DEC Network Integration Server

Management, Volume 1

Order Number: AA-R84QA-TE

Revision/Update Information: This is a revised manual

Software Version: DECNIS™ V4.1

First Printing, March 1992
Revised, September 1997

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

2 Using a Console Terminal on the DECNIS

3 Configuring Nonvolatile Memory Dynamically

4 Configuring SNMP on the DECNIS

5 Setting Up Event Logging on the DECNIS

6 System-Level Management

Part II Loading

7 Loading a DECNIS

8 Nonvolatile (Flash) Memory Loading

9 Using the DECNIS as a Proxy Load Host

10 Using the DECNIS as a BOOTP Gateway

Index

Figures

Tables

Preface

This manual describes how to manage those aspects of the DEC™ Network Integration Server that are not protocol specific. It describes the management model, basic configuration, loading, the use of the console, flash memory, event logging and security.

The DEC Network Integration Server is referred to throughout this manual as the DECNIS.

The manual *DECNIS Management, Volume 2* describes how to manage the data link, routing, bridging and X.25 functions of the DECNIS.

Audience

This manual is intended for network managers. It assumes that you understand and have some experience of:

- Local Area Networks (LANs)
- Wide Area Networks (WANs)
- OpenVMS™ (if using an OpenVMS load or management host)
- DIGITAL™ UNIX® (if using a DIGITAL UNIX load or management host).
- IBM®-compatible Personal Computers running Windows 95® or Windows NT™ (if using either of these types of system as a load or management host).

Associated Documentation

Product Documentation

- *DEC Network Integration Server Management, Volume 2*
- *DEC Network Integration Server Introduction and Glossary*
- *DEC Network Integration Server Installation and Configuration for OpenVMS and DIGITAL UNIX*

- *DTF (DIGITAL Trace Facility) User Guide*
- *clearVISN™ DECNIS Configurator User Guide*
- *DEC Network Integration Server Problem Solving*

  This is only available on line, as follows:

  - On OpenVMS or DIGITAL UNIX systems, in Bookreader™ format.
  - On Windows 95/NT PCs, as a Windows® help file.
- *DECNIS Event Messages* (supplied on line as a text file)
- *DECNIS Release Notes* (supplied on line as a text file)

Hardware Documentation

The following documents are supplied with the DECNIS hardware:

- *Installation and Service Manual*
- *Configuration Card*

The following documents are supplied with each Network Interface Card:

- *Cabling Instructions and Specifications* card
- *Problem Solving* card
- *Configuration* card

Related Documentation

- NCL online help

  This describes the NCL commands that you can use to manage the DECNIS.
- Network management documentation for the load-host operating system you are using.
- *Common Trace Facility (CTF) Use* manual

  This manual is part of the OpenVMS documentation set, and describes how to use the Common Trace Facility for problem solving.
- *Network Information* (supplied on line)

  This supplies profile information about all the public Packet Switching Data Networks that DIGITAL supports.
- *X.25 Security* manual

  This manual explains the underlying concepts of X.25 security. You can order this manual through your local DIGITAL office.

- *Bridge and Extended LAN Reference* manual

  This manual provides a general description of bridging and extended LANs. You can order this manual through your local DIGITAL office.

- RFCs (for IP routing)

  RFCs are the working notes for the internet research and development community. These notes are available in a three-volume set, the *DDN Protocol Handbook*, which can be ordered from the following address:

  Network Solutions, Inc.
  Attn: InterNIC Registration Service
  505 Huntmar Park Drive
  Herndon, VA 22070, USA
  Tel. 1-800-444-4345 or 619-455-4600

Returning Comments About this Documentation

We would like to know what you think about the DECNIS documentation set and online help.

