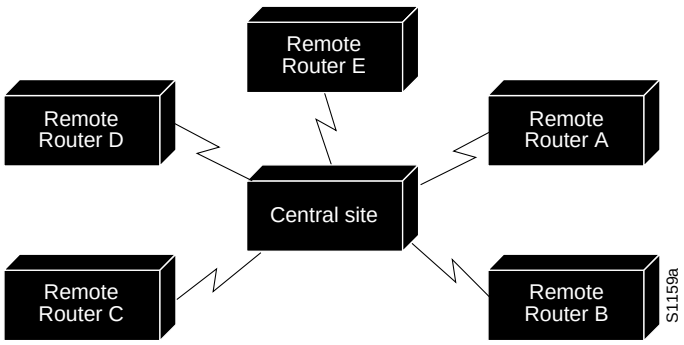
The following example specifies a dial-on-demand telephone number on interface serial 1 using the DIALER STRING command:

```
interface serial 1
dialer string 14085553434
```

Multiple Destinations

The DIALER STRING command is useful for environments in which a single site is calling another site but the other site is not calling out. Often, however, configurations are more complex, such as the hub-and-spoke environment shown in Figure 7–2. In this configuration multiple sites are calling in to a central site, and the central site is calling out to the remote sites.

Figure 7–2 Hub-and-Spoke Environment Using Dial-on-Demand



In this configuration, PPP with Challenge Handshake Authentication Protocol (CHAP) authentication often is used to inform the central site about which remote routers are connected to it. Using CHAP authentication, each remote router identifies itself by a name, which informs the central router what routers are currently connected to it. This identification process prevents the central router from placing a call to a remote router if it is already connected to that router.

Specifying Multiple Destinations

There are three forms of the interface subcommand you can use to define multiple dial-on-demand numbers for a particular interface. They are as follows:

dialer map *protocol next-hop-address* **username** *name*

dialer map *protocol next-hop-address dial-string*

dialer map *protocol next-hop-address* **username** *name dial-string*

The first form of the dialer map command is used in central-site configurations in which remote sites are calling a central site, but the central site is not calling the remote site. With this command, the remote device will authenticate the central site using CHAP, which will cause its name to be transmitted to the central site. The central site will then use this name and the next-hop-address to correctly transmit packets to the remote site. Because there is no dialer string specified, the central site cannot call the remote router.

The second form of the command is used if only one site is being called.

The third form is used if many remote sites are calling the central site and the central site also can call the remote sites. With this format, the central router can use the name to identify the remote routers after they authenticate and also can use the dialer string to call the remote routers if they are not currently connected.

The keyword **username** indicates a remote system that the local router communicates with.

The following arguments can be used with this command:

- The argument *protocol* has only one option at this time: **ip**.
- The argument *next-hop-address* specifies the IP address used to match against addresses to which packets are destined.
- The argument *name* is the name of the remote device (usually the host name).
- The *dial-string* argument is the telephone number sent to the DCE dialing device when it sees packets with the specified *next-hop-address* that match the access lists defined.

Unlike the DIALER STRING command, multiple DIALER MAP commands can be specified per interface.

The DIALER MAP command complements the DIALER STRING command. The *dial-string* defined for the DIALER STRING command is considered the *default* dialer string. If a packet does not match the *protocol* and the *next-hop-address* defined in the DIALER MAP commands (but does pass an access list that requires initiation of automatic dialing), the default dialer string is used. This is often the case for broadcasts and is always the case for bridging.

The NO DIALER MAP command deletes a particular dialer map entry. To delete the entry, include the complete configuration string (**no dialer map** *(protocol next-hop-address dial-string)*.

Note

If no DIALER STRING command is defined, and the next-hop address included in a packet does not match any of the *next-hop-address* specifications in the DIALER MAP command, dialing never will be initiated. Then an error message will be displayed indicating that dialing never will occur if DEBUG SERIAL-INTERFACE is enabled.

Examples

In the following examples, a remote site is calling a central site, but the central site is not calling the remote site. The remote device will authenticate the central site using CHAP, which will cause its name, YYY, to be transmitted to the central site. The central site will then use this name and the next-hop-address, 131.108.2.5, to correctly transmit packets to the remote site. Because there is no dialer string specified, the central site cannot call the remote router.

```
interface serial 1
dialer map ip 131.108.2.5 username YYY
```

The following example specifies a dial-on-demand telephone number on interface serial 1 for a specific destination address using the DIALER MAP command:

```
interface serial 1
dialer map ip 131.108.2.5 14155553434
```

The remote site is calling the central site, and the central site is calling the remote site. The central router can use the name, ZZZ, to identify the remote router after they authenticate and also can use the dialer string 1415555343 to call the remote router if it is not currently connected.

```
interface serial 1
dialer map ip 131.108.2.5 username ZZZ 1415555343
```

Dialer Rotary Groups

Dialer rotary groups allow you to apply a single interface configuration to a set of interfaces. Once the interface configuration is propagated to a set of interfaces, those interfaces can then be used to place calls using the standard dial-on-demand criteria. When many destinations are configured, any of these interfaces can be used for outgoing calls.

Dialer rotary groups are useful in environments that require many calling destinations. Only the rotary group needs to be configured with all of the DIALER MAP commands. The only configuration required for the interfaces is the DIALER ROTARY-GROUP command indicating that each interface is part of a dialer rotary group.

Although a dialer rotary group is configured as an interface, it is not a physical interface. Instead it represents a group of interfaces. Any number of dialer groups can be defined.

Use the following interface command to designate a dialer rotary group leader:

interface dialer *n*

The argument *n* is any number and indicates a dialer group.

Interface subcommands entered after the INTERFACE DIALER command will be applied to all physical interfaces assigned to rotary group *n*.

Example

This example identifies interface dialer 1 as the dialer rotary group leader. Interface dialer 1 is not a physical interface, but represents a group of interfaces. The interface subcommands that follow apply to all interfaces included in this group. See the DIALER ROTARY-GROUP command, defined next.

```
interface dialer 1
encapsulation ppp
dialer in-band
dialer map ip 131.108.2.5 username YYY 1415553434
dialer map ip 131.126.4.5 username ZZZ
```

Use the following interface subcommand to include an interface in a dialer rotary group:

dialer rotary-group *n*

The argument *n* is the number of the rotary group (defined by the interface dialer command).

Example

The following configuration places serial interfaces 0 and 1 into dialer rotary group 1, defined by the INFERFACE DAILER 1 command.

```
! PPP encapsulation is enabled for interface dialer 1.
interface dialer 1
encapsulation ppp
dialer in-band

! The first dialer map command allows remote site YYY and the
! central site to call each other. The second dialer map command, with no
! dialer string, allows remote site ZZZ to call the central site but
! the central site can not call remote site ZZZ (no phone number).
dialer map ip 131.108.2.5 username YYY 1415553434
dialer map ip 131.126.4.5 username ZZZ

! The DTR pulse signals for three seconds on the interfaces in dialer
! group 1. This holds the DTR low so the modem can recognize that DTR
  has been
! dropped.
pulse-time 3

! Interfaces serial 0 and serial 1 are placed in dialer rotary group 1.
  All of
! the interface subcommands (the encapsulation and dialer map commands shown
! earlier in this example)applied to interface dialer 1 apply
! to these interfaces.
interface serial 0
dialer rotary-group 1
interface serial 1
dialer rotary-group 1
```

Assigning Access Lists to a DDR Interface

Use the DIALER-GROUP interface subcommand to assign an interface to a set of access list expressions. These access list expressions define which packets cause a connection to be established and which packets keep an interface from being idle. The syntax for this command is as follows:

dialer-group *group-number*
no dialer-group

The argument *group-number* specifies the number of the dialer group to which the specific interface belongs. Acceptable values are nonzero, positive integers. This group number corresponds to a dialer group defined using the DIALER-LIST command (described in the next section).

An interface only can be associated with a single dialer group; multiple DIALER-GROUP assignment is not allowed.

The NO DIALER-GROUP command removes an interface from the specified dialer group.

Example

The following example specifies dialer group number 1. This dialer group number will be compared against any global DIALER-LIST specifications. If there is a group number match, the destination address of the packet in question is evaluated against the access list specified in the associated DIALER-LIST command. If it passes, a call is initiated or the idle timer is reset (if a call is currently connected).

```
interface serial 1
dialer-group 1
```

Using Access Lists with DDR

Control automatic dialing using DDR with the following combination: standard IP or bridging access lists; the DIALER-LIST global command; and the DIALER-GROUP interface subcommand (described in the preceding section). Refer to the *DECbrouter 90 Products, Configuration and Reference, Volume 2* and the *DECbrouter 90 Products, Configuration and Reference, Volume 3* for more information about access lists. The DIALER-LIST global configuration command has two forms. They have the following syntaxes:

dialer-list *dialer-group-number* **list** *list-number*
no dialer-list *dialer-group-number* **list** *list-number*

dialer-list *dialer-group* **protocol** *protocol-name* {**permit** | **deny**}
no dialer-list *dialer-group* **protocol** *protocol-name* {**permit** | **deny**}

Note

The **permit/deny** version of the DIALER-LIST command provides a very coarse method of allowing or denying protocols that prevent a link from being considered idle or from causing dialing. In most cases, this form of the DIALER-LIST command is not recommended because most network protocols have *periodic* messages. These periodic messages cause the interface to remain active; thus, the line is *never* disconnected. Currently, filtering using access lists is only provided for routing IP and bridging. In addition, *only* those access lists applicable to bridging and IP routing are supported by the dial-on-demand facilities.

The argument *dialer-group-number* specifies the number of a dialer group identified in any DIALER GROUP interface subcommand.

The argument *list-number* is the access list number specified in any IP or bridging access list.

The argument *protocol-name* is one of the following supported protocols:

- **ip**—IP
- **decnet**—DECnet
- **chaos**—CHAOSnet
- **xns**—XNS
- **novell**—Novell IPX
- **appletalk**—AppleTalk
- **vines**—Banyan Vines
- **apollo**—Apollo Domain
- **pup**—PUP
- **clns**—OSI Connectionless Network Service
- **bridge**—Bridging
- **stun**—Serial tunneling
- **cmns**—OSI Connection-Mode Network Services

Note

Access lists are central to the operation of dial-on-demand routing for the DECbrouter 90. In general, they are used as the screening criteria for determining when to initiate DDR calls. When an access list is found to match with the packet being tested, the idle timer is reset or a connection is attempted (assuming the line is not active, but is available). If a tested packet is deemed *uninteresting* (it does not match any access list specifications), it will be forwarded if it is intended for a destination known to be on a specific interface and the link is active. However, such a packet will not initiate a DDR call and will not reset the idle timer.

Example

Dialing occurs when an interesting packet (one that matches access list specifications) needs to be output on an interface. Using the standard access list method, packets can be classified as interesting or uninteresting. For example, to specify that IGRP updates are not interesting (relative to dial-on-demand automatic dialing), the following access list would be defined:

```
access-list 101 deny igrp 0.0.0.0 255.255.255.255 255.255.255.255 0.0.0.0
```

To permit all other IP traffic, the above would be followed by:

```
access-list 101 permit ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
```

Then, the following command would be used to place list 101 into dialer group 1:

```
dialer-list 1 list 101
```

Dial-on-Demand Access-List Configuration Examples

The examples provided in this section illustrate various DDR configurations using access lists linked to DDR facilities provided with the router.

Example 1

The following example illustrates how dial-on-demand routing can be used in an IP environment:

```
interface serial 0
ip address 131.108.126.1 255.255.255.0
dialer in-band
dialer idle-timeout 600
dialer string 555-1234
pulse-time 1
dialer-group 1
!
access-list 101 deny   igrp 0.0.0.0 255.255.255.255 255.255.255.255 0.0.0.0
access-list 101 permit ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
dialer-list 1 list 101
ip route 131.108.29.0 131.108.126.2
ip route 131.108.1.0 131.108.126.2
```

Configuration specifications in this example define the following characteristics:

- DIALER IN-BAND command selects V.25bis dialing.
- DIALER IDLE-TIMEOUT 600 command sets the dialer idle timeout to 10 minutes (600 seconds).
- DIALER STRING 555-1234 command sets the phone number to be called to 555-1234.
- PULSE-TIME 1 command sets the DTR pulse time to be one second. This is to give the modem enough time to recognize that DTR has dropped so the modem will disconnect the call.

Note

With many modems, the PULSE-TIME command must be used so that DTR is dropped for sufficient time to allow the modem to disconnect.

- **dialer-group 1** adds this interface to the dialer group defined with the dialer-list command.
- The first access list statement used specifies that IGRP updates to any address will not cause dialing. However, if a connection is already established, IGRP updates will be transmitted but will not reset the idle timers.
- The second access-list statement specifies that all other IP traffic—such as Ping, Telnet, or any other IP packet—will cause dialing.
- The DIALER-LIST command then creates dialer group 1 and states that access list 101 is to be used to match for automatic dialing.
- The IP ROUTE commands specify that there is a route to network 131.108.29.0 and to network 131.108.1.0 through 131.108.126.2. This means that several destination networks are available through the serial port.

Note

The REDISTRIBUTE STATIC command can be used to advertise static route information for dial-on-demand applications. See the REDISTRIBUTE STATIC command, described in *DECbrouter 90 Products, Configuration and Reference, Volume 2*, Chapter 6, "IP Routing Protocols." This command is useful for environments in which a workstation is connecting to a site that is reachable by DDR across multiple routers. Without this command, static routes must be defined for each route between your workstation and the DDR router.

Example 2

The following example illustrates a configuration for bridging:

```
interface serial 0
ip address 131.108.126.10 255.255.255.0
dialer in-band
dialer string 555-1234
pulse-time 1
bridge-group 1
dialer-group 1
!
!
bridge 1 protocol dec
access-list 201 deny   0x6002 0x0000
access-list 201 deny   0x6003 0x0000
access-list 201 permit 0x0000 0xFFFF
dialer-list 1 LIST 201
```

As in Example 1, the dialer is set to use V.25bis, the dialer string is specified, and a pulse time is assigned. This interface is then added to bridge group 1 and dialer group 1.

The access lists work as follows:

- The first one states that Digital MOP packets are not to cause dialing.
- The second states that DECnet Phase IV should not cause dialing.
- The last states that all other bridged packets should cause dialing.

This example could be used in a case where LAT is the only traffic to be transmitted over the link.

Note

Spanning-tree updates and HDLC keepalives implicitly do not cause dialing and do not reset the idle timers.

Example 3

The following example is a combination of Examples 1 and 2. It illustrates how to handle DDR for bridging and IP routing over a single interface.

The first **access-list** statement specifies that RIP updates to the broadcast address do not cause dialing. However, if a connection is already established, RIP updates are transmitted, although they do not reset the idle timers and do not keep the line up.

```
interface serial 0
ip address 131.108.126.10 255.255.255.0
dialer in-band
dialer string 555-1234
pulse-time 1
bridge-group 1
dialer-group 1
!
!
bridge 1 protocol dec
access-list 101 deny  udp 0.0.0.0 255.255.255.255 255.255.255.255 0.0.0.0
eq 520
access-list 101 permit ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
access-list 201 deny   0x6002 0x0000
access-list 201 deny   0x6003 0x0000
access-list 201 permit 0x0000 0xFFFF
dialer-list 1 LIST 101
dialer-list 1 LIST 201
```

Example 4

The following example demonstrates how to specify multiple destination dial strings (phone numbers):

```
interface serial 0
ip address 131.108.126.1 255.255.255.0
dialer in-band
dialer wait-for-carrier-time 40
dialer string 555-1234
pulse-time 1
dialer-group 1
dialer map ip 131.108.126.10 555-8899
dialer map ip 131.108.127.1 555-5555
!
access-list 101 deny   igrp 0.0.0.0 255.255.255.255 255.255.255.255 0.0.0.0
access-list 101 permit ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
dialer-list 1 LIST 101
```

As in Example 1, the serial interface is configured for V.25bis. It also is set to have a wait-for-carrier-time of 40 seconds. As before, a default dial string is configured, a pulse time assigned, and a dialer group specified.

The first DIALER MAP command specifies that the number 555-8899 is to be dialed for IP packets with a *next-hop-address* of 131.108.126.10. The second **dialer map** then specifies that the number 555-5555 will be called when an IP packet with a *next-hop-address* of 131.108.127.1 is detected.

Access lists for this example match those defined in Example 1.

Example 5

The following example illustrates using *floating static routes* (see *DECbrouter 90 Products, Configuration and Reference, Volume 2*, Chapter 6, "The IP Routing Protocols") and dial-on-demand to perform the same functionality as dial backup, except using V.25bis for dialing.

```
interface serial 0
ip address 131.108.126.1 255.255.255.0
dialer in-band
dialer wait-for-carrier-time 10
dialer string 555-1234
pulse-time 1
dialer-group 1
dialer map ip 131.108.126.10 555-8899
dialer map ip 131.108.127.1 555-5555
!
access-list 101 deny   igrp 0.0.0.0 255.255.255.255 255.255.255.255 0.0.0.0
access-list 101 permit ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
dialer-list 1 LIST 101
ip route 131.108.42.0 131.108.126.10 150
```

The configuration is essentially the same as in Example 4, except for the addition of an IP ROUTE statement. This statement specifies that to reach network 131.108.42.0 through this router, traffic must use 131.108.126.10 as the gateway. This causes dialing to occur to 555-8899.

Because the IP ROUTE statement specifies an administrative distance (150), dynamic routing information will override the static route—no packets will normally be sent to 131.108.126.10. However, if the route that is learned dynamically disappears, and no other dynamic route is available, the static route will be used and dialing will occur. See the *DECbrouter 90 Products, Configuration and Reference, Volume 2*, Chapter 6, "The IP Routing Protocols" for more information on static routes.

Monitoring Dial-on-Demand Routing

To obtain a general diagnostic display for serial interfaces configured for dial-on-demand routing, use the SHOW DIALER EXEC command. The command syntax is as follows:

show dialer [**interface** *interface type*]

This command displays information about the dialer interfaces. If SHOW DIALER is specified, all dialers are displayed. An interface can be specified using the SHOW DIALER INTERFACE option. The arguments *interface* and *type* specify the particular interface, and can be formed either as a two-part identifier (such as serial 0, ser 1, s 2, and so on) or as a name (serial1).

The following is a sample of output from the SHOW DIALER command with a serial interface:

```
Serial 0 - dialer type = IN-BAND
Idle timer (600 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (15 secs)
Dial String      Successes   Failures    Last called   Last status
8985                   0         2        0:24:10      Failed       Default
8986                   1         0        1:20:07      Successful
```

The following information is provided in this display:

- "IN-BAND" specification indicating that DDR is enabled
- Idle timeout specification (seconds)
- Fast-idle timer specification (seconds)
- Wait-for-carrier timer specification (seconds)
- Enable timeout specification (seconds)
- Dial strings of logged calls (telephone numbers)
- Successful connections (even if no data is passed)
- Failed connections—line busy when call attempted
- Time that last call occurred to specific dial string
- Last status of calls to specific dial string (successful or failed)
- If the DDR facility is using the dial string specified with the DIALER STRING command, the word `Default` is appended to the `Last status` entry

If the dialer is connected to another dialer (and the system from which the SHOW DIALER command was entered was the initiator of the call), a display is provided that indicates the idle time before the line is disconnected (decrements each second) and the duration of the current connection. The following is a sample of this display; it would appear after the third line in the SHOW DIALER display:

```
Time until disconnect 596 secs
Current call connected 0:00:25
```

After a call disconnects, the system displays the time remaining before being available to dial again. The following is a sample of this display; it would appear after the third line in the SHOW DIALER display.

```
Time until interface enabled 8 secs
```

If the SHOW DIALER interface command is issued for an interface on which DDR is not enabled, the system displays an error message. The following is a sample error message:

```
Serial 1 - Dialing not enabled on this interface.
```

If an interface is configured for DDR, the SHOW INTERFACES command now displays the following:

```
Serial 0 is up, line protocol is up (spoofing)
Hardware is MK5025
```

The *spoofing* indicates that the line really is not up, but the dialer is forcing the line to masquerade as "up" so that upper-level protocols will continue to operate as expected.

Debugging Dial-on-Demand Routing

When DDR is configured for a serial interface, the EXEC commands DEBUG SERIAL-INTERFACE (described earlier in this chapter) and DEBUG SERIAL-PACKET can provide information associated with DDR activity, as follows:

debug serial-interface

This command enables monitoring of serial interface activity. For DDR, the following information may appear:

- A message indicating that a call is being attempted
- A message stating that no dialer group has been defined and dialing will not occur
- Messages indicating that idle timeouts, reenable timeouts, and wait-for-carrier timeouts have occurred
- A message from the DCE device indicating the dial string attempted

The DEBUG SERIAL-PACKET command enables monitoring of serial packets.

debug serial-packet

When activated, and DDR is enabled on the interface, information concerning the cause of any calls (called *Dialing cause*) may be displayed. The cause for IP packets lists the source and destination addresses; the cause for bridged packets lists the type of packet in hexadecimal.

Configuring the Point-to-Point Protocol

The point-to-point protocol (PPP), described in RFC 1134, is designed to encapsulate internet protocol (IP) datagrams and other network layer protocol information over point-to-point links. The document "Point-to-Point Initial Configuration Options" defines the set of options that are negotiated during startup.

The current implementation of PPP supports no configuration options. The software sends no options, and any proposed options are rejected.

Of the possible upper-layer protocols, only IP is supported at this time. Thus, the only upper-level protocol that can be sent or received over a point-to-point link using PPP encapsulation is IP. Refer to RFC 1134 for definitions of the codes and protocol states.

The PPP is enabled on an interface using the ENCAPSULATION interface subcommand followed by the **ppp** keyword.

encapsulation ppp

Example

These commands enable PPP encapsulation on serial interface 0 (zero):

```
interface serial 0
encapsulation ppp
```

Challenge Handshake Access Protocol (CHAP)

Access control using CHAP is available on all serial interfaces. The authentication feature reduces the risk of security violations on your router. CHAP is only supported on lines using PPP encapsulation.

When CHAP is enabled, a remote device (a PC, workstation, router, or communication server) attempting to connect to the local communication server is requested, or "challenged," to respond. The required response is an encrypted version of a secret password, or "secret," plus a random value and the name of the remote device. This name must be configured as described in the Configuring Host Name Authentication section later in this chapter.

By transmitting this response, the secret is never transmitted, preventing other devices from stealing it and gaining illegal access to the system. Without the proper response, the remote device cannot connect to the local router.

CHAP transactions occur only at the time a link is established. The local router does not request a password during the rest of the call. (The local router can, however, respond to such requests from other devices during a call.)

The DECbrouter 90 implementation of CHAP does not support the callback option. (The callback option requires that the called router hang up and return the call to a preconfigured telephone number.)

To use CHAP, you must perform the following steps. (These steps are described later in this section.)

1. Enable CHAP on the interface.
2. Configure server authentication.

Note

To use CHAP, you must be running PPP encapsulation.

CHAP is specified in the IETF draft "The PPP Authentication Protocols" by Brian Lloyd of Lloyd and Associates and William A. Simpson of Computer Systems Consulting Services. The latest version is dated December 1991.

CHAP is specified as an additional authentication phase of the PPP link control protocol.

Enabling CHAP on the Interface

To enable CHAP on the interface, use this interface command:

ppp authentication chap

Once you have enabled CHAP, the local router requires a password from remote devices. If the remote device does not support CHAP, no traffic will be passed to that device.

Configuring Host Name Authentication

Configure the secret using the following command:

username *name* **password** *secret*

Add a username entry for each remote system that the local router communicates with and requires authentication from.

The name argument is the host name of either the local router or a remote device.

Note

To enable the local router to respond to remote CHAP challenges, one username name entry must be the same as the **hostname** *name* that has already been assigned to your router. See Setting the Host Name in Chapter 4.

The *secret* argument specifies the secret for the local router or the remote device. If there is no secret specified, and DEBUG SERIAL-INTERFACE is enabled, an error is displayed when a link is established, and the CHAP challenge is not implemented. Debugging information on CHAP is available using the DEBUG SERIAL-INTERFACE and DEBUG SERIAL-PACKET commands. See the Interface Support EXEC Command Summary for a description of these debugging commands.

The secret is encrypted when it is stored on the local router. This prevents it from being stolen. It can consist of any string of up to 11 printable ASCII characters. There is no limit to the number of username/password combinations that can be specified, allowing any number of remote devices to be authenticated using CHAP.

Example

The following example configuration enables CHAP on interface serial 0. It also defines a password for the local server, " "XXX" and a remote server, "YYY."

```
hostname XXX
interface serial 0
encapsulation ppp
ppp authentication chap
username XXX password oursystem
username YYY password theirsystem
```

When you look at your configuration file, the passwords will be encrypted and the display will look something like the following:

```
hostname XXX
interface serial 0
encapsulation ppp
ppp authentication chap
username XXX password 7 1514040356
username YYY password 7 121F0A18
```

Example:Dial-on-Demand PPP Configuration

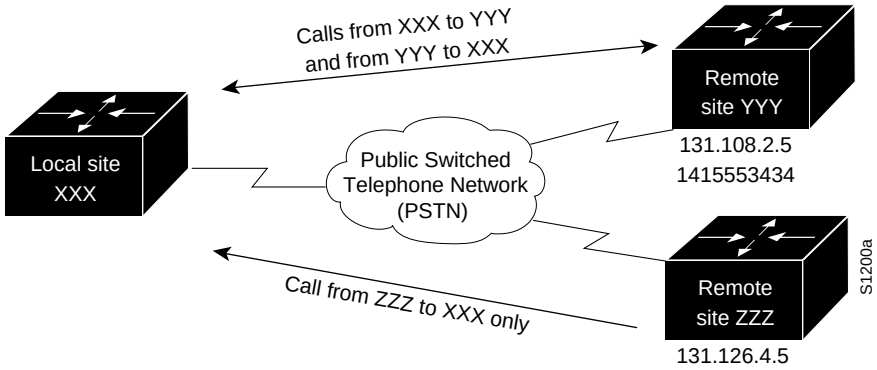
The following example shows a configuration in which several aspects of DDR are used to provide DDR capabilities between local and remote routers. The following features are shown in this example.

- DIALER MAP command
- PPP encapsulation
- CHAP
- Dialer rotary groups
- PULSE-TIME command (See Chapter 8 for a description of this command.)

```
! Enable PPP encapsulation and CHAP on interface dialer 1.
interface dialer 1
encapsulation ppp
ppp authentication chap

! Specify dial-on-demand routing supported on a synchronous serial line and
! assign a set of access-list expressions.
dialer in-band
dialer group 1

! The first dialer map command indicates that calls between the remote site
! YYY and the central site will be placed at either end. The second dialer
! map command, with no dialer string, indicates that remote
site ZZZ will call
! the central site but the central site will not call out.
dialer map ip 131.108.2.5 username YYY 1415553434
dialer map ip 131.126.4.5 username ZZZ

! The DTR pulse holds the DTR low for three seconds, so the
modem can recognize
! that DTR has been dropped.
pulse-time 3

! Place serial interfaces 0 and 1 in dialer rotary group 1. The
! interface subcommands applied to dialer rotary group 1 (for example,
! PPP encapsulation and CHAP) apply to these interfaces.
interface serial 0
dialer rotary-group 1
interface serial 1
dialer rotary-group 1

! CHAP passwords are specified for local and remote servers.
username XXX password oursystem
username YYY password theirsystem
username ZZZ password thatsystem
```

Figure 7–3 shows a configuration in which remote routers YYY and ZZZ and local router XXX are using dial-on-demand routing, as configured in the previous example. In this configuration, remote routers YYY and ZZZ can call router XXX. Router XXX has dialer string information only for router YYY and cannot call router ZZZ.