If you have any comments, or suggestions, please return them in any of the following ways:

- Send an electronic mail message to the Internet address books@reo.mts.dec.com
- Send an electronic mail message to the X.400 address S=IDC BOOKS; O=digital; OU1=reo; P=digital; A=CWMail; C=gb
- Send a fax to (+44)118 9206018

Conventions

The following conventions are used in this manual:

Italics	This indicates variable information.
decnis	This indicates that you should substitute the node name of the DECNIS. If you are using DECdns or the local namespace, enter the registered name.
DECNIS	DEC Network Integration Server
PC	An IBM-compatible personal computer

Prompts

The following prompts precede commands that you enter:

For OpenVMS: `$`
For DIGITAL UNIX: #
For the DECNIS console: `console>`
For NCL: `NCL>`

Part I

Managing the System

This part contains information on managing those aspects of the DECNIS that are not protocol specific.

It contains the following chapters:

- Chapter 1 describes the DECNIS management model and the tools you can use to manage the DECNIS.
- Chapter 2 describes how to use the DECNIS console.
- Chapter 3 describes how to modify the contents of DECNIS nonvolatile (flash) memory on the MPC-II and MPC-III.
- Chapter 4 describes how to set up SNMP to manage the DECNIS.
- Chapter 5 explains how to set up event logging from the DECNIS.
- Chapter 6 describes system-level management.

1

Tools for Managing the DECNIS

1.1 Introduction: SNMP and DEC CMIP

1.1.1 Two Management Protocols

The DECNIS supports two management protocols:

- SNMP (Simple Network Management Protocol)
- DEC CMIP (Common Management Information Protocol)

1.1.2 SNMP (Simple Network Management Protocol)

SNMP is an Internet standard protocol, specified in RFC 1157.

You can use a network management station running SNMP to monitor and change a range of the functions of the DECNIS.

You can only use SNMP to manage a running DECNIS: any changes you make to its configuration will be lost if the DECNIS is rebooted.

Chapter 4 describes how to configure the DECNIS to be manageable using SNMP.

1.1.3 DEC CMIP (Common Management Information Protocol)

DEC CMIP is a DIGITAL-proprietary management protocol. It is used to monitor and change all DECNIS functions, either dynamically or statically (see Section 1.3.1.1 and Section 1.3.1.2).

Section 1.2 to Section 1.13 describe the CMIP management model, and the tools that use CMIP.

1.2 Management Modules

All the management information for the DECNIS, including configuration details, is held in a series of modules, each representing a functional part of the system.

1.2.1 Entities

A module usually contains one or more entities, each dealing with a part of that module's function.

1.2.2 Entity Attributes

Each entity has one or more attributes, the values of which determine the behavior of the relevant DECNIS function.

1.2.3 Example

The Routing module represents the routing functions of the DECNIS. It contains the entities shown in Figure 1–1.

Each ROUTING CIRCUIT entity on the DECNIS has attributes that determine how the corresponding routing circuit behaves.

For example, the value of the HELLO TIMER attribute of a ROUTING CIRCUIT entity called CIRCUIT_1 determines how often the DECNIS will generate router Hello messages on the routing circuit CIRCUIT_1.