Figure 7–3 Dial-on-Demand Routing Configuration



Configuring the Null Interface

Support is provided for a null interface. This pseudo-interface functions similarly to the null devices available on most operating systems. This interface is always up and can never forward or receive traffic; encapsulation always fails.

The null interface provides an alternative method of filtering traffic. The overhead involved with using access lists can be avoided by directing undesired network traffic to the null interface.

To specify the null interface, specify null 0 (or null0) as the interface name and unit. The null interface can be used in any command that has an interface type as a parameter.

Example

This command configures a null interface for IP route 127.0.0.0:

```
ip route 127.0.0.0 null 0
```

Global Configuration Command Summary

The following summary lists the global configuration commands used for interface support.

[**no**] **dialer-list** *dialer-group* **list** *list-number*
[**no**] **dialer-list** *dialer-group* **protocol** *protocol-name* {*permit* | *deny*}

Controls automatic dialing using DDR using standard IP or bridging access lists.

The argument *dialer-group* specifies the number of a dialer group identified in any DIALER-GROUP interface subcommand.

The argument *list-number* is the access list number specified in any IP or bridging access list.

The argument *protocol-name* is one of the following supported protocols:

- **ip**—IP
- **decnet**—DECnet

- **chaos**—CHAOSnet
- **xns**—XNS
- **novell**—Novell IPX
- **appletalk**—AppleTalk
- **vines**—Vines
- **apollo**—Apollo Domain
- **pup**—PUP
- **clns**—OSI Connectionless Network Service
- **bridge**—Bridging
- **stun**—Serial tunneling
- **cmns**—OSI Connection-Mode Network Service

interface *type unit*

Specifies an interface. The argument *type* specifies the interface type—**ethernet** or **serial**—and the argument *unit* specifies the interface number.

Interface Support Subcommand Summary

Following are alphabetically arranged summaries of the interface subcommands for interface support.

[**no**] **backup delay** {*enable-delay* | **never**} {*disable-delay* | **never**}

Defines how much time should elapse before a line is set up or taken down. The argument *enable-delay* is the delay in seconds after the primary line goes down before the secondary line is activated. The argument *disable-delay* is the delay in seconds after the primary line goes up before the secondary line is deactivated. The **never** keyword prevents the secondary line from being activated due to a primary line failure. The **no** version disables the specified delay time.

[**no**] **backup interface** *interface-name*

Configures the serial interface as a secondary, or dial backup, line. The argument *interface-name* specifies the serial port to be set as the secondary interface line. Use the NO BACKUP interface command with the appropriate serial port designation to turn this feature off.

[**no**] **backup load** {*enable-threshold* | **never**} {*disable-load* | **never**}

Sets the traffic load thresholds. The arguments *enable-threshold* and *disable-load* are percentage numbers representing the load as a percentage of the primary line's available bandwidth. The never keyword prevents the secondary line from being activated due to load. By default, no backup loads are defined; the **no** form of the command restores this default.

[**no**] **description** *name-string*

Adds a descriptive name to an interface. The argument *name-string* is a comment to be put in the configuration. The **no** version removes the *name-string*.

[**no**] **dialer enable-timeout** *number-of-seconds*

Sets how long an interface stays down before it is available to dial again. Acceptable values are positive, nonzero integers. If no value is entered, the default is 15 seconds.The NO DIALER ENABLE-TIMEOUT command resets the enable timeout value to the default.

[**no**] **dialer fast-idle** *number-of-seconds*

Specifies the idle time before the line is disconnected for interfaces for which there is an unusually high level of contention. Acceptable values are positive, nonzero integers. If no value is entered, the default is 20 seconds. This timer is used when an interface is connected and a packet is received for a different destination. This command only applies if multiple destinations have been specified for a given interface using the DIAL MAP interface subcommand. The NO DIALER FAST-IDLE resets the timeout period to the default.

[**no**] **dialer-group** *group-number*

Assigns an interface to a set of access list expressions. These access list expressions define what packets cause a connection to be established and what packets keep an interface from being idle.

The argument *group-number* specifies the number of the dialer group to which the specific interface belongs. Acceptable values are nonzero, positive integers. This group number corresponds to a dialer group defined using the DIALER-LIST command (described below).

An interface can only be associated with a single **dialer-group**.

The NO DIALER-GROUP command removes an interface from the specified dialer-group.

[**no**] **dialer idle-timeout** *number-of-seconds*

Specifies the idle time before the line is disconnected.

The argument *number-of-seconds* specifies in seconds how much idle time must occur on an interface before the line is disconnected. Acceptable values are positive, nonzero integers. If no value is entered, the default is 120 seconds.

The NO DIALER IDLE-TIMEOUT command resets the idle timeout to the default.

[**no**] **dialer in-band**

Specifies that V.25bis dialing is to be used on the specified interface. This is only used with serial interfaces.

The NO DIALER IN-BAND command disables dial-on-demand routing for the interface.

dialer map *protocol next-hop-address* **username** *name*
dialer map *protocol next-hop-address dial-string*
dialer map *protocol next-hop-address* **username** *name dial-string*

Three forms of the interface subcommand can be used to define multiple dial-on-demand numbers for a particular interface. The first form of the dialer map command is used in central-site configurations in which remote sites are calling a central site, but the central site is not calling the remote site. The second form of the command is used if only one site is being called. The third form is used if many remote sites are calling the central site and the central site also can call the remote sites.

The keyword **username** indicates a remote system that the local router communicates with.

The following arguments can be used with this command:

- The argument *protocol* has only one option at this time: **ip**.
- The argument *next-hop-address* specifies the IP address used to match against addresses to which packets are destined.
- The argument name is the name of the remote device.
- The *dial-string argument* is the telephone number sent to the DCE dialing device when it sees packets with the specified *next-hop-address* that match the access lists defined.

The NO DIALER MAP command deletes a particular dialer map entry.

dialer rotary-group *n*

Includes an interface in a dialer rotary group. The argument *n* is the number of the rotary group (defined by the INTERFACE DIALER command).

[**no**] **dialer string** *dial-string*

Specifies the string (i.e., telephone number) to be passed to the DCE device (typically a V.25bis modem).

The argument *dial-string* specifies the character string to be passed to the V.25bis DCE device. The DECbrouter 90 places the V.25bis command CRN in front of the specified string. One DIALER STRING command is specified per interface.

The NO DIALER STRING command deletes the dialer string specified for the interface.

[**no**] **dialer wait-for-carrier-time** *number-of-seconds*

Specifies how long to wait for the carrier to come up when a call is placed. Acceptable values are positive, nonzero integers. If no value is entered, the default is 30 seconds. If a carrier signal is not detected in this amount of time, the interface is disabled until the enable timeout occurs.

The wait-for-carrier time is set to a default of 30 seconds since this is the estimated time required to make a typical connection of telephone lines.

The NO DIALER WAIT-FOR-CARRIER-TIME command resets the carrier wait time value to the default.

encapsulation *encapsulation-type*

Assigns an encapsulation method. The *encapsulation-type* argument is a keyword that identifies one of the following encapsulation types that the software supports:

- **arpa**—Ethernet Version 2.0 encapsulation
- **bfex25**—Blacker front end encryption X.25 operation
- **ddnx25-dce**—DDN X.25 DCE operation
- **ddnx25**—DDN X.25 DTE operation
- **frame-relay**—Frame relay
- **hdh**—HDH protocol
- **hdlc**—HDLC protocol
- **iso1**—IEEE 802.3 encapsulation
- **lapb-dce**—X.25 LAPB DCE operation
- **lapb**—X.25 LAPB DTE operation
- **multi-lapb-dce**—X.25 LAPB multiprotocol DCE operation

- **multi-lapb**—X.25 LAPB multiprotocol DTE operation
- **ppp**—Point-to-point protocol (PPP)
- **snap**—IEEE 802.2 Ethernet media
- **smds**—SMDS service
- **x25-dce**—X.25 DCE operation
- **x25**—X.25 DTE operation

ppp authentication chap

Enables CHAP on an interface. Once you have enabled CHAP, the local router requires a password from remote devices. If the remote device does not support CHAP, no traffic will be passed to that device.

[**no**] **shutdown**

Disables and enables an interface.

username *name* **password** *secret*

Adds a **username** entry for each remote system that the local router communicates with and requires authentication from. The name argument is the host name of either the local router or a remote device. The *secret* argument specifies the secret for the local router or the remote device.

Interface Support EXEC Command Summary

Following is an alphabetically arranged summary of the EXEC interface support commands.

clear counters [*interface-name*]

The argument *interface-name* is the name of the interface (examples are serial 0, Ethernet 1, and so on).

clear interface [*type unit*]

Resets the hardware logic on an interface. Argument *type* is the specific interface type keyword (**ethernet** or **serial**); the argument *unit* is the interface number (0 or 1).

[un]debug broadcast

Enables you to log all level 2 (MAC) broadcast packets received. This information is useful for finding the source of a broadcast storm. Use the **undebug** version of the command to stop **debug** messaging.

[un]debug packet

Enables logging of packets that the router is unable to classify. Examples of this are packets with an unknown Ethernet link type, or IP packets with an unrecognized protocol field.

[un]debug serial-interface

Enables or disables logging of serial interface events.

[un]debug serial-packet

Enables logging of serial interface events. Provides more detail than **debug serial-interface**.

show dialer [**interface** *interface type*]

Displays information the dialer interfaces.

show interfaces [*type unit*] [**accounting**]

Displays statistics for the network interfaces on the DECbrouter 90. The optional argument *type* can be one of the following: **ethernet** or **serial**. The argument *unit* specifies the interface unit or card number. The optional keyword **accounting** displays the number of packets of each protocol type that have been sent through the interface.

show version

Displays the configuration of the system hardware, the software version, the names and sources of configuration files, and the boot images.

8

Adjusting Interface Characteristics

This chapter describes how to make adjustments to the router interfaces and how to enable loopback tests. The tasks in this chapter include:

- Switching and scheduling priorities
- Priority output queuing
- Interface hold queues
- Setting the bandwidth
- Setting delay
- Setting loopback testing on an interface

To adjust or test an interface, you must be in the configuration command collection mode. To enter this mode, enter the EXEC command CONFIGURE at the EXEC prompt. Once in the command collection mode, start configuring the interface by entering the INTERFACE command.

A summary of the interface subcommands described in this chapter is included at the end of the chapter.

Adjusting Serial Interface Characteristics

The following sections describe adjustments that can be made to serial interfaces.

Specifying Transmit Delay

The DECbrouter 90 serial interface may send back-to-back data packets faster than hosts can receive them. You can slow the interface card's transmit rate by using the transmitter-delay interface subcommand. The full syntax of the command follows.

transmitter-delay *hdlc-flags*
no transmitter-delay

The argument *hdlc-flags* causes a minimum number of HDLC flags to be sent between each packet. The valid range is 2 to 62; the default is 0.

Configuring DTR Signal Pulsing

The PULSE-TIME interface subcommand enables pulsing DTR signals on the DECbrouter 90 serial interfaces. The full syntax of the command follows.

pulse-time *seconds*
no pulse-time

The argument *seconds* specifies the interval. When the serial line protocol goes down (for example, because of loss of synchronization), the interface hardware is reset and the DTR signal is held inactive for at least the specified interval. This

function is useful for handling encrypting or other similar devices that use DTR signal toggling to resynchronize.

This command also is used to specify the time allowed for a modem to recognize that DTR has dropped so that the modem can disconnect the call.

The default interval is zero seconds, which is restored by the NO PULSE-TIME command.

Example

This command enables DTR pulse signals for three seconds on interface serial 2.

```
interface serial 2
pulse-time 3
```

Adjusting Characteristics That Apply to All Interface Types

The following sections describe adjustments that apply to all interface types.

Configuring Switching and Scheduling Priorities

The normal operation of the DECbrouter 90 allows the switching operations to use as much of the central processor as is required. If the network is running unusually heavy loads that do not allow the processor the time to handle the routing protocols, give priority to the system process scheduler using the SCHEDULER-INTERVAL global configuration command. The full command syntax follows.

scheduler-interval *milliseconds*
no scheduler-interval

The SCHEDULER-INTERVAL command controls the maximum amount of time that can elapse without running the lowest-priority system processes. The minimum interval that can be specified is 500 milliseconds; there is no maximum value. The default is to allow high-priority operations to use as much of the central processor as needed. The NO SCHEDULER-INTERVAL command restores that default.

Example

This command changes the low-priority process schedule to an interval of 750 milliseconds.

```
scheduler-interval 750
```

Priority Output Queuing

Priority output queuing is a mechanism that allows the administrator to set priorities on the type of traffic passing through the network. To do so, it prioritizes the packets transmitted on an interface. Packets are classified according to various criteria, including protocol and subprotocol type, and then queued on one of four output queues.

When the DECbrouter 90 is ready to transmit a packet, it scans the priority queues in order, from highest to lowest, to find the highest-priority packet. After that packet is completely transmitted, the router scans the priority queues again. If a priority output queue fills up, packets will be dropped and, for protocols such as IP and CLNS, quench indications will be sent to the original transmitter.

Although priority queuing can be enabled for any interface, the intended application was for low-bandwidth, congested serial interfaces.

The priority output queuing mechanism allows traffic control based on protocol or interface type. Commands also are offered to set the size of the queue and defaults for what happens to packets that are not defined by priority output queue rules.

The priority output queuing mechanism can be used to manage traffic from all networking protocols and from bridges. Additional fine-tuning is available for IP and for setting boundaries on the packet size.

Note

Priority queuing introduces extra overhead that is acceptable for slow interfaces but may not be acceptable for higher-speed interfaces such as Ethernet. Also note that priority queuing does not operate over X.25.

There are four priority queues—high, medium, normal, and low—listed in order from highest to lowest priority.

Keepalives sourced by the router always are assigned to the high-priority queue; all other management traffic (such as IGRP updates) must be configured. Packets that are not classified by the priority list mechanism are assigned to the normal queue.

A priority list is a set of rules that describes how packets should be assigned to priority queues. A priority list also can describe a default priority or the queue size limits of the various priority queues.

Assigning Priority by Protocol Type

Use the PRIORITY-LIST global configuration command to establish queuing priorities based upon the protocol type. The full syntax of this command follows.

priority-list *list* **protocol** *protocol-name queue-keyword [args]*
no priority-list *list* **protocol** *protocol-name queue-keyword*

The argument *list* is an arbitrary integer between one and ten that identifies the priority list selected by the user.

The keyword **protocol** is used with the argument *protocol-name* to specify the protocol you are using and is one of the following: **ip, pup, chaos, xns, decnet, appletalk, clns, novell, apollo, vines, stun** (for serial tunneling), or **bridge** (for transparent bridging traffic).

The argument *queue-keyword* is a priority queue name; it can be **high, medium, normal,** or **low**.

Optional arguments (*args*) can be specified, depending on the *protocol-name* keyword, as follows:

- **gt** *byte-count*—Specifies a greater-than count. The priority level assigned goes into effect when a packet exceeds the value entered for the argument *byte-count*. The size of the packet also must include additional bytes because of MAC encapsulation on the outgoing interface.
- **lt** *byte-count*—Specifies a less-than count. The priority level assigned goes into effect when a packet size is less than the value entered for *byte-count*. The size of the packet also must include additional bytes due to MAC encapsulation on the outgoing interface.
- **bridge list** *list-number*—Assigns the priority level to bridged traffic according to the access list number using the bridge and **list** keywords. The *list-number* argument is the Ethernet-type code access list number assigned by the **access-list** global configuration command and the access-group list interface subcommand.
- **list** *list-number*—Assigns traffic priorities according to a specific list. The *list-number* argument is the IP access list number assigned by the access-group list interface subcommand. (For use with the IP protocol only.)
- **tcp port**—Assigns the priority level defined to TCP segments originating from or destined to a specified port (for use with the IP protocol only). Table 8–1 lists common TCP services and their port numbers.

Table 8–1 Common TCP Services and Their Port Numbers

Service	Port
Telnet	23
SMTP	25

- **udp** *port*—Assigns the priority level defined to UDP packets originating from or destined to the specified port (for use with the IP protocol only). Table 8–2 lists common UDP services and their port numbers.

Table 8–2 Common UDP Services and Their Port Numbers

Service	Port
TFTP	69
NFS	2049
SNMP	161
RPC	111
DNS	53

Note

The TCP and UDP ports listed in Table 7-1 and Table 7-2 include some of the more common port numbers. However, you can specify any port number to be prioritized; you are not limited to those listed.

Use the NO PRIORITY-LIST global configuration command with the appropriate arguments to remove an entry from the list.

Examples

This command assigns one as the arbitrary priority list number, specifies DECnet as the protocol type, and assigns a high priority to the packets transmitted on this interface.

```
priority-list 1 protocol decnet high
```

When classifying a packet, the system searches the list of rules specified by PRIORITY-LIST commands for a matching protocol type. When a match is found, the packet is assigned to the appropriate queue. The list is searched in the order it is specified, and the first matching rule terminates the search.

This command assigns a medium priority level to every DECnet packet with a size greater than 200 bytes.

```
!
priority-list 2 protocol decnet medium gt 200
!
```

This command assigns a medium priority level to every DECnet packet with a size less than 200 bytes.

```
!
priority-list 4 protocol decnet medium lt 200
!
```

This command assigns a high priority to traffic that matches IP access list 10.

```
!
priority-list 1 protocol ip high list 10
!
```

This command assigns a medium priority level to Telnet packets.

```
!
priority-list 4 protocol ip medium tcp 23
!
```

This command assigns a medium priority level to UDP Domain Name service packets.

```
!
priority-list 4 protocol ip medium udp 53
!
```

This command assigns a high priority to traffic that matches Ethernet type code access list 201.

```
!
priority-list 1 protocol bridge high list 201
!
```

When using multiple rules for a single protocol, remember that the order of the rules matters.

Assigning Priority by Interface Type

Use this variation of the PRIORITY-LIST global configuration command to establish queuing priorities on packets entering from a given interface (full syntax follows):

priority-list *list* **interface** *interface-name queue-keyword*
no priority-list *list interface interface-name queue-keyword*

The argument *list* is an arbitrary integer between one and ten that identifies the priority list selected by the user.

The keyword **interface** applies to the priority queuing mechanism to packets arriving from an interface. The argument *interface-name* specifies the name of the interface.

The argument *queue-keyword* is a priority queue name; it can be **high, medium, normal,** or **low**.

Use the NO PRIORITY-LIST command with the appropriate arguments to remove an entry from the list.

Example

The following sample command sets any packet type entering on interface Ethernet 2 to a medium priority:

```
!
priority-list 3 interface ethernet 2 medium
!
```

Assigning a Default Priority

Use the following global configuration command to assign a priority queue for those packets that did not match any other rule in the priority list:

priority-list *list* **default** *queue-keyword*
no priority-list *list* **default**

If you do not specify a default or use the **no** form of the command, the **normal** queue is assumed.

Specifying the Maximum Packets in the Priority Queues

Use this variation of the PRIORITY-LIST global configuration command to specify the maximum number of packets that can be waiting in each of the priority queues (full syntax shown):

priority-list *list* **queue-limit** *high-limit medium-limit normal-limit low-limit*
no priority-list *list* **queue-limit**

If a priority queue overflows, excess packets are discarded and quench messages can be sent, if appropriate, for the protocol. The default **queue-limit** arguments are 20 packets for the **high** priority queue, 40 for **medium**, 60 for **normal**, and 80 for **low**.

The NO PRIORITY-LIST command with the appropriate keywords and arguments restores these defaults.

Example

The following is a sample of a priority list, including the access list referenced by one of the priority list rules:

```
priority-list 1 protocol bridge high list 201
priority-list 1 protocol ip medium
priority-list 1 protocol decnet medium
priority-list 1 default low
priority-list 1 queue-limit 20 20 20 10
!
access-list 201 permit  0x6004  0x0000
!
```

Bridged traffic that matches access list 201 is given high priority. Because type code 6004 is the local area transport (LAT), this rule assigns LAT traffic high priority. IP and DECnet traffic are given medium priority, and everything else is given low priority. In this instance, the network administrator has assigned queue limits of 20 packets for every queue except the low-priority queue, which is limited to 10 untransmitted packets.

Assigning a Priority Group to an Interface

Use the PRIORITY-GROUP interface subcommand to assign the specified priority list to an interface.

priority-group *list*
no priority-group *list*

The argument *list* is the priority list number assigned to the interface. Multiple lists can be assigned per interface.The **no** version of this command removes a specific **priority-group** assignment if it includes a list argument. It removes all **priority-group** assignments if specified without the *list* argument.

Example

This example causes packets exiting interface serial 0 to be classified by priority list 1.

```
interface serial 0
priority-group 1
```

Monitoring the Priority Queuing Lists

When priority queuing is enabled on an interface, the EXEC command SHOW INTERFACES displays information about the input and output queues. This information is a triplet of numbers: the first is the number of packets currently in the queue, the second is the maximum number of packets permitted in the queue, and the third is the number of packets discarded because the queue is full.

```
Input queue: 0/75/4 (size/max/drops); Total output drops: 0
 Output queue: high 0/20/0, medium 0/40/00, normal 0/60/0, low 0/80/0
```

The total *output drops* number is the sum of the discarded numbers from the output queues, plus the count of packets discarded before being prioritized, or because of fast-switching activity.

Assigning Priority by Serial Link Address

Use the PRIORITY-LIST global configuration command to establish queuing priorities based on the address of the serial link on a STUN connection. The full syntax of this command follows.

priority-list *list* **stun** *queue-keyword address group-number address-number*
no priority-list *list* **stun** *queue-keyword address group-number address-number*

The argument *list* is an arbitrary integer between 1 and 10 that identifies the priority list selected by the user.

The keyword **stun** is used to indicate this is a serial tunnel feature.

The argument *queue-keyword* is a priority queue name; it can be **high, medium, normal** or **low**.

The keyword **address** is used with the two arguments *group-number* and *address-number* which uniquely identifies a STUN serial link. The argument *group-number* is the group number used in the STUN GROUP interface configuration subcommand. The argument *address-number* is the address of the serial link. The format of the address number is either a one-byte hex value (for example, C1) for an SDLC link or one that is specified by the STUN SCHEMA configuration command.

Controlling Interface Hold Queues

Each network interface in a router has a hold-queue limit. This limit is the number of data packets that the interface can store in its hold queue before rejecting new packets. When the interface empties one or more packets from the hold queue, the interface can accept new packets again.

To specify the hold-queue limit of an interface, use the HOLD-QUEUE interface subcommand. Its full syntax is as follows:

hold-queue *length* {**in** | **out**}
no hold-queue {**in** | **out**}

The argument *length* is the maximum number of packets in the queue. The **in** keyword specifies the input queue; the **out** keyword specifies the output queue.

The input hold queue prevents a single interface from flooding the router with too many input packets. Further input packets are discarded if the interface has too many input packets outstanding in the system. The default input hold queue is 75 packets. The default output hold-queue limit is 40 packets. This limit prevents a malfunctioning interface from consuming an excessive amount of memory.

The NO HOLD-QUEUE command with the appropriate keyword restores the default values for an interface. There is no fixed upper limit to a queue size.

If priority output queuing is being used, the length of the four output queues is set using the PRIORITY-LIST global configuration command. The HOLD-QUEUE command cannot be used to set an output hold queue length in this situation.

For slow links, use a small output hold-queue limit. This approach prevents storing packets at a rate that exceeds the link's transmission capability. For fast links, use a large output hold-queue limit. A fast link may be busy for a short time (and thus require the hold queue), but can empty the output hold queue quickly when capacity returns.

To display the current hold queue setting and the number of packets discarded because of hold queue overflows, use the EXEC command SHOW INTERFACES.

Note

Increasing the hold queue can have detrimental effects on network routing and response times. For protocols that use seq/ack packets to determine round-trip times, increasing the output queue is a very bad idea. Dropping packets is not necessarily a bad thing—it informs hosts to slow down transmissions to match available bandwidth. This is generally better than having duplicate copies of the same packet within the network (which can happen with large hold queues).

Setting Bandwidth

Higher-level protocols can use bandwidth information to make operating decisions. For example, IGRP uses the minimum path bandwidth to determine a routing metric. The TCP protocol adjusts initial retransmission parameters based on the apparent bandwidth of the outgoing interface.

To set a bandwidth value for an interface, use the BANDWIDTH interface subcommand. Its full syntax is as follows:

bandwidth *kilobits*
no bandwidth

Note

This is a routing parameter only; it does not affect the **physical** interface.

The argument *kilobits* specifies the intended bandwidth in kilobits per second. For a full bandwidth DS3, enter the value 44736. Default bandwidth values are set during startup and can be displayed with the EXEC command SHOW INTERFACES.

The BANDWIDTH subcommand sets an informational parameter only; you cannot adjust the actual bandwidth of an interface with it. For some media, such as Ethernet, the bandwidth is fixed; for other media, such as serial lines, you can change the actual bandwidth by adjusting hardware. For both classes of media, you can use the BANDWIDTH subcommand to communicate the current bandwidth to the higher-level protocols.

Use the NO BANDWIDTH command to restore the default values.

Setting Delay

As mentioned, higher-level protocols can use delay information to make operating decisions. For example, IGRP can use delay information to differentiate between a satellite link and a land link.

To set a delay value for an interface, use the DELAY interface subcommand. The full syntax of this command follows.

delay *tens-of-microseconds*
no delay

The argument *tens-of-microseconds* specifies the delay for an interface or network segment in tens of microseconds. Default delay values can be displayed with the EXEC command SHOW INTERFACES. The NO DELAY command restores the default.

Note

The DELAY subcommand sets an informational parameter only; you cannot adjust the actual delay of an interface with this subcommand.

Setting and Adjusting Packet Sizes

Each interface has a default maximum packet size or maximum transmission unit (MTU) size. This number generally defaults to the largest size possible for that type interface. On serial interfaces, the MTU size varies, but cannot be set smaller than 64 bytes. You can adjust the MTU using the MTU interface subcommand.

mtu *bytes*
no mtu

The arguments *bytes* is the desired size in bytes. The NO MTU command restores this value to its original default value. Table 8–3 lists default MTU values according to media type.

Table 8–3 Default Media MTU Values

Media Type	Default MTU
Ethernet	1500
Serial	1500

Example

The following is an example of specifying the MTU on serial port to 300 bytes:

```
interface serial 1
mtu 1000
```

Note

Changing the MTU value with the MTU interface subcommand can affect values for the protocol-specific versions of the command IP MTU and CLNS MTU. If the current values specified with the IP MTU or CLNS MTU interface subcommands are the same as the value specified with the MTU interface subcommand, then when you change the value for the MTU interface subcommand, the values for IP MTU and CLNS MTU are automatically modified to match the new **mtu** interface subcommand value. However, the reverse is not true. In other words, changing the values for the IP MTU or CLNS MTU subcommands has no effect on the value for the MTU interface subcommand.

Enabling the Loopback Test

The DECbrouter 90 software provides a loopback test to detect and distinguish equipment malfunctions between line and modem or CSU/DSU (channel service unit/digital service unit) problems on the router. If correct data transmission is not possible when an interface is in loopback mode, the interface is the source of the problem. The DSU may have similar loopback functions you can use to isolate the problem if the interface loopback tests passes. If the device does not support local loopback, this function will have no effect.

You can specify hardware loopback tests on Ethernet and serial interfaces that are attached to CSU/DSUs and support the local loopback signal. The CSU/DSU acts as a DCE device; the router as a DTE device. The local loopback test generates a CSU loop—a signal that goes through the CSU/DSU to the line, then back through the CSU/DSU to the router.

The PING command also can be useful during loopback operation; it is described in Testing Connectivity with the PING Command in Chapter 6.

The following sections describe the various loopback tests available for the supported interfaces.