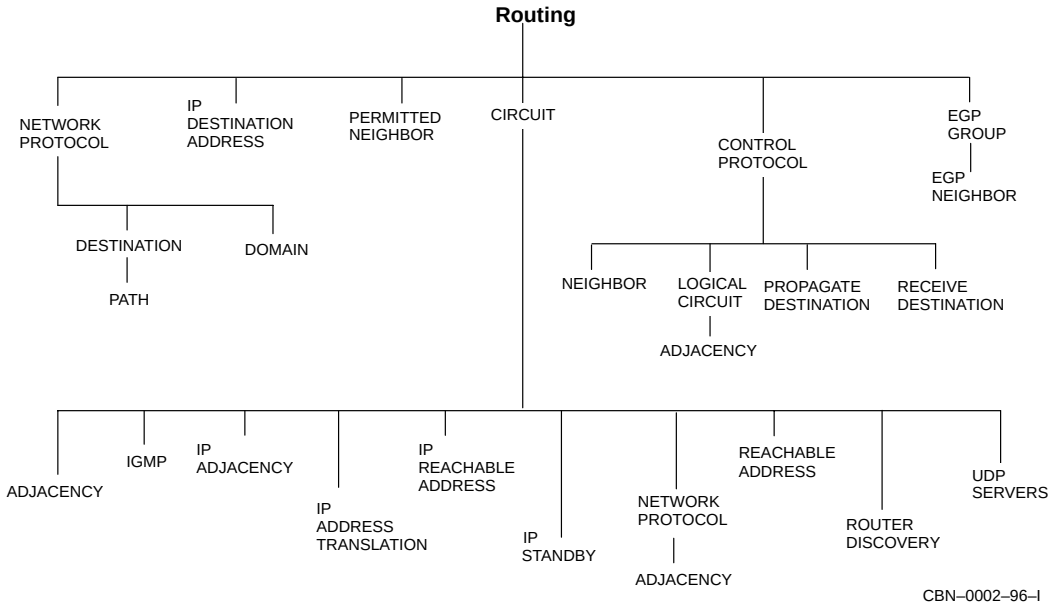
1.2.4 Entities and Management

To manage the DECNIS, you use the management tools described in the remainder of this chapter. These tools manipulate the management modules and entities, and the configuration information they contain.

Use the management tools described to:

- Set up new modules and/or entities.
- Delete existing modules and/or entities.
- Change the characteristics of existing entities.

Figure 1–1 Routing Module



1.2.5 Entities Implemented in the DECNIS

Figure 1–2 shows the management modules implemented in the DECNIS.

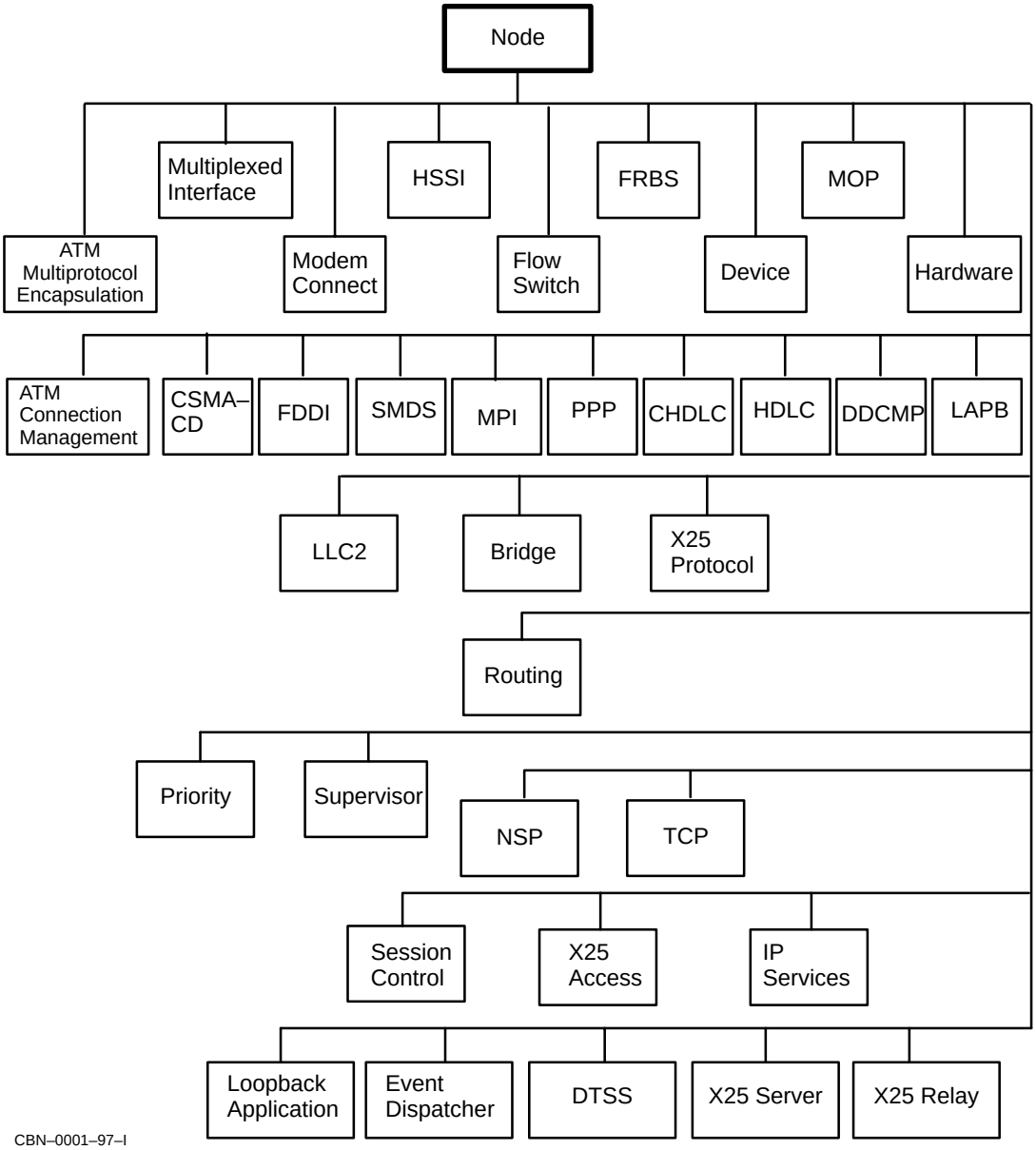
Appendix A describes all the modules and associated entities used by the DECNIS.

Appendix B lists the entities' characteristics, as implemented in the DECNIS.

Figure 16–1, Figure 16–2, Figure 16–3, Figure 16–4, and Figure 16–5 show the entities associated with a routing circuit, and how they are related.

Figure 27–2, Figure 27–3, Figure 27–4, and Figure 27–5 show the entities associated with a bridge port, and how they are related.

Figure 1–2 Management Modules Implemented in the DECNIS



1.3 Using NCL

1.3.1 What Is NCL?

The Network Control Language (NCL) is a utility that enables you to configure, manage and monitor your DECNIS on the network. You issue NCL commands to manipulate the DECNIS modules described in Appendix A.

The network management documentation for OpenVMS and DIGITAL UNIX provides information on most of the management modules described in Appendix A. The NCL online help for the DECNIS provides more detailed information on some of these modules, and information on the additional modules not covered there.

Note that on Windows 95/NT load hosts, NCL help is supplied in a Windows help file. You can refer to this help within the clearVISN DECNIS configurator, or by opening the Windows help file, **NCL Command Help**. Refer to the manual *clearVISN DECNIS Configurator User Guide* for more information.

You can issue NCL commands either **dynamically** or **statically**.

1.3.1.1 Dynamic Management: Definition

Dynamic management means making changes to the configuration of the DECNIS while it is running. To do dynamic management, you issue NCL commands to a running DECNIS. The DECNIS responds immediately to the NCL commands you issue.

When the DECNIS is rebooted, the configuration changes made dynamically are lost.

1.3.1.2 Static Management: Definition

Static management means changing the permanent configuration of the DECNIS. These changes do not take effect until the DECNIS is rebooted.

The permanent configuration consists of the management information that is loaded to the DECNIS. This is the binary version of the **NCL script files**.

The NCL script files are text files containing NCL commands. They are stored on the load host.

Refer to Section 1.5.2 and Section 1.5.3 for more information about the permanent configuration.

1.4 Using Dynamic Management

This section describes the types of system from which you can do dynamic NCL management, and gives general information on issuing NCL commands.

1.4.1 Management Systems

You can issue NCL commands from the following types of system:

- A supported management host. Refer to Section 1.4.1.1.
- The DECNIS console. Refer to Section 2.13.1.

1.4.1.1 Supported Management Hosts

Management hosts can be any of the following:

- OpenVMS or DIGITAL UNIX systems running DECnet-Plus (formerly known as DECnet/OSI).
- DIGITAL UNIX systems using Transmission Control Protocol (TCP) as the transport protocol. Refer to Section 1.10.
- IBM-compatible Personal Computers (PCs) running Windows 95 or Windows NT.

1.4.1.2 General Requirements for Management Hosts

Supported management hosts must meet the following requirements:

- The DECNIS software must be installed on the host.
- The DECNIS must be reachable from the host over the network.