Enabling Loopback on the Serial Interfaces

The serial interface supports the loopback function when a CSU/DSU or equivalent device is attached to the router. Use the LOOPBACK interface subcommand to enable loopback on the interface. The full syntax of this subcommand follows.

loopback
no loopback

The LOOPBACK interface subcommand loops the packets through the CSU /DSU to configure a CSU loop when the device supports this feature. The NO LOOPBACK subcommand disables the function.

Enabling Loopback on the Ethernet Interface

The DECbrouter 90 Ethernet interface cannot be placed into loopback mode.

Global Configuration Subcommand Summary

This section provides an alphabetical list of all the global configuration commands described in this chapter.

[**no**] **priority-list** *list* **default** *queue-keyword*

Assigns a priority queue for those packets that did not match any other rule in the priority list. If no default or the **no** form of the command is specified, the **normal** queue is assumed.

[**no**] **priority-list** *list* **interface** *interface-name queue-keyword*

Sets up priority queuing on the specified interface. The keyword **interface** applies to the priority queuing mechanism to packets arriving from an interface. The argument *interface-name* specifies the name of the interface. The arguments *list* and *queue-keyword* are described in the section Assigning Priority by Interface Type. The **no** form of the command removes the item from the list.

[**no**] **priority-list** *list protocol protocol-name queue-keyword [args]*

Sets up priority queuing on the specified interface. The keyword **protocol** applies to the protocol you are using. The argument *protocol-name* specifies the name of the protocol. The arguments *list*, *queue-keyword*, and the optional *args* are described in the section Assigning Priority by Protocol Type. The **no** form of the command removes the item from the list.

[**no**] **priority-list** *list* **queue-limit** *high-limit medium-limit normal-limit low-limit*

Specifies the maximum number of packets that can be waiting in each of the priority queues. If a priority queue overflows, excess packets are discarded and quench messages can be sent, if appropriate, for the protocol. The **no** form of the command resets all four queue sizes to their default values as follows: *high-limit* = 20; *medium-limit* = 40; *normal-limit* = 60; *low-limit* = 80.

[**no**] **scheduler-interval** *milliseconds*

Controls the maximum amount of time that can elapse without running the lowest-priority system processes. The minimum interval that can be specified is 500 milliseconds; there is no maximum value. The default is to allow high-priority operations to use as much of the central processor as needed. The NO SCHEDULER-INTERVAL command restores that default.

Interface Configuration Subcommand Summary

This section provides an alphabetical list of all the interface commands described in this chapter.

[**no**] **bandwidth** *kilobits*

Sets a bandwidth value for an interface. The argument *kilobits* specifies the intended bandwidth in kilobits per second. Default bandwidth values are set during startup and can be displayed with the EXEC command SHOW INTERFACES. This is a routing parameter only; it does not affect the *physical* interface. The **no** form of the subcommand restores the default.

[**no**] **delay** *tens-of-microseconds*

Sets a delay value for an interface. The argument *tens-of-microseconds* specifies the delay for an interface or network segment in tens of microseconds. Default delay values can be displayed with the EXEC command SHOW INTERFACES. The **no** form of the subcommand restores the default.

Note

The DELAY subcommand sets an informational parameter only; you cannot adjust the actual delay of an interface with this subcommand.

hold-queue *length* {**in** | **out**}
no hold-queue {**in** | **out**}

Specifies the hold-queue limit of an interface. The argument *length* is the maximum number of packets in the queue. The **in** keyword specifies the input queue; the **out** keyword specifies the output queue. There is no fixed upper limit to a queue size. The default for **in** is 75; the default for **out** is 40. The **no** keyword restores the default values for an interface.

[**no**] **loopback**

On the serial interfaces—loops the packets through the CSU/DSU to configure a "CSU loop," when the device supports this feature.

The **no** form of the subcommand disables the loopback test.

[**no**] **mtu** *bytes*

Adjusts the maximum transmission unit (MTU). The arguments *bytes* is the desired size in bytes. The NO MTU subcommand restores this value to its original default value.

[**no**] **priority-group** *list*

Assigns the specified priority group to an interface. The argument *list* is the priority list number assigned to the interface. The **no** version of this subcommand removes a specific **priority-group** assignment if it includes a list argument. It removes all **priority-group** assignments if specified without the *list* argument.

[**no**] **pulse-time** *seconds*

Enables pulsing DTR signals on the serial interfaces for a minimum interval of *seconds*. The **no** version of the subcommand disables pulsing.

[**no**] **transmitter-delay** *hdlc-flags*

The argument *hdlc-flags* causes a minimum of HDLC flags to be sent between each packet. The valid range is 2 to 62. The **no** form of the subcommand restores the default value of zero flags.

9

Configuring Packet-Switched Software

This chapter describes the configuration tasks for packet-switched software. These tasks include configuring the following protocols and services:

- Recommendation X.25, including link access procedure, balanced (LAPB)
- Connection-mode network service (CMNS) support to extend X.25 switching support over LAN interfaces, as defined in ISO Standards 8208 (packet level) and 8802-2 (frame level)
- Frame relay service
- Switched multimegabit data services (SMDS)

Summaries of the configuration commands available for these protocols and services are provided at the end of this chapter.

Configuring LAPB

X.25 Level 2, or LAPB, is a data encapsulation protocol that operates at level 2 (the data link level) of the OSI reference model. LAPB specifies methods for exchanging data (in units called *frames*), detecting out-of-sequence or missing frames, retransmitting frames, and acknowledging frames.

It is possible to only use LAPB as a serial encapsulation method. This can be done using a leased serial line. You must use one of the X.25 packet-level encapsulations when attaching to an X.25 network.

Using LAPB under noisy conditions can result in greater throughput than HDLC encapsulation. When LAPB detects a damaged frame, the router immediately retransmits the frame instead of waiting for host timers to expire. However, this behavior is good only if the host timers are relatively slow. In the case of quickly expiring host timers, LAPB spends much of its time retransmitting host retransmissions.

If the line is not noisy, the lower overhead of HDLC encapsulation is more efficient than LAPB. When using long-delay satellite links, the lock step behavior of LAPB makes the use of HDLC encapsulation the better choice.

The X.25 Recommendation distinguishes between two types of X.25 hosts: data terminal equipment (DTE) in LAPB encapsulation host and data circuit-terminating equipment (DCE) in LAPB encapsulation hosts.

A router using LAPB encapsulation can act as a DTE or DCE device at the protocol level.

Running a Single Network Protocol

To run datagrams of a single protocol over a serial interface using the LAPB encapsulation, use the interface subcommand:

encapsulation {**lapb** | **lapb-dce**}

The keyword **lapb** sets DTE operation; the keyword **lapb-dce** sets DCE operation. One end of the link must be DTE and the other must be DCE. By default, the single protocol is IP.

To configure another protocol, use the interface subcommand:

lapb protocol *keyword*

Possible protocol keywords include **ip, xns, decnet, appletalk, vines, clns, novell,** and **apollo**.

Running Multiple Network Protocols

To enable use of multiple network protocols on the same line at the same time, use the keyword **multi-lapb** or **multi-lapb-dce** for DTE or DCE operation, respectively.

encapsulation {**multi-lapb** | **multi-lapb-dce**}

For example, with the **multi-lapb** or **multi-lapb-dce** keyword, you can use IP, DECnet, and XNS at the same time. Both ends of the line must use the same encapsulation: either **lapb** or **multi-lapb**. One end of each line must be DCE.

Sample Configuration of LAPB Encapsulation

In the following example of LAPB encapsulation configuration, the frame size (N1), window size (K), hold timer (TH), and maximum retransmission (N2) parameters retain their default values. The ENCAPSULATION subcommand sets DCE operation for IP packets only, and the LAPB T1 subcommand sets the retransmission timer to 4,000 milliseconds (4 seconds).

Example

```
interface serial 3
encapsulation lapb-dce
lapb t1 4000
```

For more information on LAPB parameters, see the next section, Setting the X.25 Level 2 (LAPB) Parameters.

Setting the X.25 Level 2 (LAPB) Parameters

LAPB parameters are set with the LAPB interface subcommand. The interface must be running with either the LAPB or X.25 encapsulation method specified by the ENCAPSULATION interface subcommand. This subcommand takes two required arguments, *parameter* and *value*. The argument *parameter* is one of several keywords described in the following text, and the argument *value* is a decimal number representing a period of time, a bit count, or a frame count, depending on the parameter. Table 9–1 summarizes the LAPB parameters.

Table 9–1 LAPB Parameters

Parameter	Value	Value Range	Default
k	Frames	1-7	7
n1	Bits	1-16384	12000
n2	Times	1-255	20
t1	Milliseconds	1-64000	3000

Note

The LAPB "th" value is not included as a configurable parameter; the value is always 0.

Setting the Retransmission Timer

The retransmission timer determines how long a transmitted frame can remain unacknowledged before the router polls for an acknowledgment. To set the limit for the retransmission timer (the LAPB T1 parameter), use the following interface subcommand:

lapb t1 *milliseconds*

The argument *milliseconds* is the number of milliseconds from 1 through 64000. The default value is 3,000 milliseconds.

For X.25 networks, the router retransmission timer setting should match that of the network. Mismatched retransmission timers can cause excessive retransmissions and an effective loss of bandwidth.

For leased-line circuits, the retransmission timer setting is critical. The timer setting must be large enough to permit several maximum-sized frames to complete one round trip on the link. If the timer setting is too small, the router will poll before the acknowledgment frame can return, which results in an effective loss of bandwidth. If the timer setting is too large, the router waits longer than necessary before requesting an acknowledgment, which also reduces bandwidth.

To determine an optimal value for the retransmission timer, use the privileged EXEC command PING to measure the round-trip time of a maximum-sized frame on the link. Multiply this time by a safety factor that takes into account the speed of the link, the link quality, and the distance. A typical safety factor is four. Choosing a larger safety factor can result in slower data transfer if the line is noisy. However, this disadvantage is minor compared to the excessive retransmissions and effective bandwidth reduction caused by a timer setting that is too small.

Setting Frame Parameters

To specify the maximum number of bits a frame can hold, use the LAPB N1 interface subcommand:

lapb n1 *bits*

The **n1** keyword specifies the maximum number of bits (N1) a frame can hold. The argument *bits* is the number of bits from 1 through 12,000 and must be a multiple of eight. The default value is 12,000 bits (1500 bytes).

When connecting to an X.25 network, use the N1 parameter value set by the network administration, which is the maximum size of an X.25 packet. When using LAPB over leased lines, the N1 parameter should be eight times the MTU.

To specify the maximum number of times an acknowledgment frame can be retransmitted, use the LAPB N2 interface subcommand:

lapb n2 *retries*

The argument *retries* is the retransmission count from 1 through 255. The default value is 20 retransmissions.

To specify the maximum permissible number of outstanding frames, called the *window size*, use the LAPB K interface subcommand:

lapb k *window-size*

The argument *window-size* is a packet count from one to seven. The default value is seven packets.

Monitoring and Troubleshooting LAPB

To display operation statistics for an interface using LAPB encapsulation, use the EXEC command SHOW INTERFACES.

The following example output shows the state of the LAPB protocol, the current parameter settings, and a count of the different types of frames. Each frame count is displayed in the form sent/received.

```
LAPB state is DISCONNECT, T1 3000, N1 12000, N2 20, K 7,TH 3000
IFRAMEs 12/28 RNRs 0/1 REJs 13/1 SABMs 1/13 FRMRs 3/0 DISCs 0/11
```

For a description of the variable names in the SHOW INTERFACE output, see the X.25 Recommendation.

To debug LAPB problems, you must understand the X.25 recommendation.

To enable the logging of all packets received and generated, use the privileged EXEC command DEBUG LAPB. Note that this command slows down processing considerably on heavily loaded links. The following shows example output:

```
Serial0: LAPB O CONNECT (5) IFRAME 0 1
Serial0: LAPB I CONNECT (2) RR 1 (R)
Serial0: LAPB I CONNECT (5) IFRAME P 2 1 (C)
Serial0: LAPB O REJSENT (2) REJ P/F 1
Serial0: LAPB I REJSENT (2) DM F (C)
Serial0: LAPB I DISCONNECT (2) SABM (C)
Serial0: LAPB O CONNECT (2) UA
.
.
.
Serial0: LAPB T SABMSENT 357964 0
Serial0: LAPB O SABMSENT (2) SABM P
```

In the example output, each line represents a LAPB frame entering or exiting the router. The first field shows the interface type and unit number of the interface

reporting the frame event. The second field is the protocol that provided the information.

The third field is *I*,*O*, or *T* for "frame input," "frame output," or "T1 timer expired," respectively. The fourth field indicates the state of the protocol when the frame event occurred. In a timer event, the state name is followed by the current timer value and the number of retransmissions.

In a packet input or output event, the state name is followed by the size of the frame in bytes (in parentheses) and the frame type name. The next field is an optional indicator: *P/F*, *P*, or *F*, which stand for "Poll/Final," "Poll," and "Final," respectively. For IFRAME frames only, the next two numbers are the receive and send sequence numbers, respectively. For RR, RNR, and REJ frames, the next number is the receive sequence number. For FRMR frames, the next three numbers are three bytes of error data. The last optional indicator is (*C*) or (*R*) for "command" or "response," respectively.

Configuring X.25

The software for the router products supports the 1980 and 1984 Recommendation X.25 published by the French International Telegraph and Telephone Consultative Committee (CCITT). The Recommendation specifies connections between data terminal equipment (DTE) and data communications equipment (DCE). The Defense Data Network (DDN) and the International Standards Organization (ISO) specify the use of X.25 protocol for computer communications. Many public and private networks also use Recommendation X.25 as their interface technology.

Note

The *DECbrouter 90 Configuration and Reference, Volume 3*, Appendix G, "X.25 Diagnostic Codes," describes the differences between the DECbrouter 90 implementation of certain X.25 network-generated, "international problem" diagnostic codes and the definitions provided in Annex E of CCITT Recommendation X.25.

The X.25 model is a telephone network for computer data communications. To start data communications, one computer system calls another to request a communications session. The called computer system can accept or refuse the call. If the called system accepts the call, the two computer systems can begin transferring data in both directions; either system can terminate the call.

In addition to providing remote terminal access, X.25 networks provide bridging capability using a growing list of protocols—the Internet Protocol (IP), DECnet, XNS, ISO CLNS, AppleTalk, Novell IPX, Banyan VINES, and Apollo Domain.

The following sections provide an overview of the DECbrouter 90 X.25 implementation, describing the different encapsulation methods supported by X.25 and X.25 as a datagram transport, with special attention to IP. This is followed by descriptions of the DDN X.25 support provided by the router and descriptions of how to configure X.25 switching and bridging. Finally, the configurations of the X.25 Level 3 parameters and the X.25 Level 2 facilities are discussed. A summary of the interface subcommands are provided at the end of the chapter.

Note

The default values provided by the software are sufficient for most X.25 networks; however, some parameters may need to be configured, depending on the network.

Overview of Digital X.25 Support

DECbrouter 90 X.25 support can be used in two different ways:

- As a transport for datagram traffic—This entails encapsulating datagrams of IP, DECnet, and AppleTalk, inside packets on an X.25 virtual circuit. Mappings between X.25 addresses and protocol addresses allow these datagrams to be routed through an X.25 network. An X.25 Public Data Network (PDN) is used to transport LAN protocols. See Figure 9–1 for a representation of two routers sending data across an X.25 PDN.
- As an X.25 switch—X.25 calls can be routed based on their X.25 addresses either between serial interfaces on the same router (local switching) or across an IP network to another router (remote switching, also called tunneling). See point A in Figure 9–2 for an illustration of a local switching connection. See point B in Figure 9–2 an illustration of a tunneling connection. Remote X.25 switching encapsulates the X.25 packet-level inside a TCP connection, allowing X.25 equipment to be connected via a TCP/IP-based network.

Note

When configuring IP routing over X.25, you may need to make adjustments to accommodate split horizon effects. Refer to *DECbrouter 90 Products, Configuration and Reference, Volume 2*, Chapter 6, "The IP Routing Protocols," for details about how the router handles possible split horizon conflicts. By default, split horizon is enabled for X.25 networks.

Figure 9–1 Transporting LAN Protocols Across an X.25 PDN

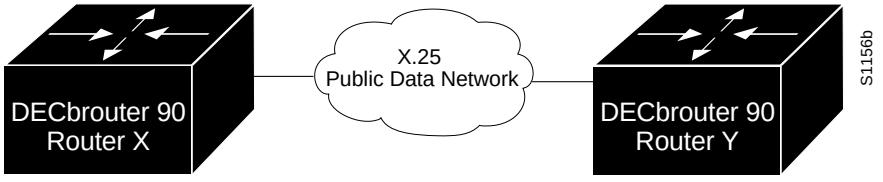
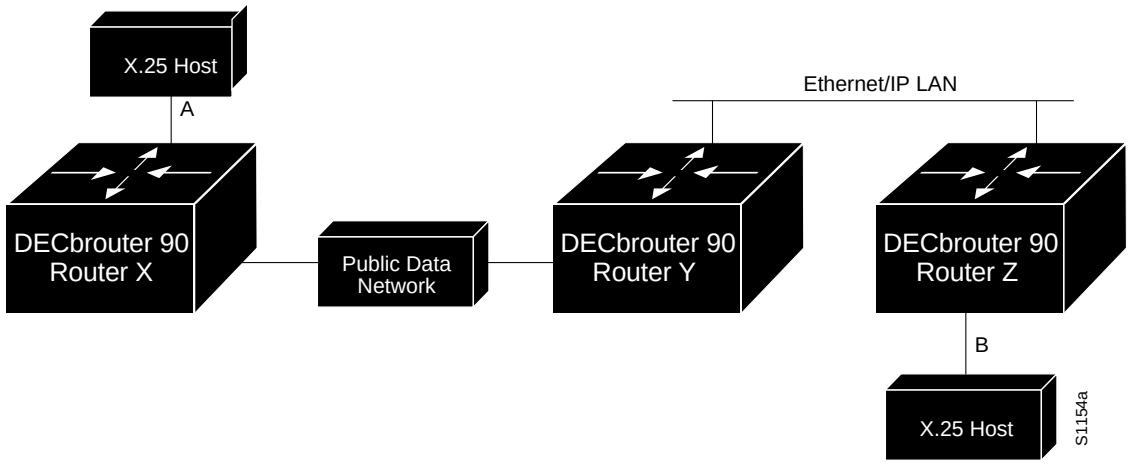


Figure 9–2 Routing X.25 Traffic Through a LAN



Transporting LAN Protocols Across an X.25 PDN

X.25 support is most commonly configured as a transport for datagrams across an X.25 network. This is accomplished by first establishing a mapping between protocol addresses (for example, IP or DECnet) and the X.121 addresses of the X.25 network. When datagrams for a particular destination are routed for the first time, a virtual circuit is set up to the appropriate X.121 address. The Call User Data portion of the initial Call Request identifies the protocol of the datagrams being carried by a particular virtual circuit. If multiple protocols are in use, multiple virtual circuits will be opened.

X.25 Encapsulation Methods

This section describes the different encapsulation methods and commands that are supported for commercial and private X.25 networks.

Encapsulation methods for DDN networks are described in the section Configuring Datagram Transport on DDN Networks in this chapter.

A router using X.25 Level 3 encapsulation can act as a DTE or DCE X.25 interface.

To set X.25 DTE operation, use the ENCAPSULATION X25 interface subcommand:

encapsulation x25

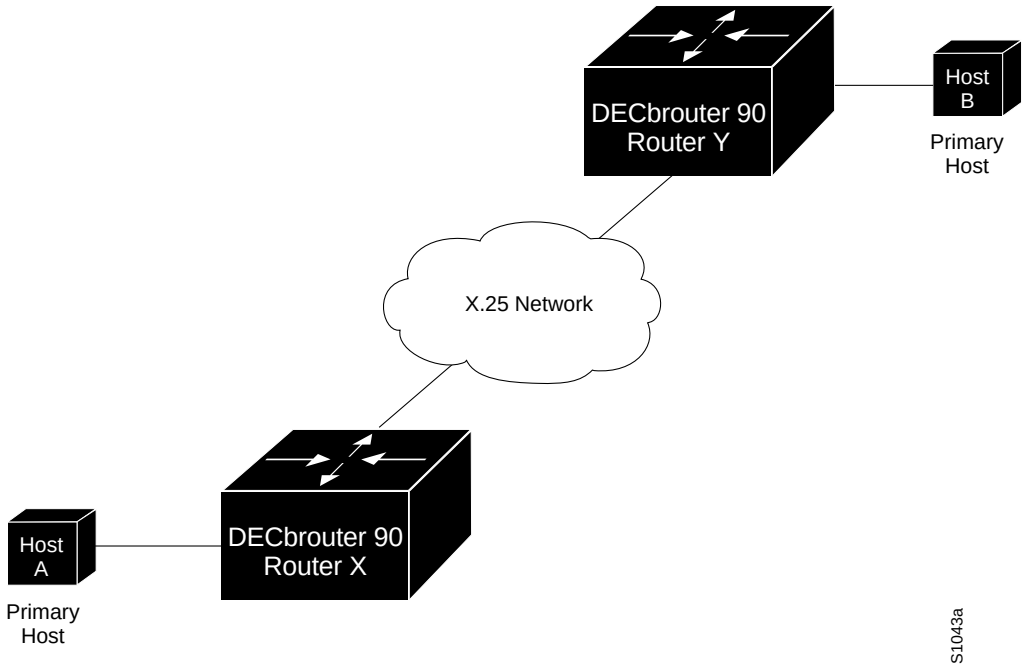
To set X.25 DCE operation, use the ENCAPSULATION X25-DCE interface subcommand:

encapsulation x25-dce

Address Mapping Issues

To transport datagrams using X.25 Level 3, the router must map network protocol addresses to X.121 addresses and vice versa. For example, Figure 9–3 illustrates Hosts A and B that want to communicate via Routers X and Y, which have an X.25 link between them.

Figure 9–3 Communicating via Routers Through an X.25 Network



To send a packet to Host B, Host A must first send the packet to Router X. From the destination address in the packet and from its routing information, Router X determines that it must send the packet to Router Y over the X.25 network. Router X must then determine the X.121 address for Router Y to open a virtual circuit. Router X uses its network protocol-to-X.121 address map to convert the protocol address of Router Y to the equivalent X.121 address. Router X can now make the call to create a virtual circuit.

Except in the case of IP using DDN X.25 encapsulation, there is no standard method for dynamically determining these mappings. Instead, static mapping tables must be configured into each router.

To display the network protocol-to-X.121 address mapping, use the EXEC command SHOW X25 MAP.

Setting Address Mappings

To specify a network-protocol-to-X.121 address mapping such as Internet-to-X.121 or DECnet-to-X.121, use the X25 MAP interface subcommand:

x25 map *protocol-keyword protocol-address X.121-address [options]*
no x25 map *protocol-keyword protocol-address X.121-address*

Note

For bridging over X.25, there is no protocol address; however, the broadcast option is required.

The argument *protocol-keyword* is one of these keywords:

- **ip**—IP
- **decnet**—DECnet
- **chaos**—CHAOSnet
- **xns**—XNS
- **novell**—Novell IPX
- **appletalk**—AppleTalk
- **vines**—VINES
- **apollo**—Apollo Domain
- **pup**—PUP
- **bridge**—Bridging
- **clns**—OSI Connectionless Network Service
- **cmns**—OSI Connection-Mode Network Service
- **compressedtcp**—TCP header compression (This is discussed in more detail in the section X.25 TCP Header Compression in this chapter.)

The *address* arguments specify the network-protocol-to-X.121 mapping.

The *option* arguments add certain features to the mapping specified and can be any of the options that follow.

Note

These options cannot be configured with the **x25 map cmns** version of the X25 MAP command.

- **accept-reverse**—Causes the router to accept incoming reverse-charged calls. If this option is not present, the router clears reverse-charged calls.
- **broadcast**—Causes the router to direct any broadcasts sent through this interface to the specified X.121 address. This option is needed when dynamic routing protocols are being used to access the X.25 network, and is required for bridging X.25.
- **cug** *number*—Specifies a Closed User Group number (from 1 to 99) for the mapping in the outgoing call.
- **modulo** *size*—Specifies the window modulo for this map. The argument *size* can be 8 or 128 to permit a maximum window size of 7 or 127, respectively.
- **nuid** *username password*—Specifies that a network ID facility be sent in the outgoing call with the specified user name and password.

- **nvc** *count*—Sets the number of virtual circuits (VCs) for this protocol/host. The default *count* is the **x25 nvc** setting of the interface. A maximum number of eight VCs can be configured for each single protocol/host pair.
- **packetsize** *in-size out-size*—Specifies input packet size (*in-size*) and output packet size (*out-size*) for the mapping in the outgoing call.
- **reverse**—Specifies reverse charging for outgoing calls.
- **rpoa** *name*—Specifies the list of transit recognized private operating agencies (RPOAs) to use in outgoing call request packets for this x25 map entry.
- **throughput** *in out*—Requests the amount of bandwidth through the X.25 network.
- **transit-delay** *number*—Specifies the transit delay value in milliseconds (0 to 65334) for the mapping in of outgoing calls for networks that support transit delay.
- **windowsize** *in-size out-size*—Specifies input window size (*in-size*) and output window size (*out-size*) for the mapping in the outgoing call.

To retract a network protocol-to-X.121 mapping, use the NO X25 MAP interface subcommand with the appropriate network protocol and X.121 address arguments.

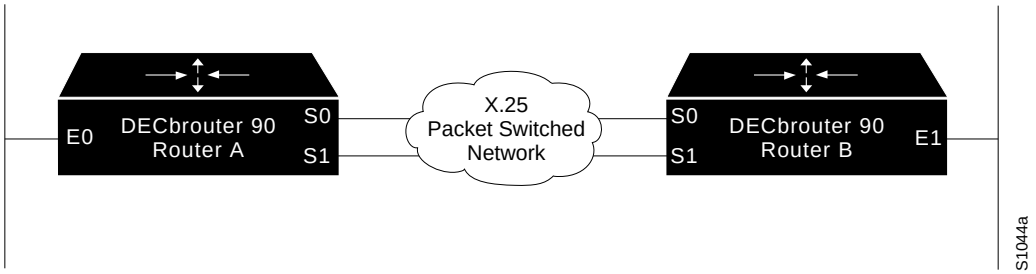
Configuring X.25 to Allow Ping Support over Multiple Lines

For PING commands to work in an X.25 environment (when load sharing over multiple serial lines), you must include entries for all adjacent interface IP addresses in the X25 MAP command for each serial interface. The example configuration that follows illustrates this point.

Example

For example, consider two routers, Router A and Router B, communicating with each other over two serial lines via an X.25 PDN (see Figure 9–4) or over leased lines. In either case, all serial lines must be configured for the same IP subnet address space. In order to allow for successful PING commands, the configuration might be as in the lists that follow. In any event, a similar configuration is required for the same subnet IP addresses to work across X.25.