A management host is typically a load host, but is not required to be.

1.4.2 Starting NCL and Issuing Commands

To issue NCL commands to manage the DECNIS, follow these steps:

1. Log on to a suitable management host or the DECNIS console.
2. Start NCL on the system, as follows:

To start NCL on...	Refer to...
OpenVMS and DIGITAL UNIX systems	Network management documentation for the host
Windows 95 or Windows NT PCs	*clearVISN DECNIS Configurator User Guide*
The DECNIS console	Section 2.13 in this manual

3. Issue the required NCL commands.

1.4.3 Specifying the DECNIS in NCL Commands

When entering NCL commands, you need to identify the DECNIS node to which the command will apply. You can include the node name in the command, or set up the DECNIS as a default node, as described in Section 1.4.4.

The node name itself can be specified in different ways, depending on the type of management host from which you are issuing commands, and the type of network being used to transport the commands to the DECNIS. Table 1–1 shows the different types of node specification. The **Example Syntax** column gives example commands; however, note that it does not show all possible variations in syntax.

Table 1–1 Node Specifications in NCL commands

Management Host	Node Specifications	Example Syntax
DECnet-Plus host	A DECdns or Local Name Service name	`SHOW NODE ORG:.SOUTH.SALES`
	DECnet Phase IV node name	`SHOW NODE nis100`
Windows 95/NT PC DIGITAL UNIX host	IP address	`SHOW NODE 16.36.16.100`
	IP Domain Name Service node name	`SHOW NODE nis100`

1.4.4 Using a Default Node for NCL Commands

If you wish to issue several NCL commands to the same DECNIS from a management host, set default to the remote system. This section describes how to do this on supported load hosts.

1.4.4.1 DIGITAL UNIX or OpenVMS Hosts

- On DIGITAL UNIX hosts, enter the following command:

  `NCL> SET NCL DEFAULT ENTITY NODE` *`decnis/username/password`*

- On OpenVMS hosts, enter the following command:

  `NCL> SET NCL DEFAULT ENTITY NODE` *`decnis`*`"`*`username password`*`"`

where *decnis* is the node name of the DECNIS, and *username* and *password* are the network management user name and password. Section 6.2.2 describes this user name and password.

1.4.4.2 Windows 95 or Windows NT Hosts

On Windows 95/NT hosts, follow these steps:

1. Start the clearVISN DECNIS configurator.
2. On the menu bar, go to the **Tools** menu and select **Start NCL**.
3. On the NCL window, click the **Defaults** button.
4. Check the box next to the Node Name or Address field, and then enter the node name or address.
5. If you have previously set up a network management user name and password, check the boxes next to the User Name and Password fields. (You set up a network management user name and password on the Security tab page under the **System** button.)
6. Enter the same user name and password that you entered for the network management user name and password.
7. Click **OK**.

1.5 Using Static Management

This section describes the tasks needed to carry out static management on the DECNIS. These tasks are:

1. Create an NCL script for the DECNIS.
2. Create a CMIP file.
3. If you wish, create a combined file.
4. Load the CMIP file and image, or the combined file, to the DECNIS.

1.5.1 Systems Used for Configuration and Loading

The system used for DECNIS configuration and loading is referred to as a load host. A load host can be any of the following types of system:

- OpenVMS or DIGITAL UNIX systems running DECnet-Plus.
- IBM-compatible PCs running Windows 95 or Windows NT.

1.5.2 NCL Scripts

An NCL script is an ASCII file containing NCL commands. The NCL script file resides on the load host for your DECNIS.

1.5.2.1 Creating NCL Scripts

There are two ways to create NCL scripts for the DECNIS:

- Using the DECNIS text-based configurator or the clearVISN™ DECNIS configurator. Each of these utilities generates a master NCL script from information you enter about your DECNIS configuration. This is the recommended method. Refer to Section 1.7 for more information about the configurators.
- Entering commands directly in an NCL script file. This is only recommended for the **user NCL script files** (also called **extra** script files). The purpose of the user NCL script files is to allow you to configure features that are not supported in the configurators. Refer to Section 1.5.6 for details.