Figure 9–4 Parallel Serial Lines to X.25 Network



Note

All four serial ports configured for the two routers in the following configuration example must be assigned to the same IP subnet address space. In this case, the subnet is 131.108.170.0.

```
! Configuration for Router A
! -------------------------
int s 0
ip 131.108.170.1 255.255.255.0
x25 address 11
x25 map ip 131.108.170.3 31370054065
x25 map ip 131.108.170.4 31370054065

int s 1
ip 131.108.170.2 255.255.255.0
x25 address 12
x25 map ip 131.108.170.4 31370054067
x25 map ip 131.108.170.3 31370054067

! Configuration for Router B
! -------------------------
int s 0
ip 131.108.170.3 255.255.255.0
x25 address 13
x25 map ip 131.108.170.1 31370054068
x25 map ip 131.108.170.2 31370054068

int s 1
ip 131.108.170.4 255.255.255.0
x25 address 14
x25 map ip 131.108.170.2 31370054069
x25 map ip 131.108.170.1 31370054069
```

Setting Encapsulation Permanent Virtual Circuits

Permanent virtual circuits (PVCs) are the X.25 equivalent of leased lines; they are never disconnected. To establish a PVC, use the X25 PVC interface subcommand:

x25 pvc *circuit protocol-keyword protocol-address [option]*
no x25 pvc *circuit protocol-keyword protocol-address*

The argument *circuit* is a virtual-circuit channel number and must be less than the virtual circuits assigned to the switched virtual circuits (SVCs). The argument protocol-keyword can be one of these keywords:

- **ip**—IP
- **decnet**—DECnet
- **chaos**—CHAOSnet
- **xns**—XNS
- **novell**—Novell IPX
- **appletalk**—AppleTalk
- **vines**—VINES

- **apollo**—Apollo Domain
- **pup**—PUP
- **bridge**—Bridging

The argument *protocol-address* is that of the host at the other end of the PVC.

The optional argument *option* is used to specify the PVC's flow control parameters if they differ from the interface defaults. The option arguments add certain features to the mapping specified and can be any of the options that follow.

- **broadcast**—Causes the router to direct any broadcasts sent through this interface to the specified X.121 address. This option is needed when dynamic routing protocols are being used to access the X.25 network, and is required for bridging X.25.
- **modulo** *size*—Specifies the window modulo for this map. The argument *size* can be 8 or 128 to permit a maximum window size of 7 or 127, respectively.
- **packetsize** *in-size out-size*—Specifies input packet size (*in-size*) and output packet size (*out-size*) for the mapping in the outgoing call.
- **windowsize** *in-size out-size*—Specifies input window size (*in-size*) and output window size (*out-size*) for the mapping in the outgoing call.

PVCs are not supported for ISO CMNS. Switched virtual circuits (SVCs) are sufficient for CMNS connections over X.25.

Note

You must specify the required network protocol-to-X.121 address mapping with an X25 MAP subcommand before you can set up a PVC. See the Setting Address Mappings section earlier in this chapter for a description of that command.

To delete a PVC, use the NO X25 PVC interface subcommand with the appropriate channel number, protocol keyword, and protocol address.

Note

When configuring X.25 to use a PVC, you must ensure that no traffic is sent toward a remote terminal server between the time the X25 MAP command is issued and the time that the X25 PVC command is issued. Otherwise, the local system will create a switched virtual circuit (SVC), and then the X25 PVC command will not be allowed.

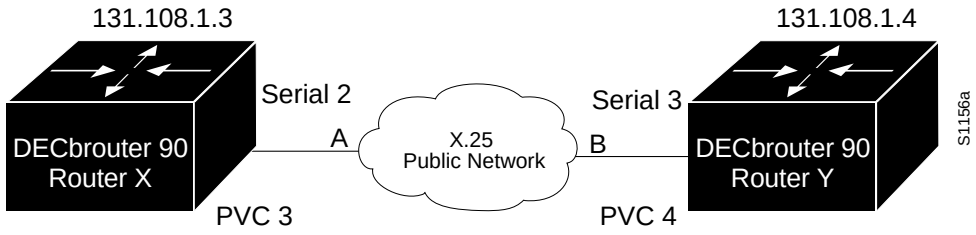
Map entries with the **broadcast** attribute are particularly likely to get traffic, due to routing protocol traffic. The simplest way to ensure that traffic is not sent while configuring an interface to use a PVC is to shut down the interface while configuring it for PVC support.

Example

This example, illustrated in Figure 9–5, demonstrates how to use the PVC to exchange IP traffic between Router X and Router Y. In this example, the PDN has established a PVC through its network connecting PVC number 3 of access point A to PVC number 4 of access point B. On Router X a connection is established between Router X and Router Y's IP address, 131.108.1.4. On Router Y a connection is established between Router Y and Router X's IP address, 131.108.1.3.

```
Router X> x25 map ip 131.108.1.3 0
interface serial 2
x25 pvc 3 ip 131.108.1.3

Router Y> x25 map ip 131.108.1.4 0
interface serial 3
x25 pvc 4 ip 131.108.1.4
```

Figure 9–5 Establishing a PVC through an X.25 Network



Protocol-to-Virtual Circuit Mapping

The call request packet that sets up a virtual circuit contains a field called call user data (CUD). Typically, the software uses the first byte of CUD to distinguish which high-level protocol will be carried by a particular virtual circuit.

Table 9–2 lists the hexadecimal values of the initial byte of CUD and its corresponding network level protocol. The use of 0x81 for ISO 8473 (CLNS) is an ISO standard. The use of 0xCC for Department of Defense IP is defined by RFC 877. The use of C0 00 80 C4 is defined by Banyan. The other values are meaningful only to this software. All the values are padded with three bytes of 0x00, except for VINES, the BFE X.25 encapsulation, and CLNS, all of which follow ISO 8473 requirements.

Table 9–2 Protocols and Initial Byte of Call User Data

Protocol	Initial CUD Byte
ISO CLNS	0x81
DOD IP	0xCC
PUP	0xCE
Chaosnet	0xCF
DECnet	0xD0
XNS	0xD1
AppleTalk	0xD2
Novell	0xD3
Apollo Domain	0xD4
VINES	0xC0 0x00 0x80xC4
Bridges	0xD5
Call User Data (CUD)	0xd8

To set the default protocol, use the X25 DEFAULT command. The full syntax follows:

x25 default *protocol*
no x25 default *protocol*

The X25 DEFAULT command specifies the protocol assumed by the router to interpret calls with unknown CUD. The argument *protocol* sets the default protocol and is either IP or PAD. Use this subcommand to change the action taken when an incoming call is received without identifying CUD. Normally, the call is cleared. When you use this subcommand, the incoming call is assumed to contain the specified default protocol.

The NO X25 DEFAULT subcommand removes the protocol specified.

Displaying Address Mappings

To display the network-protocol-to-X.121 address mappings, enter this command at the EXEC prompt:

show x25 map

The following is a sample output. It is followed by Table 9–3, which describes the fields in the output. The output includes a configuration that uses TCP header compression over X.25.

```
Serial0: novell 10.0.0c00.7b22 000000220200 PERMANENT, 1 LCN: 4*
Serial0: IP 10.1.0.1 000000010100 CONSTRUCTED
Serial1: appletalk 128.1 000000010000 PERMANENT
Serial1: decnet 28.1 000000020000 PERMANENT BROADCAST
Serial1: ip 128.1.0.3 000000030000 PERMANENT, 2 LCN: 1023*, 1024
Serial1: COMPRESSEDTCP 128.1.0.4 000000111111 PERMANENT
  Compression statistics
  Interface Serial1:
    Rcvd:    4060 total, 2891 compressed, 0 errors
             0 dropped, 1 buffer copies, 0 buffer failures
    Sent:    4284 total, 3224 compressed,
             105295 bytes saved, 661973 bytes sent
             1.15 efficiency improvement factor
    Connect: 16 slots, 1543 long searches, 2 misses, 99% hit ratio
             Five minute miss rate 0 misses/sec, 0 max misses/sec

Serial1: IP 128.1.0.4 000000111111 PERMANENT
```

Note

Instead of being interface-specific, statistics for TCP header compression over X.25 are specific to each compressed TCP map entry. This is in contrast to the output for the command SHOW IP TCP HEADER-COMPRESSION, which is described in the *DECbrouter 90 Configuration and Reference, Volume 2*.

Table 9–3 Show X25 Map Field Descriptions

Field	Description
Interface name	Example of interface name.
Protocol type	Protocol type.
Protocol address	First address: Source protocol address. Second address: X.121 address mapping.
Address mapping type	Address-mapping type examples: PERMANENT: entered with x25 map interface subcommand. INTERFACE: indicates address of a router network interface. CONSTRUCTED: derived using the DDN address conversion scheme. BROADCAST: appears on the output line if broadcasts are enabled for an address mapping.
LCN	Logical circuit numbers. If the LCN is greater than zero, a list of the LCNs for the protocol/X.121 address is displayed. An asterisk marks the current LCN.
Rcvd:	
Total	Total number of TCP packets received.
Compressed	Total number of TCP packets compressed.
Errors	Unknown packets.
Dropped	Number of packets dropped due to invalid compression.

(continued on next page)

Table 9–3 (Cont.) Show X25 Map Field Descriptions

Field	Description
Buffer copies	Number of packets that had to be copied into bigger buffers for decompression.
Buffer failures	Number of packets dropped due to a lack of buffers.
Sent:	
Total	Total number of TCP packets sent.
Compressed	Total number of TCP packets compressed.
Bytes saved	Number of bytes reduced.
Bytes sent	Number of bytes sent.
Efficiency improvement factor	Improvement in line efficiency because of TCP header compression.
Connect:	
Number of slots	Size of the cache.
Long searches	The number of times the software had to look to find a match.
Misses	The number of times a match could not be made. If your output shows a large miss rate, then the number of allowable simultaneous compression connections may be too small.
Hit ratio	Percentage of times the software found a match and was able to compress the header.
Five minute miss rate	Calculates the miss rate over the previous five minutes for a longer-term (and more accurate) look at miss rate trends.

Example X.25 Configuration

The following example shows the complete configuration for a serial interface connected to a commercial X.25 PDN for routing the IP. The IP subnetwork address 131.108.9.0 has been assigned for the X.25 network.

Note

When routing IP over X.25, the X.25 network must be treated as a single IP network or subnetwork. Map entries for routers with addresses on subnetworks other than the one on which the interface's IP address is stored are ignored by the routing software. Additionally, all routers using the subnet number should have map entries for all others. There are also issues with the broadcast flag, which apply both to IP and to other protocols with dynamic routing.

```
interface serial 1
ip address 131.108.9.1 255.255.255.0
!
encapsulation X25
!
! The "bandwidth" command is not part of the X.25
! configuration; it's especially important to understand that it doesn't
! have any connection with the X.25 entity of the same name.
!"bandwidth" commands are used by IP routing processes (currently only IGRP),
! to determine which lines are the best choices for traffic.
! Since the default is 1544, and X.25 service at that rate isn't generally
! available, most X.25 interfaces that are being used with IGRP in a
! real environment will have "bandwidth" settings.
!
! This is a 9.6 Kbaud line:
!
bandwidth 10
!
! These Level 3 parameters are defaults; they need to
! match the PDN defaults. They are negotiable on a per-call basis:
!
x25 win 7
x25 wout 7
x25 ips 512
x25 ops 512
!
! You must specify an X.121 address to be assigned to the X.25 interface by
! the PDN.
!
x25 address 31370054065
!
! The following Level 3 parameters have been set to match the network.
! You generally need to change some Level 3 parameters, most often
! those listed below. You may not need to change any Level 2
! parameters, however.
!
x25 htc 32
x25 idle 5
x25 nvc 2
!
! The following commands configure the X.25 map. If you want to exchange
! routing updates with any of the routers, they would need "broadcast" flags.
! If the X.25 network is the only path to them, static routes are
! generally used to save on packet charges. If there is a redundant path,
! it might be desirable to run a dynamic routing protocol.
!
x25 map IP 131.108.9.3 31370019134 ACCEPT-REVERSE
! (ACCEPT-REVERSE allows collect calls)
x25 map IP 131.108.9.1 31370054065
x25 map IP 131.108.9.2 31370053087
!
! If the PDN cannot handle fast back-to-back packets,use the
!"transmitter-delay" command to slow down the interface:
!
transmitter-delay 50
```

Routing X.25 Traffic through a LAN

In addition to transporting datagrams, the X.25 software implementation allows virtual circuits to be forwarded from one X.25 interface to another and from one router to another. The forwarding behavior can be controlled with switching and tunneling commands, based on a locally built table.

Higher-level protocols can share an X.25 encapsulated serial interface with the X.25 switching support. Switching or forwarding X.25 virtual circuits can be done two ways:

- Incoming calls received from a local serial interface running X.25 can be forwarded to another local serial interface running X.25. This is known as *local X.25 switching*, because the router handles the complete path. It does not matter whether the interfaces are configured as DTE or DCE, because software will take the appropriate actions.
- An incoming call also can be forwarded to another router over a LAN using the TCP/IP protocols. Upon receipt of an incoming call, a TCP stream connection is established to the router that is acting as the switch for the destination. All X.25 packets are sent and received over this reliable data stream. Flow control is maintained from local interface to remote interface. This is known as *remote X.25 switching*, or tunneling. It does not matter whether the interfaces are configured as DTE or DCE, because software will take the appropriate actions.

Running X.25 over TCP/IP provides a number of benefits. The IP datagram containing the X.25 packet can be switched by other routers using their high-speed switching abilities. X.25 connections can be sent over networks running only the TCP/IP protocols. The TCP/IP protocol suite runs over many different networking technologies, including Ethernet and T1 serial. Thus X.25 data can be forwarded over these media to another router, where it can be output to an X.25 interface.

When the connection is made locally the switching command is used, and when the connection is across a LAN the tunneling command is used. The basic switching (or tunneling) function is the same for both types of connections, but different software commands are required for the two types of connections.

The X.25 switching subsystem supports the following facilities and parameters:

- The D-bit ignored but passed through transparently
- Variable-length interrupt data
- Flow control parameter negotiation
  - — Window size up to 7
  - — Packet size up to 1024
- The basic closed user group selection
- Throughput class negotiation
- Reverse charging and fast select
- Stripped local facilities

Enabling X.25 Switching

The X25 ROUTING command enables local switching or tunneling (used for remote switching), allowing you to route X.25 traffic through a LAN. To enable X.25 switching or tunneling, use the following global command:

x25 routing
no x25 routing

X.25 calls will not be forwarded until this command is issued. The command NO X25 ROUTING disables the forwarding of X.25 calls.

Constructing the X.25 Routing Table

The X.25 routing table is consulted when an incoming call is received that should be forwarded. The called (destination) X.121 address and CUD fields of the X.25 packet are used to determine the route.

An entry in the X.25 routing table is set up or removed with the x25 route global configuration commands. The full syntax and variations of these commands follows:

x25 route *[# position] x121-pattern* [**cud** *pattern*] **interface** *interface-name*
no x25 route *[# position] x121-pattern* [**cud** *pattern*] **interface** *interface-name*

x25 route *[# position] x121-pattern* [**cud** *pattern*] **ip** *ip-address*
no x25 route *[# position] x121-pattern* [**cud** *pattern*] **ip** *ip-address*

x25 route *[# position] x121-pattern* [**cud** *pattern*] **alias** *interface-name*
no x25 route *[# position] x121-pattern* [**cud** *pattern*] **alias** *interface-name*

The order in which X.25 routing table entries are specified is significant; the list is scanned linearly for the first match. The optional argument *# position* (# followed by an integer) can be used to designate after which existing entry to insert or delete the new entry. If no positional parameter is given, the entry is appended to the end of the routing table.

The argument *x121-pattern* can be either an actual x.121 destination address or a regular expression representing a group of X.121 addresses.

The argument *pattern* is a regular expression that must match the called address and is required. Optional CUD corresponding to that X.121 address can be specified as a printable ASCII string. Both the X.121 address and Call User Data can be written using UNIX-style regular expressions. The Call User Data field specifies the data after the protocol identification field, which is four bytes. Use the SHOW X25 ROUTE command to display the X.25 routing table.

The **alias** keyword permits a way for other X.121 addresses to be treated as local. An alias accepts calls for the router.

Enter the NO X25 ROUTE command with the appropriate arguments and keywords to remove the entry from the table.

Translating X.25 Called Addresses

When interconnecting two separate X.25 networks, it is sometimes necessary to provide for address translation. Your X.25 switch supports translation of X.25 called and calling addresses using the **substitute-dest** or **substitute-source** keyword with the X25 ROUTE configuration subcommand. Addresses can be rewritten using regular expression replacement.

The **substitute-source** keyword allows substitution of the calling address. For backwards compatibility, the **substitute** keyword will be accepted as **substitute-dest** and written to nonvolatile memory in the new format. When used with the X25 USE-SOURCE-ADDRESS command, this option allows the calling address to be modified.

> **x25 route** *[# position] x121-pattern* [**substitute-source** *rewrite-pattern*] [**substitute-dest** *rewrite-pattern*] [**cud** *pattern*] **interface** *destination-interface*

For typographical reasons, this command is shown on two lines. When using the optional keywords in this variation of the X25 ROUTE subcommand, the **substitute-source** keyword must precede the **substitute-dest** keyword, and both must precede the **cud** keyword. The entire command must be on one line.

The argument *x121-pattern* can be either an actual x.121 destination address or a regular expression representing a group of X.121 addresses.

The argument *rewrite-pattern* will replace the called or calling X.121 address in routed X.25 packets, as appropriate. The backslash (\) character is treated specially in the argument *rewrite-pattern*; it indicates that the digit immediately following it selects a portion of the original called address to be inserted in the new called address. The characters \0 are replaced with the entire original address. The characters \1 through \9 are replaced with the strings that matched the first through ninth parenthesized parts of X121-*pattern*. See Table 9–4 and Table 9–5 for a summary of pattern and character matching. A more complete description of the pattern-matching characters can be found in the *DECbrouter 90 Configuration and Reference, Volume 3*, Appendix D.

Table 9–4 Pattern Matching

Pattern	Matching
\0	Replaces entire original address
\1..9	Replaces strings that match first through ninth parenthesized part of X.121 address
*	Matches 0 or more sequences of the regular expressions
+	Matches 1 or more sequences of the regular expressions
?	Matches the regular expression of the null string

Table 9–5 Character Matching

Character	Matching
^	Matches the null string at the beginning of the input string
$	Matches the null string at the end of the input string
\char	Matches *char*
.	Matches any single character

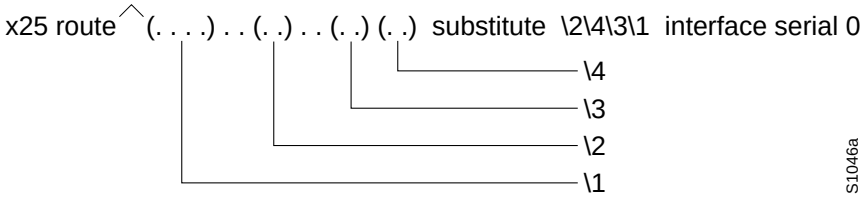
Example

This example indicates that X.25 calls to addresses whose first four data network identification code (DNIC) digits are 1111 should be routed through interface serial 3, but that the DNIC field in the addresses presented to the equipment connected to that interface should be changed to 2222. The \1 characters in the rewrite pattern indicate the portion of the original address matched by the characters. The asterisk should be inserted in the rewritten address.

```
x25 route ^1111(.*) substitute-dest 2222\1 interface serial 3
```

Figure 9–6 shows a more contrived example intended to illustrate the power of the rewriting scheme.

Figure 9–6 X.121 Address Translation Scheme



This sample would cause all X.25 calls with 14-digit called addresses to be routed through interface serial 0. The incoming DNIC field would be moved to the end of the address. The fifth, sixth, ninth, and tenth digits would be deleted, and the thirteenth and fourteenth would be moved before the eleventh and twelfth.

Configuring Switched PVCs

You can configure X.25 permanent virtual circuits (PVCs) in the X.25 switching software. This means that DTEs that require permanent circuits can be connected to the router acting as an X.25 switch and have a properly functioning connection. X.25 RESETs will be sent to indicate when the circuit comes up or goes down. Use the X25 PVC interface subcommand to configure a PVC for a given interface. The syntax of this command follows.

x25 pvc *pvc-number1* **interface** *interface-name pvc pvc-number2 [option]*

The argument *pvc-number1* is the PVC number that will be used on the local interface (as defined by the primary interface command). The argument *pvc-number2* is the number that will be used on the remote interface. The argument **interface-name** is the remote interface type and unit number (serial 0, for example), as specified by the INTERFACE command. The *option* arguments add certain features to the mapping specified and can be any of the options that follow.

- **packetsize** *in-size out-size*—Specifies input packet size (*in-size*) and output packet size (*out-size*) for the mapping in the outgoing call.
- **windowsize** *in-size out-size*—Specifies input window size (*in-size*) and output window size (*out-size*) for the mapping in the outgoing call.

Example

Consider a PVC connected between two serial interfaces on the same router. In this type of interconnection configuration, the alternate interface must be specified along with the PVC number on that interface. To make a working PVC connection, two commands must be specified, each pointing to the other as this example illustrates.

```
interface serial 0
encapsulation x25
x25 ltc 5
x25 pvc 1 interface serial 1 pvc 1
interface serial 1
encapsulation x25
x25 ltc 5
x25 pvc 1 interface serial 0 pvc 1
```

X.25 Switching Configuration Examples

The following two examples illustrate how to enable an X.25 switch, how to enable call forwarding, and how to configure a router on a Tymnet/PAD switch to accept and forward calls.

Examples

This configuration shows how to enable X.25 switching, as well as how to enter routes into the X.25 routing table.

```
!
! Enable X.25 forwarding
x25 routing
!
! Enter routes into the table. Without a positional parameter, entries
! are appended to the end of the table
x25 route ^100$ interface serial 0
x25 route 100 cud ^pad$ interface serial 1
x25 route .* ip 10.2.0.2
!
```

This routing table forwards calls for X.121 address 100 out via interface serial 0. Otherwise, if the X.121 address contains 100 anywhere within it and the CUD is the string pad, it is forwarded onto serial 1. All other X.121 addresses will match the third entry, which is a match-all pattern and will have a TCP connection established to the IP address 10.2.0.2. The router at 10.2.0.2 will then route the call according to its X.25 routing table.

This example configures a router that sits on a Tymnet/PAD switch to accept calls and have them forwarded to a Digital VAX system. This feature permits running X.25 network over a generalized, already existing IP network, thereby making it unnecessary to get another physical line for one protocol.

The router positioned next to the Digital VAX system is configured with X.25 routes, as follows:

```
x25 route vax-x121-address interface serial 0
x25 route.* ip Digital-on-tymnet-ipaddress
```

This routes all calls to the Digital VAX X.121 address out to serial 0, where the VAX system is connected running PSI. All other X.121 addresses are forwarded to the *Digital-on-tymnet* address using its IP address. This takes all outgoing calls from the VAX system and sends them to *Digital-on-tymnet* for further processing.

On the router named *Digital-on-tymnet*, you would enter these commands:

```
x25 route vax-x121-address ip Digital-on-vax
x25 route .* interface serial 0
```

This forces all calls with the VAX X.121 address to be sent to the router with the VAX system connected to it. All other calls with X.121 addresses are forwarded out to Tymnet. If Tymnet can route them, a CALL ACCEPTED packet is returned, and everything proceeds normally. If Tymnet cannot handle it, it clears the call, and the CLEAR REQUEST packet is forwarded back toward the VAX system.

Setting the X.25 Parameters

The DECbrouter 90 software provides subcommands to configure the standard Level 2 and Level 3 X.25 parameters and user facilities. This section discusses the commands and procedures needed to set these parameters, including interface addresses, virtual circuit ranges, and addresses.

Once you establish X.25 Level 3 encapsulation, you can set X.25 Level 3 parameters. These parameters are described in the following sections.

Note

If you connect a router to an X.25 network, use the parameters set by the network administrator. Also, note that the X.25 Level 2 parameters described in Setting the X.25 Level 2 (LAPB) Parameters earlier in this chapter can be configured on an X.25 interface and affect X.25 Level 3 operations.

Setting the X.25 Interface Address

To set the X.121 address of a particular network interface, use the X25 ADDRESS subcommand. The address is assigned by the X.25 network.

x25 address *X.121-address*

The argument *X.121-address* is a variable-length X.121 address.

Example

The following subcommand sets the X.121 address of the current interface to address 2:

```
x25 address 2
```

The value must match that assigned by the X.25 network.

The section DDN X.25 Dynamic Mapping in this chapter describes the addressing scheme for DDN X.25 networks.

Configuring Virtual Circuit Ranges

The X.25 protocol maintains multiple connections over one physical link between a DTE and a DCE. These connections are called virtual circuits (VCs) or logical channels (LCs). X.25 can maintain up to 4095 VCs numbered 1 through 4095; an individual VC is identified by giving its logical channel identifier (LCI) or virtual circuit number (VCN). Many networking documents use VC and LC, and VCN, LCN, and LCI interchangeably; they all mean the same thing.

An important part of X.25 operation is virtual circuit number ranges. Virtual circuit numbers are broken into four ranges (listed here in numerically increasing order):

1. Permanent virtual circuits (PVCs)
2. Incoming-only circuits
3. Two-way circuits
4. Outgoing-only circuits

The incoming-only, two-way, and outgoing-only ranges define the VC numbers over which switched virtual circuits (SVCs) can be established by placing X.25 calls, much like a telephone network establishes a switched voice circuit when a call is placed.

Rules about DCE and DTE devices initiating calls are as follows:

- Only the DCE device can initiate a call in the incoming-only range.
- Only the DTE device can initiate a call in the outgoing-only range.
- Both the DCE device and the DTE device can initiate a call over a virtual circuit number in two-way range.

There is no difference in the operation of the SVCs other than restrictions on which devices can initiate calls. These ranges can be used to prevent one side from monopolizing the virtual circuits and are useful for X.25 interfaces that have only a small total number of SVCs available.

Six X.25 parameters define the upper and lower limits of each of the three SVC ranges. An SVC range cannot overlap another range. A PVC must be assigned a number that is lower than the numbers assigned to the SVC ranges. All ranges are assigned by the network administrator and must fall between 1 and 4095.

To set the SVC range upper- and lower-limit parameters, use the X25 subcommand keywords listed in Table 9–6. Each keyword takes a circuit number as its argument. Note that the values for these parameters must be the same on both ends of an X.25 link; for connection to a public data network (PDN), these values must be set to the values assigned by the network. An SVC range is unused if its upper and lower limits are set to 0.

Note

Because the X.25 protocol requires the DTE and DCE to have identical VC ranges, the DECbrouter 90 implementation will not allow an interface's VC ranges to be modified while the X.25 line protocol is up.

Table 9–6 Range Limit Keywords for the Switched Virtual Circuit Ranges

Keyword	Limit Type	Range	Default
lic	Lower limit, incoming-only circuits	1-4095	0
hic	Upper limit, incoming-only circuits	1-4095	0
ltc	Lower limit, two-way circuits	1-4095	1
htc	Upper limit, two-way circuits	1-4095	1024
loc	Lower limit, outgoing-only circuits	1-4095	0
hoc	Upper limit, outgoing-only circuits	1-4095	0

Setting the Lowest Incoming-only Circuit Number

The **lic** keyword sets the lowest incoming-only virtual circuit number.

x25 lic *circuit-number*

The argument *circuit-number* is a virtual circuit number from 1 through 4095, or 0 if there is no incoming-only VC range. The default value is 0.

Setting the Highest Incoming-only Circuit Number

The **hic** keyword sets the highest incoming-only virtual circuit number.

x25 hic *circuit-number*

The argument *circuit-number* is a virtual circuit number from 1 through 4095, or 0 if there is no incoming-only virtual circuit range. The default value is 0.

Setting the Lowest Two-way Circuit Number

The **ltc** keyword sets the lowest two-way virtual circuit number.

x25 ltc *circuit-number*

The argument *circuit-number* is a virtual circuit number from 1 through 4095, or 0 if there is no two-way virtual circuit range. The default value is 1.

Setting the Highest Two-way Circuit Number

The **htc** keyword sets the highest two-way virtual circuit number.

x25 htc *circuit-number*

The argument *circuit-number* is a virtual circuit number from 1 through 4095, or 0 if there is no two-way virtual circuit range. The default value is 1024 for X.25 network service interfaces; 4095 for connection-mode network service (CMNS) interfaces.

Setting the Lowest Outgoing-only Circuit Number

The **loc** keyword sets the lowest outgoing-only virtual circuit number.

x25 loc *circuit-number*

The argument *circuit-number* is a virtual circuit number from 1 through 4095, or 0 if there is no outgoing-only virtual circuit range. The default value is 0.

Setting the Highest Outgoing-only Circuit Number

The **hoc** keyword sets the highest outgoing-only virtual circuit number.

x25 hoc *circuit-number*

The argument *circuit-number* is a virtual circuit number from 1 through 4095, or 0 if there is no outgoing-only virtual range. The default value is 0.

Example

The configuration in this example sets the following virtual circuit ranges: 5-20 dedicated to incoming calls only (from the DCE to the DTE), 25-1024 for either incoming or outgoing calls, no virtual circuit range dedicated to outgoing calls (from the DTE to the DCE). Up to four permanent virtual circuits can be defined on the virtual circuit numbers 1 through 4.

```
x25 lic 5
x25 hic 20
x25 ltc 25
```

Maintaining Virtual Circuits

The router can clear a switched virtual circuit after a set period of inactivity. To set this period, use the X25 IDLE interface subcommands:

x25 idle *minutes*
no x25 idle

The argument *minutes* is the number of minutes in the period. The default value is zero, which causes the router to keep the SVC open indefinitely. Calls both originated and received by the router are cleared. The NO X25 IDLE command returns to the default.

To increase datagram throughput across an X.25 network, you can establish up to eight SVCs to a host. To specify the maximum number of SVCs that can be open simultaneously to one host, use the X25 NVC interface subcommand:

x25 nvc *count*

The argument *count* is a circuit count from 1 to 8; the default is 1. A maximum of eight VCs can be configured for each protocol/host pair. This value should not be set greater than 1 for use with protocols that do not tolerate out-of-order delivery such as switched X.25 and encapsulated TCP header compression.

When the windows and output queues of all existing connections to a host are full, a new virtual circuit will be opened to the designated circuit count. If a new connection cannot be opened the data is dropped.

Note

The *count* value specified for **x25 nvc** affects the default value for the number of SVCs. It does not affect the NVC value for any X25 MAP commands that already have been configured.

Configuring the Ignore VC Timer

Upon receiving a Clear Request for an outstanding Call Request, the X.25 support code immediately tries another Call Request if it has more traffic to send. This can overrun some X.25 switches. To prevent this problem, use the X25 HOLD-VC-TIMER configuration commands:

x25 hold-vc-timer *minutes*
no x25 hold-vc-timer

The argument *minutes* is the number of minutes to prevent calls from trying to reach a previously failed destination. Incoming calls still will be accepted. The default value is 0; the NO X25 HOLD-VC-TIMER command restores this default.

Configuring the X.25 Level 3 Retransmission Timers

The X.25 Level 3 retransmission timers determine how long the router must wait before retransmitting various Request packets. You can set these timers independently using the X25 subcommand keywords listed in Table 9–7. Each keyword requires a time value in seconds as its argument. The last column shows the default timer values in seconds. Four of the timers apply to DTE devices and the other four to DCE devices.

Table 9–7 Retransmission Timer Keywords and Defaults

Keyword (DTE/DCE)	Affected Request Packet	Time (seconds) (DTE/DCE)	Time (seconds) (DXE)
t20/t10	Restart Request	180/60	60
t21/t11	Call Request	200/180	180
t22/t12	Reset Request	180/60	60
t23/t13	Clear Request	180/60	180

Note

When setting this timer for a CMNS configuration, you are setting it for a DXE interface, which can be either DTE or DCE.

Setting the DTE Restart Request Retransmission Timer

The **t20** keyword sets the limit for the Restart Request retransmission timer (T20) on DTE devices.

x25 t20 *seconds*

The argument *seconds* is a time value in seconds. The default is 180 seconds.

Setting the DCE Restart Request Retransmission Timer

The **t10** keyword sets the limit for the Restart Request retransmission timer (T10) on DCE devices.

x25 t10 *seconds*

The argument *seconds* is a time value in seconds. The default value is 60 seconds.

Setting the DTE Call Request Limit Timer

The **t21** keyword sets the limit for the Call Request completion timer (T21) on DTE devices.

x25 t21 *seconds*

The argument *seconds* is a time value in seconds. The default value is 200 seconds.

Setting the DCE Call Request Limit Timer

The **t11** keyword sets the limit for the Call Request completion timer (T11) on DCE devices.

x25 t11 *seconds*

The argument *seconds* is a time value in seconds. The default value is 180 seconds.

Setting the DTE Reset Request Retransmission Timer

The **t22** keyword sets the limit for the Reset Request retransmission timer (T22) on DTE devices.

x25 t22 *seconds*

The argument *seconds* is a time value in seconds. The default value is 180 seconds.

Setting the DCE Reset Request Retransmission Timer

The **t12** keyword sets the limit for the Reset Request retransmission timer (T12) on DCE devices.

x25 t12 *seconds*

The argument *seconds* is a time value in seconds. The default value is 60 seconds.

Setting the DTE Clear Request Retransmission Timer

The **t23** keyword sets the limit for the Clear Request retransmission timer (T23) on DTE devices.

x25 t23 *seconds*

The argument *seconds* is a time value in seconds. The default value is 180 seconds.

Setting the DCE Clear Request Retransmission Timer

The **t13** keyword sets the limit for the Clear Request retransmission timer (T13) on DCE devices.

x25 t13 *seconds*

The argument *seconds* is a time value in seconds. The default value is 60 seconds.

Updating the X.121 Address

Some X.25 calls, when forwarded by the X.25 switching support, need the calling (source) X.121 address updated to that of the outgoing interface. This is necessary when forwarding calls from private data networks to public data networks. Outgoing calls forwarded over a specific interface can have their calling X.121 addresses updated by using the X25 USE-SOURCE-ADDRESS subcommand. The full syntax is as follows:

x25 use-source-address
no x25 use-source-address

The NO X25 USE-SOURCE-ADDRESS command prevents updating the source addresses of outgoing calls.

Setting X.25 Packet Sizes

X.25 networks use maximum input and output packet sizes set by the network administration. You can set the router input and output packet sizes to match those of the network with the x25 subcommand keywords **ips** and **ops**, respectively:

x25 ips *bytes*
x25 ops *bytes*

The argument *bytes* is a byte count in the range of 16 through 1024. The default value is 128 bytes.

Larger packet sizes are better than smaller ones, because smaller packets require more overhead processing.

Note

Set the **x25 ips** and **x25 ops** keywords to the same value unless your network supports asymmetry between input and output packets.

To send a packet larger than the X.25 packet size over an X.25 virtual circuit, a router must break the packet into two or more X.25 packets with the M-bit ("more data" bit) set. The receiving device collects all packets with the M-bit set and reassembles them.

Note

For optimal throughput, all X.25 interfaces that may carry routed traffic through an IP network should be configured with the same default packet sizes and window sizes.

Setting the Flow Control Modulus

X.25 supports flow control with a sliding window sequence count. The window counter restarts at zero upon reaching the upper limit, which is called the window modulus. To set the window modulus, use the X25 MODULO interface subcommand:

x25 modulo *modulus*

The argument *modulus* is either 8 or 128. The default value is 8. The value of the modulo parameter must agree with that of the device on the other end of the X.25 link.

Configuring Packet Acknowledgment

To specify upper limits on the number of outstanding unacknowledged packets, use the commands X25 WIN (for input window) and x25 **wout** (for output window).

x25 win *packets*
x25 wout *packets*

The argument *packets* is a packet count. The packet count for **win** and **wout** must be between one and modulus minus one. The default value is two packets.

Note

For optimal throughput, all X.25 interfaces that may carry routed traffic through an IP network should be configured with the same default packet sizes and window sizes.

The X25 WIN command determines how many packets the router can receive before sending an X.25 acknowledgment. The **wout** limit determines how many sent packets can remain unacknowledged before the router uses a hold queue. To maintain high bandwidth use, assign these limits the largest number that the network allows.

Note

Set **win** and **wout** to the same value unless your network supports asymmetry between input and output window sizes.

You can instruct the router to send acknowledgment packets when it is not busy sending other packets, even if the number of input packets has not reached the **win** count. This approach improves line responsiveness at the expense of bandwidth. To enable this option, use the X25 TH subcommand:

x25 th *delay*

The argument *delay* must be between zero and the input window size. A value of one sends one Receiver Ready acknowledgment per packet at all times. The default value is zero, which disables the delayed acknowledgment strategy. A delay value smaller than the window value may be advisable to smooth the data flow of large messages transmitted on serial leased lines to X.25 networks.

The router sends acknowledgment packets when the number of input packets reaches the count you specify, providing there are no other packets to send. For example, if you specify a count of one, the router can send an acknowledgment per input packet.

Suppressing the Calling Address

To omit the calling address in outgoing calls, use the X25 SUPRESS-CALLING-ADDRESS interface subcommand:

x25 suppress-calling-address
no x25 suppress-calling-address

The **suppress-calling-address** keyword omits the calling (source) X.121 address in Call Request packets. This option is required for networks that expect only subaddresses in the calling address field. The calling address is sent by default.

Use the NO X25 SUPPRESS-CALLING-ADDRESS subcommand to reset this subcommand to the default state.

Suppressing the Called Address

To omit the called address in outgoing calls, use the X25 SUPPRESS-CALLED-ADDRESS interface subcommand:

x25 suppress-called-address
no x25 suppress-called-address

The **suppress-called-address** keyword omits the called (destination) X.121 address in Call Request packets. This option is required for networks that expect only subaddresses in the called address field. The called address is sent by default.

Use the NO X25 SUPPRESS-CALLED-ADDRESS subcommand to reset this subcommand to the default state.

Defining a Packet Hold Queue

To define the number of packets to be held until a virtual circuit is established, use the X25 HOLD-QUEUE interface subcommand:

x25 hold-queue *queue-size*
no x25 hold-queue *[queue-size]*

The argument *queue-size* defines the number of packets. By default, this number is zero. Use the NO X25 HOLD-QUEUE command without an argument to remove this command from the configuration file; enter the command with a queue size value of zero to return the default.

Accepting Reverse-Charged Calls

To instruct the router to accept all reverse-charged calls, use the X25 ACCEPT-REVERSE interface subcommand:

x25 accept-reverse
no x25 accept-reverse

The **accept-reverse** keyword causes the interface to accept reverse-charged calls by default. This behavior also can be configured on a per-peer basis using the X25 MAP subcommand. The NO X25 ACCEPT-REVERSE command disables this facility.

Forcing Packet-Level Restarts

To force a packet-level restart when the link level is restarted, use the X25 LINKRESTART interface subcommand:

x25 linkrestart
no x25 linkrestart

This command restarts X.25 Level 2 (LAPB) when errors occur. This behavior is the default and is necessary for networks that expect this behavior. Use the **no x25 linkrestart** to turn this feature off.

Use of the NO X25 LINKRESTART command is strongly discouraged except when connecting to networks that require packet-level restart to be turned off. LAPB frames can be duplicated or lost when LAPB errors occur, which can cause problems for the X.25 packet layer. These problems can be avoided if the packet layer is notified of the LAPB error, causing a restart of the X.25 service.

Setting the Packet Network Carrier

To set the packet network carrier, use the X25 RPOA interface subcommand:

x25 rpoa *name number*
no x25 rpoa *name*

The X25 RPOA interface subcommand specifies a list of transit RPOAs to use, referenced by name. The argument *name* must be unique with respect to all other RPOA names. It is used in the X25 FACILITY and X25 MAP interface subcommands. The argument *number* is a number that is used to describe an RPOA. The NO X25 RPOA command removes the specified name.

Setting X.25 Parameters on a Per-Call Basis

To override interface settings on a per-call basis, use the X25 FACILITY interface subcommand. The full syntax of the command follows.

x25 facility *keyword argument*
no x25 facility *keyword argument*

The command enables X.25 facilities, which are sent between DTE and DCE devices to negotiate certain link parameters.

The argument keyword specifies one of the following keywords, followed by the required *argument*:

- **cug** *number*—Specifies a closed user group number from 1 through 99 to provide an extra measure of network security.
- **packetsize** *in-size out-size*—Sets the size in bytes of input packets (*in-size*) and output packets (*out-size*). Both values should be the same.
- **reverse**—Reverses charges on all Call Request packets from the interface.
- **windowsize** *in-size out-size*—Sets the packet count for input windows (*in-size*) and output windows (*out-size*). Both values should be the same.
- **throughput** *in out*—-Sets the requested throughput values for input and output throughput across the network.

- **rpoa** *name0*—Specifies the list of transit recognized private operating agencies (RPOAs) to use in outgoing Call Request packets.
- **transit-delay** *number*—Specifies the transit delay value in milliseconds (0 to 65334) for the mapping in of outgoing calls, for networks that support transit delay.

The NO X25 FACILITY command with the appropriate keyword and argument removes the facility.

X.25 TCP Header Compression

TCP header compression is supported over X.25 links through the use of the X25 MAP COMPRESSEDTCP interface subcommand:

x25 map compressedtcp

The implementation of compressed TCP over X.25 uses a virtual circuit (VC) to pass the compressed packets. The noncompressed packets use another VC.

The interface subcommand X25 MAP COMPRESSEDTCP is required to make the X.25 calls complete for compressed packets. The command syntax is:

x25 map compressedtcp *ip-address x.121-address [options]*
no x25 map compressedtcp *ip-address x.121-address [options]*

The argument *ip-address* is the IP address, and *x.121-address* is the X.121 address. The *options* argument accepts the same options as those for the X25 MAP command described in the preceding section, but includes the following additional option:

- **passive**—This keyword applies to compressed TCP mappings. If this option is specified, outgoing packets are compressed only if incoming TCP packets on the VC for this map are compressed. When the passive option is not set, all compressible traffic intended for this address map is compressed.

Note

You cannot configure the X25 MAP command's NVC option with a value that is more than one compressed TCP map. If you attempt to do so, the router generates an error message indicating that it cannot be done.

The CUD of compressed TCP calls is the single byte 0xd8.

Note

Typically, the DECbrouter 90 uses the first byte of Call User Data to distinguish which high-level protocol will be carried by a particular virtual circuit.

The NO X25 MAP COMPRESSEDTCP disables TCP header compression for the link.

Bridging on X.25

The DECbrouter 90 transparent bridging software supports bridging of X.25 frames. To configure this capability, add this variation of the X25 MAP interface subcommand to the bridging configuration file:

x25 map bridge *X.121-address* **broadcast** [*options-keywords*]

The command specifies Internet-to-X.121 address mapping. The keyword **bridge** specifies bridging over X.25. The argument *X.121-address* is the X.121 address. The keyword **broadcast** is required for bridging X.25 frames. The argument *options-keywords* represents services that can be added to this map. See the section Setting Address Mappings earlier in this chapter. For further information about bridging on X.25 and for an example configuration, refer to the *DECbrouter 90 Configuration and Reference, Volume 3,* Chapter 7, "Configuring Transparent Bridging."

Configuring Datagram Transport on DDN Networks

The Defense Data Network (DDN) X.25 protocol has two versions: basic service and standard service. Using DDN X.25 basic service, network devices send raw X.25 data across the DDN and assume no structure within the data portion of the X.25 packet. Basic service users can interoperate only with other basic service users.

DDN X.25 standard service requires that the X.25 data packets carry IP datagrams. Because the DDN packet-switch nodes (PSNs) can extract an Internet packet from within an X.25 packet, they can pass data to either an 1822-speaking host or to another standard service host. Thus, hosts using the older 1822 network interface can interoperate with hosts using standard service.

The DDN X.25 Standard is the required protocol for use with DDN PSNs. The Defense Communications Agency (DCA) has certified DECbrouter 90's DDN X.25 Standard implementation for attachment to the DDN.

Enabling DDN X.25

A router using DDN X.25 basic service can act as either a DTE or a DCE device. To set operation type, use the encapsulation interface subcommand:

encapsulation {**ddnx25** | **ddnx25-dce**}

Both of these keywords cause the router to specify the standard service facility in the Call Request packet, which notifies the PSNs to use standard service.

Using standard service, the DDN can provide efficient service for virtual circuits with high-precedence values. If the router receives an Internet packet with a nonzero Internet precedence field, it opens a new virtual circuit and sets the precedence facility request to the DDN-specified precedence mapping in the Call Request packet. Different virtual circuits are maintained based on the precedence mapping values and the number of virtual circuits permitted.

Blacker Emergency Mode

For environments that require a high level of security, your router software supports the Defense Data Network Blacker Front-End Encryption (BFE) and Blacker Emergency Mode.

Blacker Emergency Mode allows your BFE device and your router to function in emergency situations. When the router is configured to participate in emergency mode and the BFE device is in emergency mode, the router sends address translation information to the BFE device to assist the it in sending information.

The DECbrouter 90's implementation of Blacker Emergency Mode adheres to the specifications outlined in the DCA Blacker Interface Control document, published March 21, 1989.

BFE Device Configuration

Your BFE device is configured to be in one of three possible modes as follows:

- Enters emergency mode when requested to by the network. If the router is configured to respond to a BFE device in emergency mode, or if the EXEC command BFE ENTER is used, the router sends address translation information to the BFE device.
- Never enters emergency mode.
- Notifies the router that emergency mode window is open and waits for the router to tell it to enter emergency mode. If the router is configured to respond to a BFE in emergency mode, or if the EXEC command bfe enter is used, the router sends a special address translation packet to the BFE device. The "special" data includes a command to the BFE to enter emergency mode.

Router Configuration

If the router is attached to a BFE device, the **bfex25** keyword must be used with the ENCAPSULATION subcommand. Use this command to configure BFE encapsulation:

encapsulation bfex25

This encapsulation provides a mapping from Class A IP addresses to the type of X.121 addresses expected by the BFE encryption device.

The table resulting from the following x25 REMOTE-RED command provides the address translation information the router sends to the BFE when the BFE is in emergency mode. Use the following command to set up the table that lists the BFE nodes (host or gateways) to which the router will send packets:

x25 remote-red *host-ip-address* **remote-black** *Blacker-Internet-address*

The *host-ip-address* argument is the IP address of the host or a router that the packets are being sent to. The *Blacker-Internet-address* argument is the IP address of the remote BFE device in front of the host that the packet is being sent to.

Use the following command to configure the circumstances under which the router will participate in emergency mode:

x25 bfe-emergency {**never** | **always** | **decision**}

The default keyword is **never**. When set to never, the router does not send address translation information to the BFE. When it does not receive address translation information, the BFE cannot open a new connection for which it does not know the address.

When the **always** keyword is used and the router has been configured to create an address translation table, the router passes address translations to the BFE when the BFE enters emergency mode.

When the **decision** keyword is used, the router waits until it receives a diagnostic packet from the BFE device indicating that the emergency mode window is open. (The window is only open when a condition exists that allows the BFE is to enter emergency mode.) When the diagnostic packet is received, the router's participation in emergency mode depends on how it is configured using the X25 BFE-DECISION command described next in this section.

Use the following command to configure how a router configured as X25 BFE-EMERGENCY DECISION will participate in emergency mode:

x25 bfe-decision {**no** | **yes** | **ask**}

The default value is **no**. When set to no, the router will not participate in emergency mode and will not send address translation information to the BFE.

When set to **yes**, the router will participate in emergency mode, sending address translation information to the BFE when the BFE enters emergency mode. The router gets this information from the table setup by using the x25 remote-red command described earlier in this section.

When the **ask** keyword is used, the router will prompt the administrator to use the EXEC BFE {ENTER | LEAVE} command to send or not send address translation information to the BFE device.

Use the following interface EXEC command to set the router to participate in emergency mode or to end participation in emergency mode if your system is configured for x25 bfe-emergency decision and x25 bfe-decision ask.

bfe {**enter** | **leave**} *interface-type unit*

The keyword options are **enter** and **leave**. The keyword **enter** will cause the router to send a special address translation packet that includes an "enter emergency mode" command to the BFE if the emergency mode window is open. If the BFE is already in emergency mode, this command enables the sending of address translation information.

If the BFE is in emergency mode, the **leave** keyword disables the sending of address translation information from the router to the BFE device.

The *interface-type* argument is the interface name, and unit is the interface number.

Example

In the example that follows, interface Serial 0 is configured to require an EXEC command from the administrator before it participates in emergency mode. The host IP address is 21.0.0.12, and the address of the remote BFE unit is 21.0.0.1. When the BFE enters emergency mode, the router will prompt the administrator for EXEC command BFE ENTER to direct the router to participate in emergency mode.

```
interface serial 0
x25 bfe-emergency decision
x25 remote-red 21.0.0.12 remote-black 21.0.0.1
x25 bfe-decision ask
```

Monitoring Blacker Emergency Mode

Use the following command to display the one-to-one mapping of the host IP addresses and the remote BFE device's IP addresses:

show x25 remote-red

When all of the X.25 bfe nodes (host or gateways) are entered by using the X25 REMOTE-RED command, the SHOW command that follows can be used to display the table.

Example

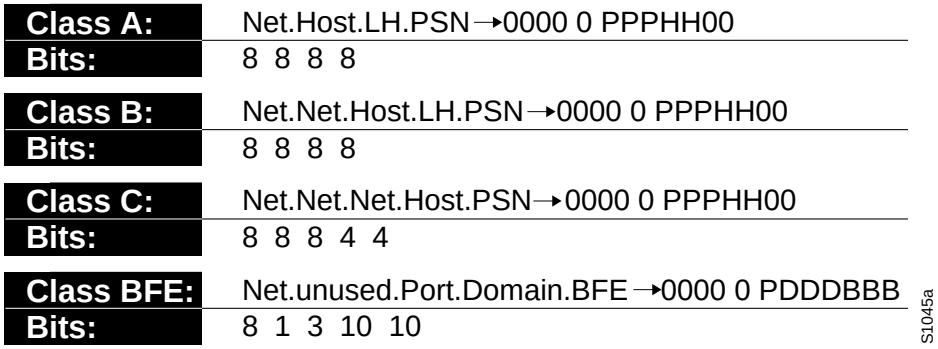
The following show display provides the IP addresses of the host unit (remote-red) and remote BFE units (bfe-remote) for each BFE entry:

```
router> show x25 remote-red
 Entry Remote-red bfe-remote Interface
 1 21.0.0.3 21.0.0.7 serial3
 2 21.0.0.10 21.0.0.6 serial1
 3 21.0.0.24 21.0.0.8 serial3
```

DDN X.25 Dynamic Mapping

The DDN X.25 standard implementation includes a scheme for dynamically mapping all classes of Internet addresses to X.121 addresses without a table. This scheme requires that the Internet addresses conform to the formats shown in Figure 9–7. These formats segment the Internet addresses into network (N), host (H), logical address (L), and IMP (I) portions. (The acronym IMP, which stands for interface message processor, is the predecessor of PSN, which stands for packet switch node.) For the BFE encapsulation, the Internet address is segmented into Port (P), Domain (D), and BFE ID number (B).

Figure 9–7 DDN Internet/X.121 Address Conventions



The DDN conversion scheme uses the host and IMP portions of an Internet address to create the corresponding X.121 address. Strictly speaking, the DDN conversion mechanism is limited to Class A Internet addresses. However, the router can convert Class B and Class C addresses as well. As indicated in Table 8-8, this method uses the last two octets of a Class B address as the host and IMP identifiers, and the upper and lower four bits in the last octet of a Class C address as the host and IMP identifiers, respectively.

The DDN conversion scheme uses a physical address mapping if the host identifier is numerically less than 64. (This limit derives from the fact that a PSN cannot support more than 64 nodes.) If the host identifier is numerically larger than 64, the resulting X.121 address is called a logical address. The DDN does not use logical addresses.

The format of physical DDN X.25/X.121 addresses is ZZZZFIIIHHZZ(SS), where each character represents a digit. ZZZZ represents four zeros, F is zero to indicate a physical address, III represents the IMP (PSN) octet from the Internet address padded with leading zeros, HH is the host octet from the Internet address padded with leading zeros, and ZZ represents two zeros. (SS) represents the optional and unused subaddress.

The physical and logical mappings of the DDN conversion scheme always generate a 12-digit X.121 address. Subaddresses are optional; when added to this scheme, the result is a 14-digit X.121 address. The DDN does not use subaddressing.

Packets using routing and other protocols that require broadcast support can successfully traverse X.25 networks, including the DDN. This traversal requires the use of network protocol-to-X.121 maps because the router must know explicitly where to deliver broadcast datagrams. (X.25 does not support broadcasts.) You can mark network protocol-to-X.121 map entries to accept broadcast packets; the router then sends broadcast packets to hosts with marked entries. If you do not specify the address for an interface configured for DDN X.25, the router uses the DDN mapping technique to obtain the X.121 address of an interface.

To display the network protocol-to-X.121 address mapping, enter this command at the EXEC prompt:

show x25 map

DDN X.25 Configuration Subcommands

Normally the X.25 parameters of a DDN connection are configured using the x25 and lapb interface subcommands described earlier in this chapter. There are a few DDN-specific subcommands, however.

This X.25 implementation allows you to enable or disable the ability to open a new virtual circuit based on the IP type of service (TOS) field.

To do this, use the X25 IP-PRECEDENCE interface subcommand. The full syntax of this command follows.

x25 ip-precedence
no x25 ip-precedence

By default, the router opens one virtual circuit for each TOS> There is a problem associated with this in that some hosts send nonstandard data in the TOS field, thus causing multiple, wasteful virtual circuits to be created. The command no x25 ip-precedence causes the TOS field to be ignored when opening virtual circuits.

Maintaining X.25

To clear all virtual circuits at once, use the privileged EXEC command CLEAR X25-VC. This command takes an interface type keyword and a unit number as arguments to identify the interface with which the virtual circuits are associated.

To clear a particular virtual circuit, add the two arguments described above the CLEAR X25-VC command, and include a logical circuit number (LCN) value as a third argument. The command syntax is as follows:

clear x25-vc *interface unit [lcn]*

The CLEAR X25-VC command clears all X.25 virtual circuits at once. The argument *interface* is the interface type. The argument *unit* is the interface unit number. The optional argument *lcn* clears the specified virtual circuit.

Monitoring X.25 Level 3 Operations

The router provides EXEC SHOW commands to provide information on interface and virtual circuit operation. Use the EXEC command SHOW INTERFACES to display interface parameters and statistics. Use the EXEC command SHOW X25 VC to display virtual circuit parameters and statistics.

Displaying Interface Parameters and Statistics

To display parameter information for a serial interface using the X.25 Level 3 protocol, use the EXEC command SHOW INTERFACES. To display X.25 Level 3 parameters for LAN interfaces (such as Ethernet), use the SHOW CMNS EXEC command described later in this chapter. For serial X.25 interfaces, the following is an example of output:

```
X25 address 000000010100, state R1, modulo 8, idle 0, timer 0, nvc 1
Window size: input 2, output 2, Packet size: input 128, output 128
Timers: T20 180 T21 200 T22 180 T23 180 TH 0
Channels: Incoming 1-1024 Two-way 1-1024 Outgoing 1-1024
RESTARTs 1/20 CALLs 1000+2/1294+190/0+0 DIAGs 0/0
Window is closed
```

On the first line, the `address` field indicates the calling address used in the Call Request packet. The `state` field indicates the state of the interface: R1 is the

normal-ready state, R2 indicates the DTE not-ready state, and R3 indicates the DCE not-ready state. If the state is R2 or R3, the device is awaiting acknowledgment for a Restart Request packet. The modulo field displays the modulo value, which determines the sequence numbering scheme used. The idle field shows the number of minutes the router waits before closing idle virtual circuits. The timer field displays the value of the interface timer, which is zero unless the interface state is R2 or R3. The nvc field displays the maximum number of simultaneous virtual circuits permitted to and from a single host.