1.5.3 CMIP Files and Combined Files

You must convert the NCL script(s) to a **CMIP file** before you can load it to the DECNIS. You can, if you wish, combine the CMIP file, the system image and the profile files into a single file, called the **combined file**.

CMIP File: Definition

A CMIP file is a binary version of the NCL script(s). It is sometimes referred to as a configuration file. It can be loaded separately, or as part of a combined file. Section 1.5.4 describes how to create a CMIP file.

Combined File: Definition

A combined file is a single file that contains the system image, CMIP file and profile files. It is sometimes referred to as the image/CMIP/profile file. Section 1.5.5 describes how to create a combined file.

If you create a combined file, it is always loaded into flash memory.

Note that you can modify the combined file before you load it; refer to Section 1.9.

Profile Files: Definition

These files are used to set various synchronous line and X.25 timers to their optimum settings. For the names of the profile files, see the DECNIS installation manual for your load host.

System Image: Definition

The system image is the software for the DECNIS. It is always loaded into flash memory, whether it is loaded separately or as part of the combined file.

The system image has the following name:

- On OpenVMS or DIGITAL UNIX load hosts:

  NIS041.SYS

- On Windows 95/NT load hosts:

  install-directory\COMMON\NISV41\NISV41.SYS

The DECNIS system image is a double image, containing two internal images:

- An image that only supports MPC–I features.
- An image that supports all features.

The DECNIS only loads one of the internal images into nonvolatile memory. Which internal image is loaded depends on which management processor card is installed.

For more information, refer to the DECNIS installation manual for your load host.

1.5.3.1 Dynamically Updating Flash Memory

Once you have loaded the DECNIS, you can use DECNIS console commands to add one or more CMIP files to flash memory. You can do this regardless of whether you have loaded a combined file or a separate CMIP file and system image. Refer to Chapter 3 for details.

1.5.4 Creating a CMIP File

This section describes how to convert an NCL script to a CMIP file.

1.5.4.1 OpenVMS or DIGITAL UNIX Hosts

On these hosts, you can convert the NCL script files to a CMIP file within the DECNIS text-based configurator. Alternatively, use the following commands:

- OpenVMS hosts:

  ```
  $ @SYS$MANAGER:NIS$SCRIPT_COMPILE NCL-script
  ```

- DIGITAL UNIX hosts:

  ```
  # /usr/lib/dnet/nis_script_compile NCL-script
  ```

where *NCL-script* is the name of the NCL script.

1.5.4.2 Windows 95 or Windows NT Hosts

On Windows 95/NT hosts, you can convert the NCL script file to a CMIP file within the clearVISN DECNIS configurator. Refer to Section 1.7.4.

1.5.5 Creating a Combined File

This section describes how to create a combined file.

1.5.5.1 OpenVMS or DIGITAL UNIX Hosts

On these hosts, you can create a combined file within the DECNIS text-based configurator. Alternatively, create a combined file by following these steps:

1. Create a CMIP file, as described in Section 1.5.4.
2. Combine the CMIP file and the image and profile files by entering the following command:

   - OpenVMS load hosts:

     ```
     $ @SYS$MANAGER:NIS$IMAGE_COMPRESS.COM NIS041 client-name
     ```

   - DIGITAL UNIX load hosts:

     ```
     # /usr/lib/dnet/nis_combine nis041 client-name
     ```

   where *client-name* is the load client name of the DECNIS.

1.5.5.2 Windows 95 or Windows NT Hosts

On Windows 95/NT hosts, you create the combined file within the clearVISN DECNIS configurator. Refer to Section 1.7.4.

1.5.6 User NCL Script Files

User NCL files are extra NCL script files that you use to enter NCL commands in addition to those generated by the configurators.

1.5.6.1 Purpose of the User NCL Script Files

The purpose of the user NCL script files is to allow you to change your DECNIS configuration without editing the master NCL script file. Edit the user NCL script files if you want to:

- Change default information that you cannot change from within the configurator, for example, timer values.
- Set up facilities that you cannot set up within the configurator, for example, setting up the DECNIS as a CONS LAN/WAN Relay.