On the second line, the Window size and Packet size fields show the default window and packet sizes for the interface. Each virtual circuit can override these values using facilities specified with the X25 MAP or X25 FACILITY subcommands.

The third line shows the values of the Request packet timers (T10 through T13 for a DCE device, and T20 through T23 for a DTE device). The fourth line shows the channel sequence ranges. The last line shows packet statistics for the interface using these formats:

```
sent/received\ successful+failed\ callssent+callssentfailed/\
callsreceived+callsreceivedfailed/\
callsforwarded+callsforwardedfailed\
```

Displaying General Virtual Circuit Parameters and Statistics

The EXEC command SHOW X25 VC displays the details of the active X.25 switched virtual circuits. To examine a particular virtual circuit, add an LCN argument to the SHOW X25 VC command. The following is example output:

```
LC1: 1, State: D1, Interface: Serial0
Started 0:55:03, last input 0:54:56, output 0:54:56
Connected to IP [10.4.0.32] <Ð>000000320400
Window size input: 7, output: 7
Packet size input: 1024, output: 1024
PS: 2 PR: 6 Remote PR: 2 RCNT: 1 RNR: FALSE
Retransmits: 0   Timer (secs): 0  Reassembly (bytes): 0
Held Fragments/Packets: 0/0
Bytes 1111/588 Packets 18/22 Resets 0/0 RNRs 0/0 REJs 0/0 INTs 0/0
```

On the first line, the LCI field displays the virtual circuit number. The State field displays the state of the virtual circuit (which is independent of the states of other virtual circuits); D1 is the normal ready state. (See the CCITT X.25 recommendation for a description of virtual circuit states.) The Interface field shows the interface used for the virtual circuit.

On the second line, the Started field shows the time elapsed since the virtual circuit was created, the last input field shows time of last input, and the output field shows time of last output.

The third line describes the type of connection provided. This example shows a circuit that can be used to exchange IP traffic with the network-protocol address 10.4.0.32, and that the connection was made using the displayed X.121 address. This line will change depending on the type of connectivity. For example, a locally switched PVC would cause a display on this line similar to the following: Switched PVC to serial PVC 3 connected. Before a circuit is fully established (that is, the call has not yet been accepted) this line will report that the SVC is a half-baked connection.

The Precedent field, which appears only if you have specified DDN encapsulation, indicates IP precedence.

The fourth and fifth lines show window and packet sizes. These sizes can differ from the interface default values if facilities were offered and accepted in the Call Request and Call Accepted packets.

On the sixth line, the PS and PR fields show the current send and receive sequence numbers, respectively. The Remote PR field shows the last PR number received from the other end of the circuit. The RCNT field shows the count of unacknowledged input packets. The RNR field shows the state of the Receiver Not Ready flag; this field is true of the network sends a receiver-not-ready packet.

On the seventh line, the Retransmits field shows the number of times a packet has been retransmitted. The Timer field shows a nonzero time value if a packet has not been acknowledged or if virtual circuits are being timed for inactivity. The Reassembly field shows the number of bytes received for a partial packet (a packet in which the more data bit is set).

On the eighth line, the Fragments part of the Held Fragments/Packets field shows the number of X.25 packets being held. (In this case, Fragments refers to the X.25 fragmentation of higher-level data packets.) The Packets part of the Held Fragments/Packets field shows the number of higher-level protocol packets currently being held pending the availability of resources, such as the establishment of the virtual circuit.

On the last line, the Bytes field shows the total numbers of bytes sent and received. The Packets, Resets, RNRs, REJs, and INTs fields show the total sent and received packet counts of the indicated types. (RNR is receiver not ready, REJ is reject, and INT is interrupt).

Debugging X.25

When installing an X.25 link, you can use the EXEC commands DEBUG with different keywords as follows:

debug x25

Enables the monitoring of all X.25 traffic.

debug x25-events

Enables the monitoring of all X.25 traffic but does not display information about X.25 data or acknowledgment packets.

debug x25-vc *number*

Allows you to watch the specified port number with a virtual circuit using the DEBUG X25 and DEBUG X25-EVENTS commands.

The following is example output:

```
Serial0 (704760940): X25 O R2 RESTART (5) 8 lci 0 cause 0 diag 0
Serial0 (704760944): X25 I R2 RESTART (5) 8 lci 0 cause 7 diag 0
Serial0 (704762944): X25 O D2 RESET REQUEST (5) 8 lci 1 cause 0 diag 0
Serial0 (704762952): X25 I D2 RESET CONFIRMATION (3) 8 lci 1
Serial0 (704763372): X25 O P2 CALL REQUEST (19) 8 lci 10
From(14): 31250000000101 To(14): 31109090096101
Facilities: (0)
Cannot route call
Serial0 (704763372): X25 I P6 CLEAR REQUEST (5) 8 lci 10 cause 19 diag 67
Serial0 (704763839): X25 O P6 CLEAR CONFIRMATION (3) 8 lci 10
```

For each event, the first field identifies the interface on which the activity occurred, the second field indicates the time between packets (in milliseconds), and the third field indicates that it was an X.25 event. The fourth field indicates whether the X.25 packet was input (I) or output (O). The fifth field is the state of virtual circuit as defined in the X.25 standard.

The sixth field is the type of X.25 packet that triggered the event, and the seventh field (in parentheses) gives the total length of the X.25 packet in bytes. The eighth field is the window modulus. The ninth field (labeled lci) shows the virtual circuit number. The tenth field (labeled cause) gives any cause code, and the eleventh field (labeled diag) gives any diagnostic code. Cause and diagnostic codes are not present for all messages.

For Call Request and Call Connected packets, the router shows additional information on separate lines. The From field shows the calling X.121 address and the To field shows the called X.121 address. The number in parentheses after the field name—(14) for example—specifies the number of digits in the address. The Facilities field indicates the length (in bytes) of the requested facilities and the facilities contents.

Configuring CMNS Routing Support

The DECbrouter 90 support of Connection-Mode Network Service (CMNS) provides a mechanism through which local X.25 switching can be extended to different media (for example, Ethernet) by using OSI-based NSAP addresses. This implementation essentially runs X.25 (packet level) over LLC2 (frame level) to facilitate the extension of X.25 to other media. Digital's CMNS implementation supports services defined in ISO Standards 8208 (packet level) and 8802-2 (frame level).

In addition, the DECbrouter 90 CMNS implementation allows LAN-based OSI resources, such as a DTE host and a Sun workstation, to be interconnected through the router's LAN interfaces and to a remote OSI-based DTE through a WAN interface (using, for example, an X.25 PSN).

Note

CMNS is implicitly enabled whenever an X.25 ENCAPSULATION interface subcommand is included with a serial-interface configuration.

All local mapping is performed by statically mapping MAC addresses to NSAP addresses and X.121 addresses to NSAP addresses.

Implementing CMNS routing on the DECbrouter 90 involves the following basic steps:

1. Enable CMNS on an interface.
2. Map NSAP addresses to a MAC-address or X.121 address.

The discussions that follow detail commands used to perform these steps and provide example configurations illustrating CMNS-based routing.

Enabling CMNS

To enable CMNS on a nonserial interface, use the CMNS ENABLE interface subcommand. The command syntax is as follows:

cmns enable
no cmns enable

After executing this command, all the X.25-related interface subcommands are made available on the LAN interfaces (Ethernet and others), as well as on serial interfaces. The **no** version disables this feature.

Specifying CMNS Address Mappings

After enabling CMNS on a nonserial interface (or specifying X.25 encapsulation on a serial interface), you must use the X25 MAP CMNS interface subcommand to map NSAP addresses to either MAC-layer addresses or X.121 addresses, depending on the application. The syntax for this command is:

x25 map cmns *NSAP MAC-address*
no x25 map cmns *NSAP MAC-address*

x25 map cmns *NSAP [X.121-address]*
no x25 map cmns *NSAP [X.121-address]*

The *NSAP, MAC-address,* and *X.121-address* arguments specify the NSAP address-to-MAC address or NSAP address-to-X.121 address mappings. The argument *NSAP* can be either the actual DTE NSAP address or the NSAP prefix of the NSAP address. The NSAP prefix is sufficient for a best match to route a call.

Note

For CMNS support over dedicated serial links (such as leased lines), an X.121 address is not needed, but can be optionally included. You must specify the X.121 address for CMNS connections over a packet-switched network, and you must specify a MAC address for CMNS connections over an Ethernet.

To retract a mapping, use the NO X25 MAP CMNS interface subcommand with the appropriate address arguments.

Example Address Mappings

The following examples illustrate enabling CMNS and configuring X.121 and MAC-address mappings:

```
interface ethernet 0
cmns enable
x25 map cmns 38.8261.1000.0150.1000.17 0000.0c00.ff89
! Above maps NSAP to MAC-address on Ethernet0
!
interface serial 0
encapsulation x25
x25 map cmns 38.8261.1000.0150.1000.18 3110451
! Above maps NSAP to X.121-address on Serial0
! assuming the link is over a PDN
!
interface serial 1
encapsulation x25
x25 map cmns 38.8261.1000.0150.1000.20
! Above specifies cmns support for Serial1
! assuming that the link is over a leased line
```

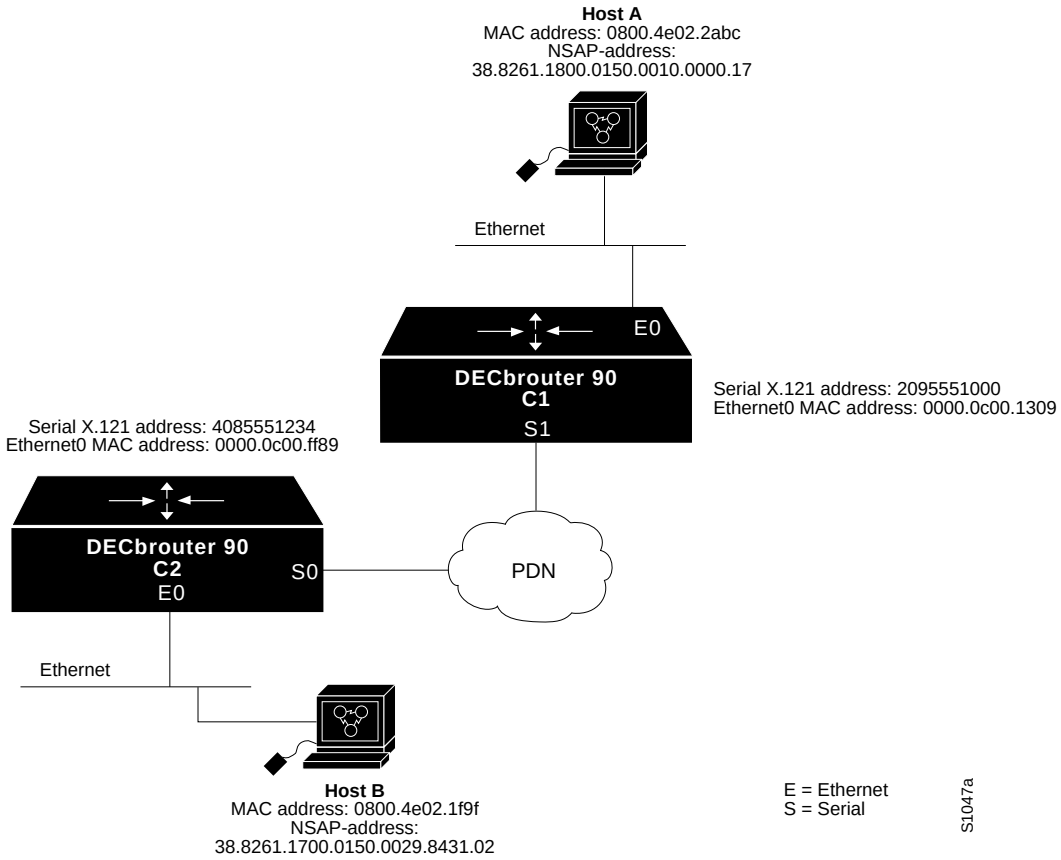
CMNS Configuration Examples

The following two examples illustrate common CMNS implementations for switching traffic over different media. The first example depicts switching CMNS over a packet-switched public data network (PDN); the second example illustrates switching CMNS over a leased line. These applications are quite similar; there are subtle differences in the use of the X25 MAP CMNS command.

Example 1: Switching CMNS over a PDN

Figure 9–8 illustrates the general network topology for a CMNS switching application where calls are being made between resources on opposite sides of a remote link to Host A and Host B, with a PDN providing the connection.

Figure 9–8 Example Network Topology for Switching CMNS over a PDN



The following configuration listing allows resources on either side of the PDN to call Host A or Host B. This configuration allows traffic intended for the remote NSAP address specified in the X25 MAP CMNS commands (for the serial ports) to be switched through the serial interface for which CMNS is configured.

The CMNS-related configuration for Router C2 in Figure 9–8 would be as follows:

```
! This configuration specifies that any traffic from any other
! interface intended for any NSAP address with NSAP prefix 38.8261.17
! will be switched to MAC address 0800.4e02.1f9f
! through Token Ring 0
!
interface ethernet 0
cmns enable
x25 map cmns 38.8261.17 0800.4e02.1f9f
!
! This configuration specifies that traffic from any other interface
! on DECbrouter 90 C2 that is intended for any NSAP address with
! NSAP-prefix 38.8261.18 will be switched to
! X.121 address 2095551000 through Serial 0
!
interface serial 0
encapsulation x25
x25 address 4085551234
x25 map cmns 38.8261.18 2095551000
```

The CMNS-related configuration for Router C1 in Figure 9–8 would be as follows:

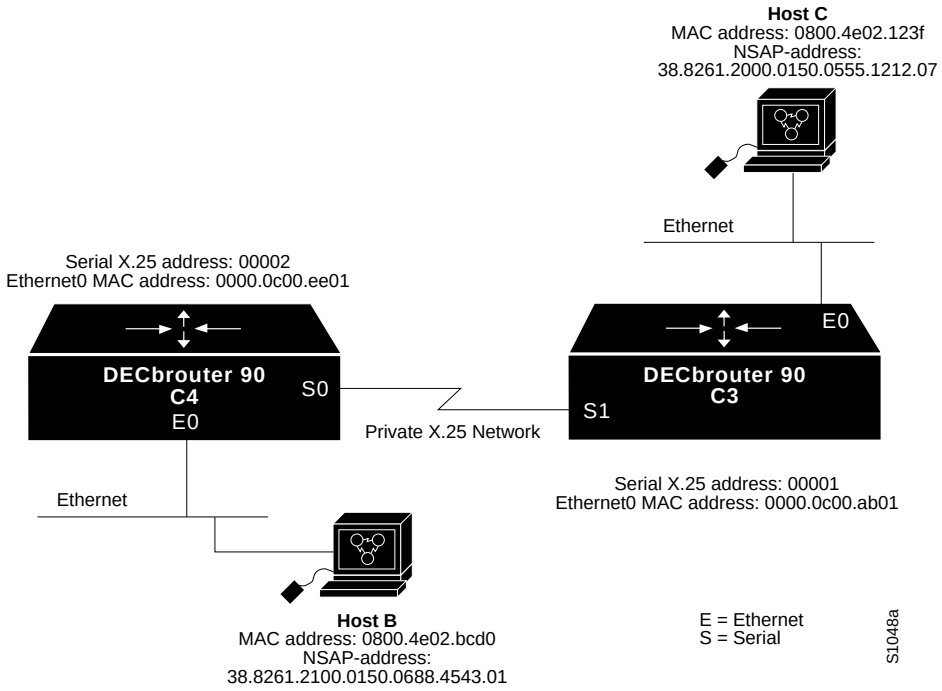
```
! This configuration specifies that any traffic from any other
! interface intended for any NSAP address with NSAP 38.8261.18
! will be switched to MAC address 0800.4e02.2abc through Ethernet 0
!
interface ethernet 0
cmns enable
x25 map cmns 38.8261.18 0800.4e02.2abc
!
! This configuration specifies that traffic from any other interface
! on DECbrouter 90 C1 that is intended for any NSAP address with
! NSAP-prefix 38.8261.17 will be switched to X.121 address
! 4085551234 through Serial 1
!
interface serial 1
encapsulation x25
x25 address 2095551000
x25 map cmns 38.8261.17 4085551234
```

Example 2: Switching CMNS over Leased Lines

Figure 9–9 illustrates the general network topology for a CMNS switching application where calls are being made by resources on opposite sides of a remote link to Host B and Host C with a dedicated leased line providing the connection.

The following configuration listing allows resources on either side of the leased line to call Host B or Host C. This configuration allows traffic intended for the remote NSAP address specified in the X25 MAP CMNS commands (for the serial ports) to be switched through the serial interface for which CMNS is configured.

Figure 9–9 Example Network Topology for Switching CMNS over a Leased Line



A key difference for this configuration in comparison with Example 1 is that with no PDN, the specification of an X.121 address in the X25 MAP CMNS command is not necessary. The specification of an X.25 address also is not needed, but is included for symmetry with Example 1.

The CMNS-related configuration for Router C4 in Figure 9–9 would be as follows:

```
! This configuration specifies that any traffic from any other
! interface intended for any NSAP address with NSAP 38.8261.21
! will be switched to MAC address 0800.4e02.bcd0 through Token Ring 0
!
interface ethernet 0
cmns enable
x25 map cmns 38.8261.21 0800.4e02.bcd0
!
! This configuration specifies that traffic from any other interface
! on DECbrouter 90 C4 that is intended for any NSAP address with
! NSAP-prefix 38.8261.20 will be switched through Serial 0
!
interface serial 0
encapsulation x25
x25 address 00002
x25 map cmns 38.8261.20
```

The CMNS-related configuration for Router C3 in Figure 9–9 would be as follows:

```
! This configuration specifies that any traffic from any other
! interface intended for any NSAP address with NSAP 38.8261.20
! will be switched to MAC address 0800.4e02.123f through Ethernet 0
!
interface ethernet 0
cmns enable
x25 map cmns 38.8261.20 0800.4e02.123f
!
! This configuration specifies that traffic from any other interface
! on DECbrouter 90 C3 that is intended for any NSAP address with
! NSAP-prefix 38.8261.21 will be switched through Serial 1
!
interface serial 1
encapsulation x25
x25 address 00001
x25 map cmns 38.8261.21
```

Monitoring CMNS Traffic Activity

To display information pertaining to CMNS traffic activity, use the SHOW CMNS EXEC command. The command syntax is as follows:

show cmns *interface-name*

The following is an example of the output of SHOW CMNS INTERFACE:

```
Ethernet 0 is up, line protocol is up
  CMNS protocol processing disabled
```

This example indicates that CMNS is disabled on Ethernet 0.

The following display sample illustrates SHOW CMNS displays for a serial interface.

```
Serial 0 is down, line protocol is down
  X25 address 222222, state R1, modulo 8, idle 0, timer 0, nvc 1
      Window size: input 2, output 2, Packet size: input 1024, output 1024
      Timers: T10 60 T11 180 T12 60 T13 60 TH 0
      Channels: Incoming 1-1024 Two-way 1-1024 Outgoing 1-1024
      RESTARTs 0/0 CALLs 0+0/0+0/0+0 DIAGs 0/0
```

Refer to "Monitoring X.25 Level 3 Operations" earlier in this provided in the preceding display.

The following display illustrates CMNS-related information presented on an Ethernet display for SHOW CMNS:

```
Ethernet 0 is up, line protocol is up
  Hardware address is 0000.0c01.1515 (bia 0000.0c01.1515), state R1
      Modulo 8, idle 0, timer 0, nvc 1
      Window size: input 2, output 2, Packet size: input 128, output 128
      Timer: TH 0
      Channels: Incoming 1-1024 Two-way 1-1024 Outgoing 1-1024
      RESTARTs 0/0 CALLs 0+0/0+0/0+0 DIAGs 0/0\
```

- The address field indicates the Hardware address (DTE address of the interface).
- The state field indicates the state of the interface. R1 is the normal ready state (this should always be R1).
- The Modulo field displays the modulo value, which determines the sequence numbering scheme used.
- The idle field shows the number of minutes the router waits before closing idle virtual circuits.
- The timer field shows the value of the interface timer and should always be zero.
- The nvc field displays the maximum number of simultaneous virtual circuits permitted to the and from a single host.
- The Window size and Packet size fields show the default window and packet size for the interface. Each virtual circuit can override these values using facilities specified with the X25 FACILITY subcommand.
- The timer field indicates the value of timer TH is for the X.25 hold timer for delayed acknowledgment.
- The Channels field displays the channel sequence range for this interface per LLC2 connection.

On the last line, packet statistics for the interface are displayed which use the following format:

- RESTARTs—Restarts sent/received
- CALLS—Successful calls+failed calls/calls sent+calls failed/calls received+calls failed
- DIAGs—-Diagnostic messages sent+received.

Monitoring CMNS with LLC2 Parameters

There is an addendum to the SHOW LLC command in the case of CMNS calls; for example, an LLC2 connection provides the following information:

```
LLC2 Connections to:
Ethernet0 DTE=0800.4E02.1F9F,000000000000 SAP=7E/7E, State=NORMAL
   V(S)=2, V(R)=3, Last N(R)=2, Local window=7, Remote Window=7
   akmax=3, n2=8, Next timer in 5196
   xid-retry timer     0/60000   ack timer       0/1000
   p timer             0/1000    idle timer   5196/10000
   rej timer           0/3200    busy timer      0/9600
   akdelay timer       0/3200
```

```
CMNS Connections to:
   Address 0800.4E02.1F9F via Ethernet0
   Protocol is up
   Interface Type X25-DCE RESTARTS 0/1
   Timers: T10 60 T11 180 T12 60 T13 60
```

The general LLC2-related information in the preceding display is documented in the *DECbrouter 90 Configuration and Reference, Volume 3*. The following information defines fields in the CMNS Connections to: portion of the display.

On the first line, the Address field is the Remote DTE address that connected to via the LAN interface (in this example, Ethernet0).

On the second line, Protocol is up indicates that an existing LLC2 connection exists with the remote DTE and has passed X.25 initialization (Restart negotiation) of that LLC2 connection. Therefore, the link is ready for any incoming/outgoing call request on this LLC2 connection.

On the third line, the Interface Type field indicates the type of this LLC2 connection. Options are X25-DCE, X25-DTE or X25-DXE. The RESTARTS field represents the number of restarts sent/received on this LLC2 connection.

On the last line, Timers shows the value for the Request packet timers for this LLC2 connection (T10 through T13 for a DCE interface type; T20 through T23 for a DTE interface type).

Displaying CMNS Virtual Circuit Parameters and Statistics

When the protocol type used for the connection is CMNS, the display generated with SHOW X25 VC differs slightly from the display outlined in the preceding description.

A sample display output follows that depicts two complementary interfaces, both running CMNS, providing transport of CMNS traffic to each other:

```
LCI: 1,  State: P4,  Interface: Serial1
Started 0:23:00, last input never, output never
Connected to CMNS [37.1111] <--> 313131 via Ethernet0 LCN 4095
to 0000.0c01.487d
Window size input: 6, output: 6
Packet size input: 1024, output: 1024
PS: 0  PR: 0  ACK: 0  Remote PR: 0  RCNT: 0  RNR: FALSE
Retransmits: 0  Timer (secs): 0  Reassembly (bytes): 0
Held Fragments/Packets: 0/0
Bytes 0/0 Packets 0/0 Resets 0/0 RNRs 0/0 REJs 0/0 INTs 0/0
--More--
LCI: 4095,  State: P4,  Interface: Ethernet0
Started 0:23:01, last input never, output never
Connected to CMNS [36.3030.3030.3030.30] <--> 0000.0c01.487d
via Serial1 LCN 1to 313131
Window size input: 6, output: 6
Packet size input: 1024, output: 1024
PS: 0  PR: 0  ACK: 0  Remote PR: 0  RCNT: 0  RNR: FALSE
Retransmits: 0  Timer (secs): 0  Reassembly (bytes): 0
Held Fragments/Packets: 0/0
Bytes 0/0 Packets 0/0 Resets 0/0 RNRs 0/0 REJs 0/0 INTs 0/0
```

The following display fields differ for CMNS virtual circuits:

- On the first line, the LCI field displays the virtual circuit number; range is 1 to 4095. The State field displays the state of the virtual circuit (which is independent of the states of other virtual circuits); P4 indicates the interface is in the data transfer state. (See the CCITT X.25 recommendation for a description of virtual circuit states.) The Interface field shows the interface

used for the virtual circuit. With CMNS, this can indicate Ethernet as well as Serial.

- On the third line, the Connected to field shows in brackets the NSAP address for the device at the indicated X.121 address. For Serial1, it also indicates the logical channel number (LCN) used (1 to 4095) and the MAC address of the node to which the interface is connected.

Debugging CMNS

When installing a link on which CMNS is to be used, you can use the EXEC command DEBUG with different keywords as follows:

debug x25

Enables the monitoring of all X.25 traffic, including CMNS-based traffic.

debug x25-events

Enables the monitoring of all X.25 traffic, but does not display information about X.25 data or acknowledgment packets.

debug x25-vc *number*

Allows you to watch the specified port number with a virtual circuit using the DEBUG X25 and DEBUG X25-EVENTS commands.

Configuring Frame Relay

This section describes frame relay configuration, the DECbrouter 90 implementation of frame relay, the protocols supported, and the hardware needed to operate with a frame relay network.

Frame relay is described as an encapsulation method and is directed mainly to users with large T1 network installations. The DECbrouter 90 supports the two generally implemented specifications for frame relay logical management interfaces (LMIs):

- The *Frame Relay Interface* specification produced by Northern Telecom, Digital Equipment Corporation, Stratacom, and Cisco Systems.
- The ANSI-adopted frame relay specification,T1.617 Annex D.

The DECbrouter 90 implementation also conforms to the link access procedure (LAP-D) defined by the CCITT under its I-series (ISDN) recommendation as I122, "Framework for Additional Packet Model Bearer Services," and IETF encapsulation in accordance with RFC 1294.

The DECbrouter 90 frame relay implementation currently supports routing on AppleTalk, DECnet, IP, ISO CLNS, Novell IPX, VINES, and transparent bridging.

The DECbrouter 90 supports the local management interface (LMI), as specified in the joint *Frame Relay Interface* specification. The LMI includes support for a keepalive mechanism, a multicast group, and a status message.

The keepalive mechanism provides an exchange of information between the router and the switch to verify data is flowing. The multicast mechanism provides the router with its local data link connection identifier (DLCI) and the multicast DLCI. The status mechanism provides an ongoing status report on the DLCIs known by the switch.

The DECbrouter 90 frame relay implementation sends congestion information from frame relay to DECnet Phase IV and CLNS. This mechanism promotes forward explicit congestion notification (FECN) and backward explicit congestion notification (BECN) bits from the frame relay layer to upper layer protocols after checking for the FECN or BECN bit on the incoming DLCI. Use this frame relay congestion information to adjust the sending rates of end hosts. FECN/BECN-bit promotion is enabled by default on any interface using frame relay encapsulation. No configuration is required.

Note

When configuring IP routing over frame relay, you may need to make adjustments to accommodate split-horizon effects. Refer to the *DECbrouter 90 Configuration and Reference, Volume 2* for details about how the router handles possible split horizon conflicts. By default, split horizon is *disabled* for frame relay networks.

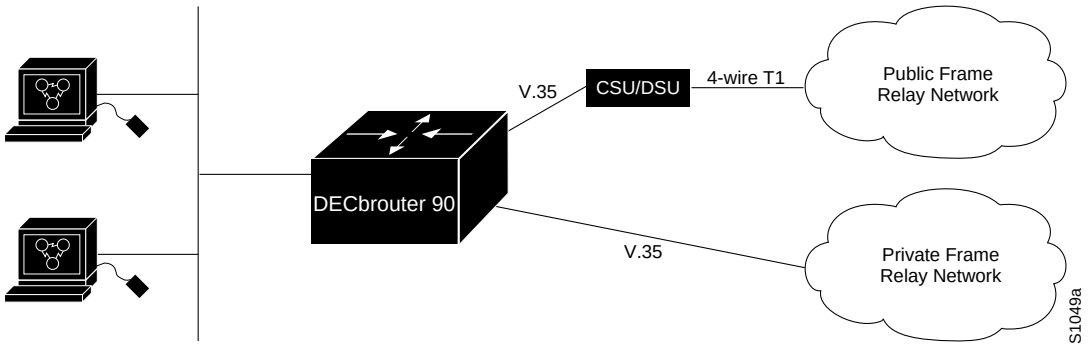
Configuring the Hardware

The following hardware configuration is required for frame relay connections:

- Each router connects directly to the frame relay switch.
- Each router connects directly to a channel service unit/digital service unit (CSU/DSU) first, and is connected to a remote frame relay switch.

The CSU/DSU converts V.35 or RS-449 signals to the properly coded T1 transmission signal for successful reception by the frame relay network. Figure 9–10 illustrates the connections between the different components.

Figure 9–10 Frame Relay Physical Connection



The frame relay interface actually consists of one physical connection between the router and the switch that provides the service. This single physical connection provides direct connectivity to each device on a network.

Specifying Frame Relay Encapsulation

Use the ENCAPSULATION FRAME-RELAY interface subcommand to specify frame relay encapsulation on an interface. The DECbrouter 90 frame relay feature supports IP encapsulation in conformance with RFC 1294, allowing interoperability between multiple vendors. Use the IETF form of frame relay encapsulation if your router is connected to another vendor's equipment across a frame relay network.

encapsulation frame-relay [**ietf**]
no encapsulation frame-relay [**ietf**]

Examples

These commands configure frame relay encapsulation on interface serial 1:

```
interface serial 1
encapsulation frame-relay
```

These commands configure frame relay encapsulation in conformance with RFC 1294 on interface serial 1. Use the **ietf** keyword if your communication server is connected to another vendor's equipment across a frame relay network.

```
interface serial 1
encapsulation frame-relay ietf
```

Enabling ANSI Frame Relay LMI

Use the FRAME-RELAY LMI-TYPE ANSI interface subcommand to specify use of the ANSI LMI. The command syntax is as follows:

frame-relay lmi-type ANSI
no frame-relay lmi-type ANSI

This command specifies the exchange of LMI messages as defined by ANSI standard T1.617.

The NO FRAME-RELAY LMI-TYPE ANSI returns the LMI type to the default as defined by the Cisco/Stratacom/Northern Telecom/Digital specification. The LMI type is set on a per-interface basis and is shown in the output of the SHOW INTERFACE command.

Example

The following commands configure frame relay encapsulation on serial 0 and specify use of the ANSI LMI:

```
interface serial 0
encapsulation frame-relay
frame-relay lmi-type ANSI
```

Setting the Frame Relay Keepalive Timer

The FRAME-RELAY KEEPALIVE interface subcommand enables and disables the LMI mechanism for serial lines using the frame relay encapsulation. The full syntax of this command follows.

frame-relay keepalive *seconds*
no frame-relay keepalive

The argument *seconds* defines the keepalive interval. The interval must be set and must match the interval set on the switch. The default keepalive interval is ten seconds.

Note

The FRAME-RELAY KEEPALIVE and KEEPALIVE commands perform the same function; both commands enable the keepalive sequence. The keepalive sequence is part of the LMI protocol, so these commands also control the enabling and disabling of the LMI.

Example

This command sets the keepalive timer on the router for a period that is two or three seconds faster (shorter interval) than the interval set on the keepalive timer of the frame relay switch. The difference in keepalive intervals ensures proper synchronization between the router and the frame relay switch.

```
frame-relay keepalive 8
```

Mapping Between an Address and the DLCI

The FRAME-RELAY MAP subcommand defines the mapping between an address and the DLCI used to connect to the address. There can be many DLCIs known by a router that can send data to many different places, but all this data will be multiplexed over the one physical link. The frame relay map tells the router how to get from a specific protocol and address pair to the correct DLCI. The full syntax of this command follows.

frame-relay map *protocol protocol-address DLCI* [**broadcast**] [**ietf**)]
no frame-relay map *protocol protocol-address*

The argument *protocol* is one of these keywords: **ip, decnet, appletalk, xns, novell, vines,** or **clns**. The keyword is followed by the corresponding *protocol address* and the DLCI number. The optional keyword **broadcast** specifies that broadcasts should be forwarded to this address when multicast is not enabled. See the section Defining a DLCI for Multicast for more information about the multicast command.

The optional keyword *ietf* is used to select RFC 1294 encapsulation. If you are using the ietf option, the argument *protocol* will be ip. Use the IETF form of the map command if your router is connected to another vendor's equipment across a frame relay network.

The **no** version of the command deletes the entry.

Examples

This command maps IP address 131.108.123.1 to DLCI 100.

```
frame-relay map IP 131.108.123.1 100
```

The commands in the following example, which set IETF encapsulation at the interface level, result in the same configuration as the commands used in the next example, which set IETF on a per-DLCI basis. In this example, the keyword **ietf** sets the default encapsulation method for all maps to IETF.

```
encapsulation frame-relay IETF
frame-relay map ip 131.108.123.2 48 broadcast
frame-relay map ip 131.108.123.3 49 broadcast
```

In the following configuration, IETF encapsulation is configured on a per-DLCI basis. This configuration has the same result as the configuration in the previous example.

```
encapsulation frame-relay
frame-relay map ip 131.108.123.2 48 broadcast ietf
frame-relay map ip 131.108.123.3 49 broadcast ietf
```

The following configuration provides backward compatibility and interoperability. These functions are possible because of the flexibility provided by defining each map entry separately.

```
encapsulation frame-relay
frame-relay map ip 131.108.123.2 48 broadcast ietf
!interoperability is provided by ietf encapsulation
frame-relay map ip 131.108.123.3 49 broadcast ietf
frame-relay map ip 131.108.123.7 58 broadcast
! this line allows the router to connect with a device
running an older version of software
frame-relay map DECNET 21.7 49 broadcast
```

Configure IETF based on map entries and protocol for more flexibility. Use this method of configuration for backward compatibility and interoperability.

Configuring for ISO CLNS

This variation of the FRAME-RELAY MAP command is used for the ISO CLNS protocol:

frame-relay map clns *DLCI* **broadcast**

This variation of the FRAME-RELAY MAP command is used for bridging:

frame-relay map bridge *DLCI* **broadcast**

In both these variations, there is no need to specify a protocol address; however, the **broadcast** keyword is required for both.

The optional keyword **broadcast** specifies that broadcasts should be forwarded to this address when the multicast is not enabled. The default is not to forward broadcasts. The **broadcast** keyword is required for ISO CLNS and bridging applications.

The NO FRAME-RELAY MAP subcommand with the appropriate arguments deletes the entry.

Examples

This command maps IP address 131.108.123.1 to DLCI 100. Broadcasts are not forwarded.

```
frame-relay map IP 131.108.123.1 100
```

This command uses DLCI 144 for bridging.

```
frame-relay map bridge 144 broadcast
```

This command uses DLCI 125 for ISO CLNS routing.

```
frame-relay map clns 125 broadcast
```

Requesting Short Status Messages

The FRAME-RELAY SHORT-STATUS interface subcommand instructs the router to request the short status message from the switch (see Version 2.3 of the joint *Frame Relay Interface* Specification from Stratacom, August 9, 1990). The full syntax of this command follows.

frame-relay short-status
no frame-relay short-status

The default is to request the full status message. Use the NO FRAME-RELAY SHORT-STATUS command to override the default.

Example

This command returns the interface to the default state of requesting full status messages.

```
no frame-relay short-status
```

Setting a Local DLCI

The FRAME-RELAY LOCAL-DLCI interface subcommand sets the source DLCI for use when the LMI is not supported. If LMI is supported and the multicast information element is present, the DECbrouter 90 sets its local DLCI based on information provided via the LMI. The full syntax of this command follows.

frame-relay local-dlci *number*
no frame-relay local-dlci

The argument *number* is the local, or source, DLCI number. The NO FRAME-RELAY LOCAL-DLCI command removes the DLCI number.

Note

The FRAME-RELAY LOCAL-DLCI command is provided mainly to allow testing of the frame relay encapsulation in a setting where two routers are connected back-to-back. This command is not required in a live frame relay network.

Example

This command specifies 100 as the local DLCI.

```
frame-relay local-dlci 100
```

Defining a DLCI for Multicast

The FRAME-RELAY MULTICAST-DLCI interface subcommand defines a DLCI to be used for multicasts. The full syntax of this command follows.

frame-relay multicast-dlci *number*
no frame-relay multicast-dlci

This command should be used only when the multicast facility is not supported. Network transmissions (packets) sent to a multicast DLCI are delivered to all routers defined as members of the multicast group. The multicast DLCI is identified by the argument *number*. (Note that this is not the multicast group number, which is an entirely different value.) The NO FRAME-RELAY MULTICAST-DLCI command removes the multicast group.

Note

The FRAME-RELAY MULTICAST-DLCI command is provided mainly to allow testing of the frame relay encapsulation in a setting where two routers are connected back-to-back. This command is not required in a live frame relay network.

Example

This command specifies 1022 as the multicast DLCI.

```
frame-relay multicast-dlci 1022
```

Frame Relay Configuration Examples

The following examples illustrate how to configure a router to support frame relay connections.

Two Routers in Static Mode

The following examples illustrate how to configure two routers for static mode.

Example 1—First Router

```
interface  serial 0
ip address  131.108.64.2 255.255.255.0
encapsulation  frame-relay
frame-relay keepalive 10
frame-relay map ip  131.108.64.1 43
```

Example 2—Second Router

```
interface  serial 0
ip address  131.108.64.1 255.255.255.0
encapsulation  frame-relay
frame-relay keepalive 10
frame-relay map ip  131.108.64.2 44
```

Routing DECnet Packets

The following example illustrates how to send all DECnet packets destined for address 56.4 out on DLCI 101. In addition, any DECnet broadcasts for interface serial 1 will be sent on the DLCI.

Example

```
interface  serial 1
decnet routing 32.6
encapsulation  frame-relay
frame-relay map decnet 56.4 101 broadcast
```

Routing Novell Packets

The following example illustrates how to send packets destined for Novell address 200.0000.0c00.7b21 out on DLCI 102.

Example

```
interface  ethernet 0
novell network 2abc
!
interface serial 0
novell network 200
encapsulation frame-relay
frame-relay map novell 200.0000.0c00.7b21 102 broadcast
!
```

Monitoring Frame Relay

Use the EXEC commands in this section to monitor frame relay connections.

Monitoring the Frame Relay Interface

When using the frame relay encapsulation, the EXEC command SHOW INTERFACE includes information on the multicast DLCI, the DLCI of the interface, and the LMI DLCI used for the LMI.

The multicast DLCI and the local DLCI can be set using the FRAME-RELAY MULTICAST-DLCI and the FRAME-RELAY LOCAL-DLCI subcommands, or provided through the local management interface. The status information is taken from the LMI, when active.

Enter this command at the EXEC prompt:

show interfaces *[type unit]*

Following is sample output.

```
router> show interfaces serial 1

Serial 1 is up, line protocol is up
  Hardware type is MK5025
  Internet address is 131.108.122.1, subnet mask is 255.255.255.0
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME-RELAY, loopback not set, keepalive set (10 sec)
  multicast DLCI 1022,  status defined, active
  source DLCI    20, status defined, active
  LMI DLCI 1023, LMI sent 10, LMI stat recvd 10, LMI upd recvd 2
  Last input 7:21:29, output 0:00:37, output hang never
  Output queue 0/100, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 0 bits/sec, 0 packets/sec
     47 packets input, 2656 bytes, 0 no buffer
     Received 5 broadcasts, 0 runts, 0 giants
     5 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 57 abort
     518 packets output, 391205 bytes
     0 output errors, 0 collisions, 0 interface resets, 0 restarts
     1 carrier transitions
```

In this display, the multicast DLCI has been changed to 1022 with the FRAME-RELAY MULTICAST-DLCI interface subcommand. The statistics for the LMI are the number of status inquiry messages sent (LMI sent), the number of status messages received (LMI recvd), and the number of status updates received (upd recvd). See the *Frame Relay Interface* specification for additional explanations of this output. Chapter 7 provides an explanation about the other fields seen in the SHOW INTERFACES command.

Monitoring ANSI Frame Relay

Use the SHOW INTERFACE EXEC command to determine the LMI type implemented. The following sample display illustrates the resulting display from a SHOW INTERFACE command executed for a serial interface with the ANSI LMI enabled.

```
router> show interface serial 1

Serial 1 is up, line protocol is up
  Hardware is MK5025
  Internet address is 131.108.121.1, subnet mask is 255.255.255.0
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME-RELAY, loopback not set, keepalive set
  LMI DLCI    0, LMI sent 10, LMI stat recvd 10
  LMI type is ANSI Annex D
  Last input 0:00:00, output 0:00:00, output hang never
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 1 packets/sec
  Five minute output rate 1000 bits/sec, 1 packets/sec
     261 packets input, 13212 bytes, 0 no buffer
     Received 33 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     238 packets output, 14751 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets, 0 restarts
```

Two fields distinguish the use of the ANSI LMI from the default LMI. In the sixth line of this display, the LMI DLCI field indicates a 0 value. If the LMI were set to the default, the value would be 1023. In addition, the next line is added to the display, stating that the LMI type is ANSI Annex D.

Displaying Frame Relay Map Entries

Use this EXEC command to display the current frame relay map entries and information about these connections:

show frame-relay map

Sample output follows.

```
Serial1: IP 131.108.122.2 dlci 10(0XA,0xA0), dynamic
         status defined, active
```

The display lists the interface, the protocol, the protocol address, and the DLCI being used to reach this address. If the optional broadcast keyword was entered for a static map entry, this also will be shown.

The DLCI is displayed in three forms. For example, if the DLCI is 10, the representations would be `10(0xA,0xA0)`. The displays show the decimal value, the hexadecimal value, and the value of the DLCI as it would appear on the wire. In addition, the display indicates whether this is a static or dynamic entry. Status information for the DLCI is displayed if provided by the LMI.

For the CLNS protocol, the map has the following form:

```
Serial0: CLNS  dlci 100(0X64,1840), static, broadcast, BW = 64000
         status defined, active
Serial0: CLNS  dlci 102(0X66,1860), static, broadcast, BW = 64000
         status defined, active
```

Displaying Global Frame Relay Statistics

Use the SHOW FRAME-RELAY TRAFFIC command to display global frame relay statistics. Enter this command at the EXEC prompt:

show frame-relay traffic

Sample output follows.

```
router> show frame-relay traffic

Frame Relay statistics:
ARP requests sent 14, ARP replies sent 0
ARP request recvd 0, ARP replies recvd 10
LMI sent 10, LMI stat recvd 10, LMI upd recvd 2, Multicast sent 48
```

Statistics for all frame relay interfaces in the router also are included in this display.

Debugging Frame Relay

Use the EXEC commands described in this section to troubleshoot and monitor activity on the interface configured for frame relay. For each DEBUG command, there is a corresponding UNDEBUG command that turns messaging off.

debug frame-relay-events

Enables logging of key events in the transmission or receipt of packets encapsulated using frame relay.

debug frame-relay-lmi

Enables logging of information on the local management interface packets exchanged between the router and the frame relay service provider.

debug frame-relay-packet

Displays all packets being sent out on the frame relay network. The display identifies the output interface, the protocol identifier, and the size of the packet being sent.

Configuring Switched Multimegabit Data Services (SMDS)

The switched multimegabit data service (SMDS) is a wide-area networking service offered by the regional bell operating companies (RBOCs) and other telephone service carriers such as AT&T and MCI/Sprint. The SMDS protocol is based on cell relay technology as defined in the Bellcore Technical advisories, which is in turn based on the IEEE 802.6 Standard (also called the distributed queue dual bus (DQDB) metropolitan area network media access control protocol). For technical references, please see the bibliography in the "References and Recommended Readings" list at the end of this publication.

The DECbrouter 90 provides an interface to an SMDS network using DS1 transmission facilities at the rate of 1.544 Mbps. Connection to the network is made through a device called an SDSU—an SMDS CSU/DSU (Channel Service Unit/Digital Service Unit) developed jointly by Cisco Systems and Kentrox. The SDSU attaches to the DECbrouter 90 through an RS-449 connection. On the other side, the SDSU terminates a DS1 line.

The DECbrouter 90 implementation of SMDS supports the IP, DECnet, AppleTalk, XNS, Novell IPX, Ungermann-Bass Net/One, and OSI internetworking protocols. IP routing is fully dynamic; that is, the routing tables are determined and updated dynamically. Routing of the other supported protocols requires that you establish a static routing table of SMDS neighbors in a user group. Once this is set up, all interconnected routers provide dynamic routing.

This section describes the DECbrouter 90 implementation of SMDS and how to configure, maintain, and debug SMDS.

Note

When configuring IP routing over SMDS, you may need to make adjustments to accommodate split-horizon effects. Refer to the *DECbrouter 90 Configuration and Reference, Volume 2* for details about how the DECbrouter 90 handles possible split-horizon conflicts. By default, split horizon is *disabled* for SMDS networks.

Special SMDS Requirements

You need the following hardware, equipment, and special software to configure the SMDS implementation serial port of the DECbrouter 90.

- RS-449 transition cable
- The SDSU device

Configuring SMDS

Follow these steps to configure SMDS service on the router:

1. Determine the protocols you will be running over SMDS. Obtain from the service provider the group addresses that you will need to support those protocols.
2. Obtain the SMDS hardware (individual) address from the service provider for each router that will interface directly into the SMDS network (that is, customer premises equipment).

Note

You also will need to know the addresses of the other routers with which you will be communicating to set up the static routing tables. Note that static mapping is needed only for protocols other than IP or CLNS. For IP and CLNS, the routing is fully dynamic. (For more details, see related routing chapters in this manual.)

3. Configure the desired serial interface with SMDS encapsulation.
4. Set up the static map for the desired protocols using the SMDS STATIC-MAP command.
5. Set up the multicast map for the desired protocols using the SMDS MULTICAST command.

Using SMDS Addresses

All addresses for SMDS service are assigned by the service provider and can be assigned to individuals and groups.

A group address (also defined as a multicast address) is entered in the SMDS configuration software using an E1 prefix. A C1 prefix is used to specify individual addresses. The software expects the addresses to be entered in a slightly modified E.164 format. E.164 format is 64 bits. The first four bits are type code followed by 10 BCD digits padded to the full 60 bits with ones.

An example of an E.164 address follows.

```
C14155561313FFFF
```

Note

To simplify the addresses, the full E.164 address is not required. The trailing FFFFs are not needed. They are not displayed, and it is not necessary to type them when entering an address.

The addresses can be entered with periods similar to Ethernet-style notation, or simply as a string of digits.

An example of an individual address entered in Ethernet-style notation looks like this:

```
C141.5555.1212
```

An example of a group address looks like this:

```
E18009999999
```

Enabling SMDS

Enter the ENCAPSULATION SMDS interface subcommand to enable SMDS service on the desired interface:

encapsulation smds

The interface to which this command applies must be a serial interface. All subsequent SMDS configuration commands only apply to an interface with encapsulation SMDS.

Example

Following is an example of the commands you use to configure the SMDS service on interface serial 0:

```
!
interface serial 0
encapsulation smds
```

Note

The maximum packet size allowed in the SMDS specifications (TA-772) is 9188. This is larger than what can be used by routers with most media. Therefore, the MTU defaults to 1500 bytes to be consistent with Ethernet. If a larger MTU is used, the MTU command must be used before the ENCAPSULATION SMDS command is used.

Specifying the SMDS Address

Enter the SMDS ADDRESS interface subcommand to specify the SMDS individual address for a particular interface. The format of the command follows.

smds address *smds-address*
no smds address *smds-address*

Enter an individual address provided by the SMDS service provider for the argument *smds-address*. Enter the address as described earlier in this chapter in the section Using SMDS Addresses. This address is protocol-independent.

Enter the NO SMDS ADDRESS command to remove the address from the configuration file.

There is no default for this command.

Example

The following example shows how to specify an individual address.

```
!
interface serial 0
smds address C141.5797.1313
!
```

Note

If bridging is enabled on any interface, the SMDS address is erased and must be reentered.

Enabling the Address Resolution Protocol

Enter the SMDS ENABLE-ARP interface subcommand to enable the address resolution protocol (ARP). The full syntax of this command follows.

smds enable-arp
no smds enable-arp

The multicast address for ARP must be set before this command is issued.

By default, ARP is not enabled. Once ARP has been enabled, use the NO SMDS ENABLE-ARP command to return the line to the default state.

Defining a Static Map for an Individual Address

Enter the SMDS STATIC-MAP interface subcommand to configure a static mapping between an individual SMDS address and a higher-level protocol address. The full syntax of the command follows.

smds static-map *protocol-type protocol-address smds-address*
no smds static-map *protocol-type protocol-address smds-address*

Do not enter this command for broadcast or multicast addresses. For those addresses, use the SMDS MULTICAST interface subcommand described in the section Mapping to a Multicast Address later in this chapter.

Enter the name of the protocol for the *protocol-type* argument, followed with the address (specified in the corresponding protocol address format) in place of the *protocol-address* argument. Provide the SMDS address for the *smds-address* argument to complete the mapping. You must use one of these keywords to define the protocol type: **ip, decnet, appletalk, xns, novell,** or **clns**.

Use the NO SMDS STATIC-MAP command with the appropriate arguments to remove the map.

Example

Following is an example of the SMDS STATIC-MAP command:

```
!
smds static-map XNS 111.00C0.2711.0123  C141.5688.1212
!
```

The command will map XNS address 111.00C0.2711.0123 to the individual SMDS address C141.5688.1212.

Mapping to a Multicast Address

Enter the SMDS MULTICAST interface subcommand to map an SMDS group address to a broadcast or multicast address used by higher-level protocols. The full syntax of the command follows.

smds multicast *protocol-type smds-address*
no smds multicast *protocol-type smds-address*

Enter the name of the protocol for the *protocol-type* argument and follow with the SMDS address to answer the *smds-address* argument to complete the mapping. You can use these keywords to define the protocol type:

- **ip**—IP
- **arp**—ARP
- **decnet**—DECnet
- **decnet_router**—DECnet multicast address for all routers
- **decnet_node**—DECnet multicast address for all end systems
- **appletalk**—AppleTalk
- **aarp**—AppleTalk ARP address
- **xns**—XNS
- **novell**—Novell IPX

- **clns**—ISO CLNS
- **clns_is**—Multicast address for all CLNS Intermediate Systems
- **clns_es**—Multicast address for all CLNS End Systems

Since SMDS does not incorporate broadcast addressing, a group address for a particular protocol must be defined to serve the broadcast function. There is no default for this command.

Use the NO SMDS MULTICAST command with the appropriate arguments to remove a multicast address.

Example

Following is an example of the SMDS MULTICAST command:

```
!
smds multicast IP  E180.0999.9999
!
```

The command maps the IP broadcast address to the SMDS group address E180.0999.9999.

Enabling the AT&T SMDS Service

The DECbrouter 90 implementation of SMDS includes a configuration option that allows you to enable or disable the router's ability to interface to an AT&T SMDS switch that implements AT&T's SMDS d15-mode. Please consult with your service provider to find out whether this command is needed.

Use the SMDS D15-MODE interface subcommand in order to operate with an AT&T SMDS switch that implements the AT&T d15-mode packet structure. The command syntax follows.

smds d15 mode
no smds d15-mode

When this switch is disabled using the NO SMDS D15-MODE specification, the router will use AT&T's d11 packet structure.

When this switch is enabled, the router will use AT&T d15 packet structure. It should be off for systems that have not been upgraded.

Use the NO SMDS D15-MODE command to turn this function off. By default, the function is on.

Configuring Specific Protocols

An SMDS network can be thought of in much in the same way as an X25 cloud. The premises equipment, in this case a router, represents the edge of the cloud. The service provider enables communication across the cloud. However, proper configuration is needed for communication to occur. This configuration will differ between protocol families.

One major difference between protocol families is dynamic versus static routing among the routers (called remote peers) on the periphery of the cloud. For IP and CLNS, routing across the SMDS cloud is fully dynamic. No action on the user's part is needed for mapping of higher-level protocol addresses to SMDS addresses to occur. For the other supported protocols, you must make a static entry for each

of the other neighbors. This entry provides a router with the information that it needs to communicate with all other neighbor routers.

Up until now, this discussion has centered on the peer routers. What about all of the end nodes and routers behind the SMDS router? The static entries only need to be made for those routers that are SMDS remote peers. Nothing additional needs to be done in order to communicate with other nodes behind the peer routers.

Some protocol families need separate definitions for associated subprotocols. The next sections illustrates how to implement these kind of configurations. See Table 9–8 for a list of protocol families and the multicasts that are needed.

Table 9–8 Protocol Families and Types of Multicasts Needed

Protocol Family	Multicasts Needed
IP	IP, ARP
DECnet	DECNET, DECNET_NODE, DECNET_ROUTER
CLNS	CLNS, CLNS_ES,CLNS_IS
Novell	NOVELL
XNS	XNS
AppleTalk	APPLETALK,AARP

Configuring IP

Both IP and ARP should be configured with the SMDS MULTICAST command. ARP should be enabled. The results of the ARP activity can be shown with the SHOW ARP command. If desired, static ARP entries can be made by using this command:

arp *ip-address address* **smds**

The argument *ip-address* is the IP address. The argument *address* is the SMDS address. The keyword **smds** appended to the end of this command is required.

Configuring AppleTalk

Currently, dynamic address assignment does not work. Therefore, an AppleTalk address must be assigned to the interface, and each remote router peer must be listed with an SMDS STATIC-MAP command. The AppleTalk ARP (AARP) multicast address must still be configured with the SMDS MULTICAST command. ARP should be enabled.

Configuring XNS and Novell

For XNS and/or Novell, the multicast address must be configured.

For Novell, RIP Routing packets, SAP packets, NetBios Name Lookups, directed broadcasts, and traffic to the helper addresses (if that helper address is a broadcast address) will be sent to the SMDS Novell multicast address.

For XNS, only RIP, directed broadcasts, and helper traffic will be sent to the XNS multicast address.

For XNS and/or Novell configurations, a static map entry must be made for each remote peer.

Configuring CLNS

Multicasts must be configured for CLNS_ES and CLNS_IS. No static maps are necessary. ESHs, ISHs, and Router Hellos are sent to the multicast address, and neighbor entries are created automatically.

IP Fast Switching

SMDS fast switching of IP packets provides faster packet transfer on serial links with speeds above 56K. Use fast switching if you use high-speed, packet-switched, datagram-based WAN technology (such as X.25 or frame relay) offered by telephone companies.

Configuring Fast Switching

This section describes the commands to use to configure fast switching To configure the interface, use the following command:

interface type unit

The argument *type* is the type of interface, and unit is the number of the interface.

To configure the interface for SMDS encapsulation, use the following command:

encapsulation smds

Use the following command to configure the interface for fast switching:

ip route-cache

Example

In the following example, serial interface number 5 is configured for SMDS encapsulation and fast switching of IP packets.

```
interface serial 5
encapsulation smds
ip route-cache
```

Debugging Fast Switching

The debug serial-interface debugging command prints only condensed SMDS packet information. For more debugging information, use the serial-packet debugging command. See Debugging the Serial Interface section in Chapter 7 of this manual.

SMDS Configuration Examples

This section provides some examples of configurations to use as models for your configuration files.

Typical Multiprotocol Configuration

Following is a typical interface configured for IP, DECnet, ISO CLNS, Novell, XNS, and AppleTalk.

Example

```
interface Serial 1
ip address 1.1.1.2 255.0.0.0
decnet cost 4
appletalk address 92.1
appletalk zone smds
clns router igrp FOO
novell net 1a
xns net 17
encapsulation SMDS
! smds configuration follows
smds address c120.1580.4721
no smds d15-mode
smds static-map APPLETALK 92.2 c120.1580.4592
smds static-map APPLETALK 92.3 c120.1580.4593
smds static-map APPLETALK 92.4 c120.1580.4594
smds static-map NOVELL 1a.0c00.0102.23ca c120.1580.4792
smds static-map XNS  17.0c00.0102.23ca c120.1580.4792
smds static-map NOVELL 1a.0c00.0102.23dd c120.1580.4728
smds static-map XNS 17.0c00.0102.23aa c120.1580.4727
smds multicast NOVELL e180.0999.9999
smds multicast XNS e180.0999.9999
smds multicast ARP e180.0999.9999
smds multicast IP e180.0999.9999
smds multicast APPLETALK e180.0999.9999
smds multicast AARP e180.0999.9999
smds multicast CLNS_IS e180.0999.9990
smds multicast CLNS_ES e180.0999.9990
smds multicast DECNET_ROUTER e180.0999.9992
smds multicast DECNET_NODE e180.0999.9992
smds enable-arp
```

Configuration with a Remote Peer on the Same Network

An example of a remote peer on the same SMDS network follows. Note that this router does not have DECnet routing enabled.

Example

```
interface Serial 0
ip address 1.1.1.1 255.0.0.0
appletalk address 92.2
appletalk zone smds
clns router igrp FOO
novell net 1a
xns net 17
encapsulation SMDS
! smds configuration follows
smds address c120.1580.4792
no smds d15-mode
smds static-map APPLETALK 92.1 c120.1580.4721
smds static-map APPLETALK 92.3 c120.1580.4593
smds static-map APPLETALK 92.4 c120.1580.4594
smds static-map NOVELL 1a.0c00.0102.23cb c120.1580.4721
smds static-map XNS 17.0c00.0102.23cb c120.1580.4721
smds static-map NOVELL 1a.0c00.0102.23dd c120.1580.4728
smds static-map XNS 17.0c00.0102.23aa c120.1580.4727
smds multicast NOVELL e180.0999.9999
smds multicast XNS e180.0999.9999
smds multicast ARP e180.0999.9999
smds multicast IP e180.0999.9999
smds multicast APPLETALK e180.0999.9999
smds multicast AARP e180.0999.9999
```

```
smds multicast CLNS_IS e180.0999.9990
smds multicast CLNS_ES e180.0999.9990
smds enable-arp
```

Monitoring SMDS Service

Use the following EXEC commands to monitor the SMDS service.

Displaying SMDS Individual Addresses

Use the SHOW SMDS ADDRESSES command to display the individual addresses and the interface that they are associated with. Enter this command at the EXEC prompt:

show smds addresses

A sample display of the command output follows.

```
SMDS address - Serial0   c141.5555.1212
```

Displaying Mapped SMDS Addresses

Use the SHOW SMDS MAP command to display all SMDS addresses that are mapped to higher-level protocol addresses. Enter this command at the EXEC prompt:

show smds map

The display for this command includes all addresses entered with both the SMDS STATIC-MAP and SMDS MULTICAST command.

A sample display of the command output follows.

```
Serial0:  ARP maps to e180.0999.9999 multicast
Serial0:  IP maps to e180.0999.9999 multicast
Serial0:  XNS 1006.AA00.0400.0C55 maps to c141.5688.1212 static
```

Note

Trailing Fs are implied in displays showing the SMDS addresses.

Displaying SMDS Counters

Use the SHOW SMDS TRAFFIC command to display all the SMDS counters. Enter this command at the EXEC prompt:

show smds traffic

A sample display of the command output follows.

```
0 Bad BA size errors
0 Bad Header extension errors
0 Invalid address errors
```

In the display:

- The `Bad BA size errors` field lists the number of corrupted packets received based on the expected Level 3 PDU buffer allocation size.

- The `Bad Header extension errors` field lists the number of invalid packets received in which the header extension length did not match what was expected in the Level 3 PDU.
- The `Invalid address errors` field lists the number of packets passed that were incorrectly sent to router. Both individual and multicast address errors are included in this count.

Debugging SMDS

Use the following EXEC commands to debug the SMDS service. For each DEBUG command, there is a corresponding UNDEBUG command that turns the messages off.

debug arp

Use this command to see whether ARPs are being sent or received. This command prints one line for each ARP sent or received.

debug serial-interface

Use this EXEC command to enable logging of SMDS events. This command prints a one-line message for each packet that is sent. The packet size, packet type, and SMDS source and destination addresses are printed. If a packet is received with an incorrect destination address, it will be noted and counted.

Packet-Switched Software Global Configuration Command Summary

Summaries of the global configuration commands relevant to each of the packet-switching technologies described in this chapter are collected in the listing that follows. These commands are listed in alphabetical order, according to technology type.

X.25 Global Configuration Command Summary

This section provides an alphabetically arranged summary of the X.25 global configuration commands.

x25 bfe-decision {**no** | **yes** | **ask**}

Configures how or if the router will participate in making the decision to enter emergency mode. When the router's mode is set to **decision** (this mode is set using the X25 BFE-EMERGENCY command described next), the type of decision must be indicated by using the X25 BFE-DECISION command. If the decision type is set to **no** (the default value), the router will not participate in the decision to enter or leave emergency mode. If the decision type is set to **yes**, the router will participate in entering emergency mode. If the decision type is set to **ask**, the router will prompt the administrator to use an EXEC command to place the router in and out of emergency mode.

x25 bfe-emergency {**never** | **always** | **decision**}

Configures the circumstances under which the router will enter emergency mode. The default keyword value is **never**. When set to **never**, the router does not go into emergency mode. When set to **always**, the router enters emergency mode when directed by the BFE. When set to **decision** the router waits until it receives a diagnostic packet from the BFE device indicating that an emergency mode window is open before it enters emergency mode.

x25 remote-red *host-ip-address* **remote-black** *Blacker-Internet-address*

Sets up the table that lists the BFE nodes to which the router will send packets. The host-ip-address argument is the IP address of the host or a router that the packets are being sent to. The Blacker-Internet-address argument is the IP address of the remote Blacker front-end unit in front of the host that the packet is being sent to.

[no] x25 route *[# position]x121-pattern* [**cud** *pattern*] **interface** *interface-name*
[no] x25 route *[# position]x121-pattern* [**cud** *pattern*] **ip** *ip-address*
[no] x25 route *[# position]x121-pattern* [**cud** *pattern*] **alias** *interface-name*
[no] x25 route *[#position]x121-pattern* [**substitute-source** *rewrite-pattern*]
[substitute-dest *rewrite-pattern]* [**cud** *pattern*] **interface** *destination-interface*

Inserts or removes an entry in the X.25 routing table. The *x121-pattern pattern* parameter is the X.121 address of the called destination and is required. The alias keyword permits a way for other X.121 addresses to be treated as local. The substitute-source keyword allows substitution of the calling address. The argument *rewrite-pattern* replaces the called or calling X.121 address in routed X.25 packets.

[**no**] **x25 routing**

Enables or disables X.25 switching. X.25 calls will not be forwarded until this command is issued. The command NO X25 ROUTING disables the forwarding of X.25 calls.

Packet-Switched Software Interface Subcommand Summary

Summaries of the interface subcommands relevant to each of the packet-switching technologies described in this chapter are collected in the listing that follows. These commands are listed in alphabetical order, according to technology type. Interface subcommands are summarized for the following:

- CMNS
- Frame Relay
- LAPB
- SMDS
- X.25

CMNS Interface Subcommand Summary

This section provides an alphabetical list of all the interface subcommands used in the X.25 interface.

[**no**] **cmns enable**

Enables CMNS on a nonserial interface. After executing this command, all the X.25-related interface subcommands are made available on the Ethernet interface as well as on serial interfaces. The NO CMNS ENABLE command disables CMNS for the interface.

[**no**] **x25 map cmns** *NSAP MAC-address*
[**no**] **x25 map cmns** *NSAP [X.121-address]*

After enabling CMNS on a nonserial interface (or specifying X.25 encapsulation on a serial interface), you must use the X25 MAP CMNS interface subcommand to map NSAP addresses to either MAC-layer addresses or X.121 addresses, depending on the application.

The *NSAP, MAC-address,* and *X.121-address* arguments specify the NSAP address-to-X.121 address or NSAP address-to-MAC address mappings.

To retract a mapping, use the NO X25 MAP CMNS interface subcommand with the appropriate address arguments.

Frame Relay Interface Subcommand Summary

Following is an alphabetically arranged summary of the frame relay interface subcommands.

encapsulation frame-relay

Specifies frame relay encapsulation on a specific interface.

[**no**] **frame-relay keepalive** *seconds*

Enables and disables the LMI mechanism for serial lines using the frame relay encapsulation.

[**no**] **frame-relay local-dlci** *number*

Sets the source DLCI for use when the LMI is not supported. If LMI is supported and the multicast information element is present, the router sets its local DLCI based on information provided via the LMI. The argument *number* is the local, or source, DLCI number.

[**no**] **frame-relay lmi-type ANSI**

Specifies the exchange of local management interface messages as defined by ANSI standard T1.617.

The **no frame-relay lmi-type ANSI** returns the LMI type to the default as defined by the Cisco/Stratacom/Northern Telecom/Digital specification.

[**no**] **frame-relay map** *protocol protocol-address DLCI* [**broadcast**] [**ietf**]

Defines the mapping between an address and the DLCI used to connect to the address. The frame relay map tells the router how to get from a specific protocol and address pair to the correct DLCI. The argument *protocol* can be one of these keywords: **ip, decnet, appletalk, xns, novell, vines, clns**. The keyword is followed by the corresponding protocol address and the DLCI number. The optional **broadcast** flag specifies that broadcasts should be forwarded to this address when the multicast is not enabled. The default is not to forward broadcasts.

The optional keyword **ietf** is used to select RFC 1294 encapsulation. If you are using the **ietf** option, the argument *protocol* will be ip. Use the IETF form of the map command if your router is connected to another vendor's equipment across a frame relay network.

The **no** version of the command deletes the entry.

[**no**] **frame-relay map bridge** *DLCI* **broadcast**

This variation of the FRAME-RELAY MAP command is used for bridging.

[no] frame-relay map clns *DLCI* **broadcast**

This variation of the FRAME-RELAY MAP command is used for the ISO CLNS protocol.

no frame-relay map

Deletes the frame relay map entry.

[no] frame-relay multicast-dlci *number*

Defines a DLCI to be used for multicasts and should only be used when the multicast facility is not supported. Network transmissions (packets) sent to a multicast DLCI are delivered to all routers defined as members of the multicast group. The argument *number* identifies the multicast group.

[no] frame-relay short-status

Instructs the router to request the short status message from the switch (see Version 2.3 of the joint *Frame Relay Interface* specification). The default is to request the full status message.

LAPB Interface Subcommand Summary

This section provides an alphabetical list of all the interface subcommands used in the LAPB interface.

encapsulation {**lapb** | **lapb-dce**}

Enables running datagrams of a single protocol over a serial interface using the LAPB encapsulation. The keyword **lapb** sets DTE operation; the keyword **lapb-dce** sets DCE operation. One end of the link must be DTE, and the other must be DCE. By default, the single protocol is IP.

encapsulation {**multi-lapb** | **multi-lapb-dce**}

Enables use of multiple network protocols on the same line at the same time.

lapb k *window-size*

Specifies the maximum permissible number of outstanding frames, called the *window size*. The argument *window-size* is a packet count from one to seven. The default value is seven packets.

lapb n1 *bits*

Specifies the maximum number of bits a frame can hold, as indicated by the **n1** keyword. The argument *bits* is the number of bits from 1 through 12,000 and must be a multiple of eight. The default value is 12,000 bits (1500 bytes).

lapb n2 *retries*

Specifies the maximum number of times an acknowledgment frame can be retransmitted. The argument *retries* is the retransmission count from 1 through 255. The default value is 20 retransmissions.

lapb protocol *keyword*

Enables configuration of other protocols over a serial interface using LAPB encapsulation. Possible protocol keywords include **ip, xns, decnet, appletalk, vines, clns, novell,** and **apollo**.

lapb t1 *milliseconds*

Sets the limit for the retransmission timer (the LAPB T1 parameter). The argument *milliseconds* is the number of milliseconds from 1 through 64000. The default value is 3,000 milliseconds.

SMDS Interface Subcommand Summary

This section provides an alphabetically arranged summary of the SMDS interface subcommands. These commands must be preceded by an **interface** command.

arp *ip-address address* **smds**

Allows inclusion of static ARP entries. The argument *ip-address* is the IP address. The argument *address* is the SMDS address. The keyword **smds** appended to the end of this command is required.

encapsulation smds

Enables or disables SMDS on a particular interface. It should precede any SMDS command. This command has no default.

ip route-cache

Configures an SMDS interface for fast switching.

[**no**] **smds address** *smds-address*

Sets or removes the SMDS individual address for a particular interface. The argument *smds-address* is the individual address provided by the SMDS service provider and is protocol independent. This command has no default.

[**no**] **smds d15-mode**

Allows for interoperation with an AT&T SMDS switch that implements the AT&T d15-mode packet structure. When this feature is disabled using the **no smds d15-mode** specification, the router will use AT&T's d11-mode implementation.

[**no**] **smds enable-arp**

Enables or disables ARP processing on a particular interface. The multicast address for ARP must be set before this command is issued. Default is **no smds enable-arp**.

[**no**] **smds multicast** *protocol-type smds-group-address*

Maps an SMDS group address to a broadcast or multicast address used by higher-level protocols. This command has no default.

[**no**] **smds static-map** *protocol-type protocol-address smds-address*

Sets up a static mapping between an SMDS address and a higher-level protocol address. This should not be used for broadcast addresses; for broadcast addresses, use the SMDS MULTICAST command. This command has no default.

X.25 Interface Subcommand Summary

This section provides an alphabetical list of all the interface subcommands used in the X.25 interface.

encapsulation bfex25

This encapsulation provides a mapping from Class A IP addresses to the type of X.121 addresses expected by the BFE encryption device.

encapsulation {**ddnx25** | **ddnx25-dce**}

Causes the router to specify the standard service facility in the Call Request packet, which notifies the PSNs to use standard service.

encapsulation hdh

Enables the HDH protocol.

encapsulation x25

Sets X.25 DTE operation.

encapsulation x25-dce

Sets X.25 DCE operation.

hdh {**packet** | **message**}

Enables router support for both the packet and message modes of HDH. The packet keyword specifies the packet mode; the message keyword specifies the message mode.

[**no**] **x25 accept-reverse**

Instructs the router to accept all reverse-charged calls. This behavior can also be configured on a per-peer basis using the X25 MAP subcommand. The **no** form of the command resets the default state.

x25 address *X.121-address*

Sets the X.121 address of a particular network interface. The address is assigned by the X.25 network. The argument *X.121-address* is a variable-length X.121 address.

[no] x25 default *protocol*

Specifies or removes the protocol assumed by the CPT to interpret calls with unknown Call User Data. The argument protocol sets the default protocol and is either **ip** or **pad**.

x25 facility *keyword argument*
no x25 facility *keyword argument*

Overrides interface settings on a per-call basis.

The command enables X.25 facilities, which are sent between DTE and DCE devices to negotiate certain link parameters.

The argument *keyword* specifies one of the following keywords, followed by the required *argument*:

- **cug** *number*—Specifies a closed user group number from 1 through 99 to provide an extra measure of network security.
- **packetsize** *in-size out-size*—Sets the size in bytes of input packets (*in-size*) and output packets (*out-size*). Both values should be the same.
- **reverse**—Reverses charges on all Call Request packets from the interface.
- **windowsize** *in-size out-size*—Sets the packet count for input windows (*in-size*) and output windows (*out-size*). Both values should be the same.
- **throughput** *in out*—Sets the requested throughput values for input and output throughput across the network.
- **rpoa** *name*—Specifies the list of transit Recognized Private Operating Agencies (RPOAs) to use in outgoing Call Request packets.
- **transit-delay** *number*—-Specifies the transit delay value in milliseconds (0 to 65334) for mapping in outgoing calls for networks that support transit delay.

  The NO X25 FACILITY command with the appropriate keyword and argument removes the facility.

x25 hic *circuit-number*

Sets the highest incoming virtual circuit number. The argument *circuit-number* is the circuit number from 1 through 4095. The default value is 0 (range disabled).

x25 hoc *circuit-number*

Sets the highest outgoing virtual circuit number. The argument *circuit-number* is the circuit number from 1 through 4095. The default value is 0 (range disabled).

[**no**] **x25 hold-queue** *queue-size*

Defines the number of packets to be held until a virtual circuit is established. The argument *queue-size* defines the number of packets. By default, this number is zero. The **no** form without an argument removes this command from the configuration file; the command with a queue size value of zero returns the default.

[**no**] **x25 hold-vc-timer** *minutes*

Prevents overruns on X.25 switches for traffic through the VCs. The argument *minutes* is the number of minutes to prevent calls to a previously failed destination. Incoming calls still will be accepted.

x25 htc *circuit-number*

Sets the highest two-way channel (HTC). The argument *circuit-number* is a circuit number from 1 through 4095. The default value is 1024.

[**no**] **x25 idle** *minutes*

Sets the period of inactivity once an SVC has been cleared. The argument *minutes* is the number of minutes in the period. Calls both originated and terminated by the router are cleared. The default value is 0 (zero), which causes the router to keep the SVC open indefinitely. The **no** variation restores this default.

[**no**] **x25 ip-precedence**

Enables or disables the ability to open a new virtual circuit based on the IP type of service (TOS) field. By default, the DECbrouter 90 opens one virtual circuit for each type of service.

x25 ips *bytes*
x25 ops *bytes*

Set the router packet size to match those of the network. The **ips** keyword specifies the router input packet size, while the keyword **ops** specifies the router output packet size. The argument *bytes* is a byte count in the range of 128 through 1024. The default value is 128 bytes. Larger packet sizes are better than smaller ones because smaller packets require more overhead processing.

Note

Set the **x25 ips** and **x25 ops** keywords to the same value unless your network supports asymmetry between input and output packets.

x25 lic *circuit-number*

Sets the lowest incoming virtual circuit number. The argument *circuit-number* is a circuit number from 1 through 4095. The default value is 0 (range disabled).

[**no**] **x25 linkrestart**

Forces a packet-level restart when the link level is restarted and restarts X.25 Level 2 (LAPB) when errors occur. This behavior is the default and is necessary for networks that expect this behavior. The **no** form of the command turns off the default.

x25 loc *circuit-number*

Sets the lowest outgoing virtual circuit number. The argument *circuit-number* is the circuit number from 1 through 4095. The default value is 0 (range disabled).

x25 ltc *circuit-number*

Sets the lowest two-way channel (LTC). The argument *circuit-number* is the circuit number from 1 through 4095. The default value is one.

x25 map bridge *X.121-address* **broadcast** [*options-keywords*]

Configures support for bridging of X.25 frames. The command specifies Internet-to-X.121 address mapping. The keyword **bridge** specifies bridging over X.25. The argument *X.121-address* is the X.121 address. The keyword **broadcast** is required for bridging X.25 frames. The argument *options-keywords* represents services that can be added to this map.

[**no**] **x25 map compressedtcp** *ip-address x.121-address [options]*

Specifies a network protocol-to-X.121 address mapping such as Internet-to-X.121 or DECnet-to-X.121. Refer to the description of the x25 map interface subcommand for supported protocols. This version is required to make the X.25 calls complete for compressed packets.

The argument *ip-address* is the IP address and x.121-address is the X.121 address. The *options* arguments are the same options as those for the x25 map command.

The Call User Data of compressed TCP calls is the single byte 0xd8.

The NO X25 MAP COMPRESSEDTCP disables TCP header compression for the link.

[**no**] **x25 map** *protocol-keyword protocol-address X.121-address [options]*

Specifies a network protocol-to-X.121 address mapping such as Internet-to-X.121 or DECnet-to-X.121. The argument *protocol-keyword* can be one of these protocol types: **ip, decnet, chaos, xns, novell, appletalk, vines, apollo, pup, clns, bridge, cmns,** and **compressedtcp**. The *address* arguments specify the network protocol-to-X.121 mapping. The *options*

arguments add certain features to the mapping specified and can be any of the following:

- **accept-reverse**—Causes the router to accept incoming reverse-charged calls. If this option is not present, the router clears reverse-charged calls.
- **broadcast**—Causes the router to direct any broadcasts sent through this interface to the specified X.121 address. This option is needed when dynamic routing protocols are being used to access the X.25 network, and is required for bridging X.25.
- **cug** *number*—Specifies a closed user group number (from 1 to 99) for the mapping in the outgoing call.
- **modulo** *size*—Specifies the maximum window size for this map. The argument *size* permits windows of 8 or 128 on the same interface.
- **nuid** *username password*—Specifies that a network ID facility be sent in the outgoing call with the specified user name and password.
- **nvc** *count*—Sets the number of virtual circuits (VCs) for this map/host. The default *count* is the **x25 nvc** setting of the interface. A maximum number of eight VCs can be configured for a single map/host.
- **packetsize** *in-size out-size*—Specifies input packet size (*in-size*) and output packet size (*out-size*) for the mapping in the outgoing call.
- **reverse**—Specifies reverse charging for outgoing calls.
- **rpoa** *name*—Specifies the list of transit Recognized Private Operating Agencies (RPOAs) to use in outgoing Call Request packets for this x25 map entry.
- **throughput** *in out*—Requests the amount of bandwidth through the X.25 network.
- **transit-delay** *number*—Specifies the transit delay value in milliseconds (0 to 65334) for mapping in outgoing calls for networks that support transit delay.
- **windowsize** *in-size out-size*—Specifies input window size (*in-size*) and output window size (*out-size*) for the mapping in the outgoing call.

x25 modulo *modulus*

Sets the modulus. The argument *modulus* is either 8 or 128. The default value is eight. The value of the modulo parameter must agree with that of the device on the other end of the X.25 link.

x25 nvc *count*

Specifies the maximum number of switched virtual circuits that can be open simultaneously to one host. The argument *count* is a circuit count from 1 to 8; the default is 1.

[**no**] **x25 pvc** *circuit protocol-keyword protocol-address [options]*

Establishes or deletes permanent virtual circuits (PVCs). The argument *circuit* is a virtual circuit channel number; it must be less than the lower limit of the incoming call range in the virtual circuit channel sequence (set using the **lic** keyword). The argument *protocol-keyword* can be one of the supported protocol types: **ip, decnet, chaos, xns, novell, appletalk, vines, apollo, pup,** and **bridge**. The argument *protocol-address* is that of the host at the other end of the PVC.

The optional argument *options* can be one of the following:

- **packetsize** in-size out-size—Specifies input packet size (in-size) and output packet size (out-size) for the mapping in the outgoing call.
- **windowsize** in-size out-size—Specifies input window size (in-size) and output window size (out-size) for the mapping in the outgoing call.

Note

You must specify the required network protocol-to-X.121 address mapping with an **x25 map** sub before you can set up a PVC.

x25 pvc *pvc-number* **interface** *interface-name pvc-number [options]*

Configures a PVC for a given interface. The argument *pvc-number* is the PVC number that will be used on the local interface (as defined by the primary interface command). The argument *interface-name* is the interface type and unit number (serial 0, for example), as specified by the **interface** keywords.

The optional argument *options* can be one of the following:

- **packetsize** *in-size out-size*—Specifies input packet size (*in-size*) and output packet size (*out-size*) for the mapping in the outgoing call.
- **windowsize** *in-size out-size*—Specifies input window size (*in-size*) and output window size (*out-size*) for the mapping in the outgoing call.

[**no**] **x25 rpoa** *name number*

Specifies a list of transit RPOAs to use, referenced by name. The argument *name* must be unique with respect to all other RPOA names. It is used in the X25 FACILITY and X25 MAP interface subcommands. The argument *number* is a number that is used to describe an RPOA.

[**no**] **x25 suppress-called-address**

Omits the called address in outgoing calls The **suppress-called-address** keyword omits the called (destination) X.121 address in Call Request packets. This option is required for networks that expect only subaddresses in the called address field. The called address is sent by default. The **no** form resets this subcommand to the default state.

[**no**] **x25 suppress-calling-address**

Omits the calling (source) X.121 address in Call Request packets. This option is required for networks that expect only subaddresses in the calling address field. The calling address is sent by default. The **no** form of the command resets the default state.

x25 t10 *seconds*

Sets the limit for the Restart Request retransmission timer (T10) on DCE devices. The argument *seconds* is a time value in seconds. The default value is 60 seconds.

x25 t11 *seconds*

Sets the limit for the Call Request completion timer (T11) on DCE devices. The argument *seconds* is a time value in seconds. The default value is 180 seconds.

x25 t12 *second*

Sets the limit for the Reset Request retransmission timer (T12) on DCE devices. The argument *seconds* is a time value in seconds. The default value is 60 seconds.

x25 t13 *seconds*

Sets the limit for the Clear Request retransmission timer (T13) on DCE devices. The argument *seconds* is a time value in seconds. The default value is 60 seconds.

x25 t20 *seconds*

Sets the limit for the Restart Request retransmission timer (T20) on DTE devices. The argument *seconds* is a time value in seconds. The default is 180 seconds.

x25 t21 *seconds*

Sets the limit for the Call Request completion timer (T21) on DTE devices. The argument *seconds* is a time value in seconds. The default value is 200 seconds.

x25 t22 *seconds*

Sets the limit for the Reset Request retransmission timer (T22) on DTE devices. The argument *seconds* is a time value in seconds. The default value is 180 seconds.

x25 t23 *seconds*

Sets the limit for the Clear Request retransmission timer (T23) on DTE devices. The argument *seconds* is a time value in seconds. The default value is 180 seconds.

x25 th *delay*

Instructs the router to send acknowledgment packets when it is not busy sending other packets, even if the number of input packets has not reached the **win** count, which improves line responsiveness at the expense of bandwidth. The argument *delay* must be between zero and the input window size. A value of one sends one Receiver Ready acknowledgment per packet at all times. The default value is zero, which disables the delayed acknowledgment strategy.

[**no**] **x25 use-source-address**

Updates the source address of outgoing calls forwarded over a specific interface. The **no** variation prevents the update.

x25 win *packets*
x25 wout *packets*

Set the upper limits on the number of outstanding unacknowledged packets. The **win** keyword specifies the upper limits of the number of outstanding unacknowledged packets in the input window and determines how many packets the router can receive before sending an X.25 acknowledgment.

The **wout** keyword specifies the upper limits of the number of outstanding unacknowledged packets in the output window. The **wout** limit determines how many sent packets can remain unacknowledged before the router uses a hold queue.

To maintain high bandwidth utilization, assign these limits the largest number that the network allows. The argument *packets* is a packets count. The packet count for **win** and **wout** can range from one to the window modulus. The default value is two packets.

Note

Set **win** and **wout** to the same value unless your network supports asymmetry between input and output window sizes.

X.25 EXEC Command Summary

This section provides an alphabetical list of X.25 EXEC commands.

bfe {**enter** | **leave**} *interface-type unit*

Implements Blacker emergency mode on an interface. The keyword options are **enter** and **leave**, indicating whether the interface will go in or out of emergency mode. The *interface-type* argument is the interface name, and *unit* is the interface number. Use this command to enter or leave the emergency mode of operation if your system is in emergency mode **decision** and decision type **ask**.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X