The user NCL script files are generated by the configurator the first time a DECNIS is configured. When they are first generated, they are empty files. The configurator compiles the user NCL script files and the master NCL script to create a CMIP file.

Note

You are advised not to change the permanent configuration of the DECNIS by editing the master NCL script. Any changes you make to the master NCL script will be lost when you next run the configurator.

Note that it is not possible to edit the master NCL script generated by the clearVISN DECNIS configurator.

1.5.6.2 Types of User NCL Script File

The user NCL script files are described below. See the Appendix G for the file specifications on each type of load host.

User NCL Script Files	Purpose
CREATE user NCL script	Insert NCL CREATE commands to create new entities.
SET user NCL script	Insert NCL commands to change the values of an entity's characteristic attributes, such as SET, ADD, REMOVE, and BLOCK.
ENABLE user NCL script	Add NCL ENABLE commands, to enable the entities created by commands in the CREATE user NCL script.

1.5.6.3 Editing User NCL Script Files in the clearVISN DECNIS Configurator

You can edit the user NCL script files within the clearVISN DECNIS configurator, as described in the manual *clearVISN DECNIS Configurator User Guide*.

You cannot edit the user NCL script files within the DECNIS text-based configurator.

1.5.6.4 How the DECNIS Uses the User NCL Script Files

The user NCL script files are invoked when you convert the master NCL script to a CMIP file, as shown in Figure 1–3.

1.5.6.5 Example

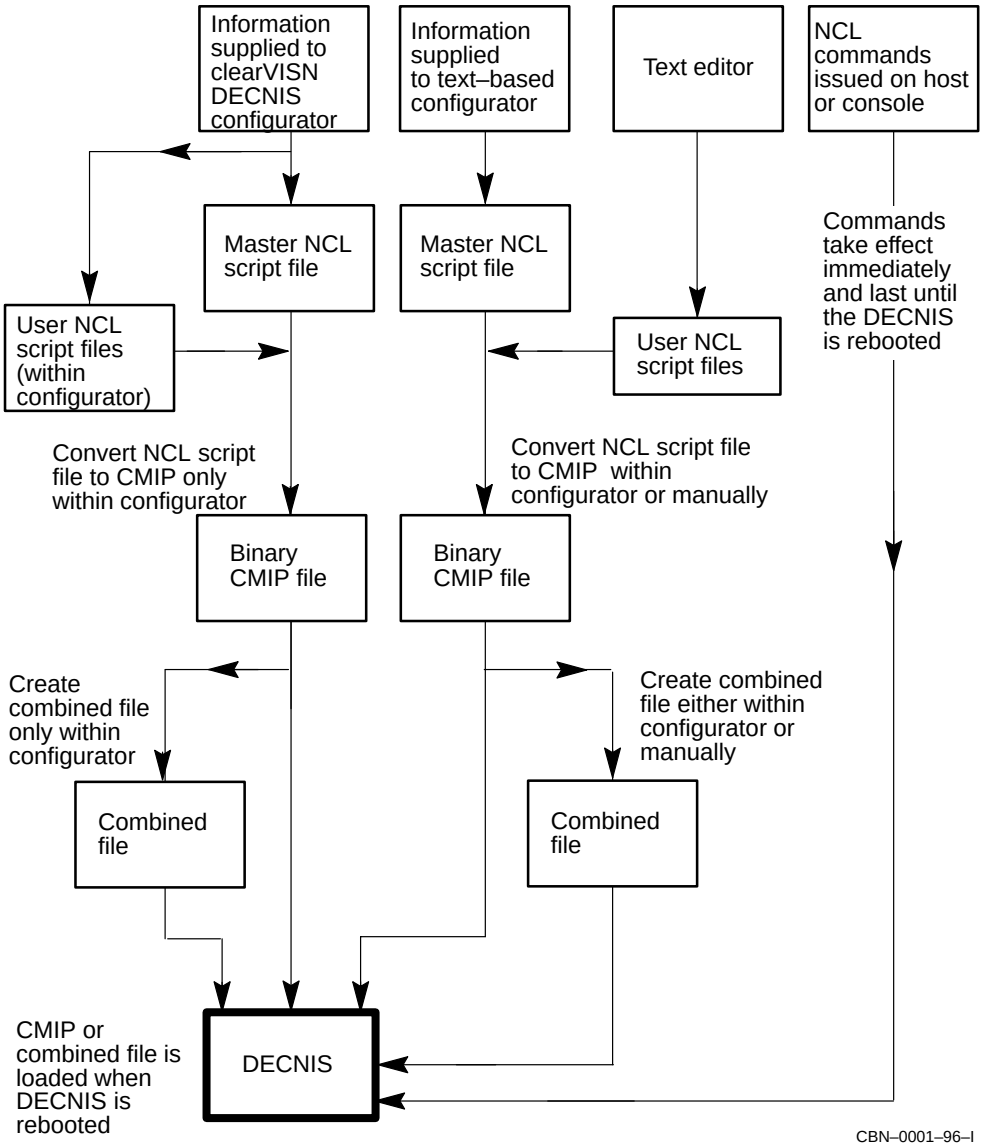
To change the value for the acknowledge timer on an HDLC data link, follow these steps:

1. Find the name of the HDLC link, by examining the master NCL script.
2. Find the command required to change the value of the acknowledge timer:

   ```
   NCL> SET HDLC LINK link-name ACKNOWLEDGE TIMER n
   ```

3. Insert this command in the SET user NCL script.
4. Create a configuration load file; refer to Section 1.5.3 for details.
5. Reboot the DECNIS.

Figure 1–3 Managing the DECNIS Using NCL and the DECNIS Configurators



1.6 Logging Errors During CMIP Compilation

When you create a CMIP file, the CMIP compiler checks the NCL script for errors. If there are any errors, they are written to a log file.

The type of log file varies according to the type of load host:

- On OpenVMS VAX load hosts, the log file contains both syntax errors and semantic errors. If there are errors of either type, no CMIP file is created.
- On OpenVMS Alpha and DIGITAL UNIX load hosts, the log file only contains syntax errors. If there are syntax errors, no CMIP file is created.

  However, once you have created a CMIP file, a separate **NCL checking** utility can be run. This checks the NCL script again, and produces a log file containing any semantic errors.

  Thus, on these load hosts, there are two log files: one with syntax errors and one with semantic errors.

The term **syntax errors** refers to mistakes in the format of individual NCL commands, for example, spelling mistakes.

The term **semantic errors** refers to mistakes in the script as a whole, for example, incompatible commands or missing commands.

1.6.1 CMIP Error Log Files

Table 1–2 shows the names of the log files produced by CMIP compilation.

Table 1–2 CMIP Error Log Files

Load Host	CMIP Log File	Error Type
DIGITAL UNIX	/usr/lib/dnet/nis_*client*.log	Syntax
OpenVMS Alpha	SYS$COMMON:[MOM$SYSTEM]NIS_*client*.LOG	Syntax
OpenVMS VAX	SYS$COMMON:[MOM$SYSTEM]NIS_*client*.LOG	Syntax and semantic

where *client* is the load client name of the DECNIS.

1.6.2 The NCL Checking Utility in the Configurator

On DIGITAL UNIX load hosts, the configurator automatically runs the NCL checking utility after it has created a CMIP file.

When it runs the NCL checking utility, the configurator produces the following log file containing semantic errors:

/usr/lib/dnet/nis_*client*.lis

1.6.3 Running the NCL Checking Utility

You can run the NCL checking utility from the command line on OpenVMS Alpha, and DIGITAL UNIX load hosts. Note that you supply your own name for the log file.

1.6.3.1 Procedure

To run the NCL checking utility, enter the appropriate command as shown in the table:

On this load host...	Enter this command...
OpenVMS Alpha	NIS$NCHK :== $ SYS$SYSTEM:NIS$NCHK.EXE NIS$NCHK *ncl-script log-file*
DIGITAL UNIX	/usr/lib/dnet/nis_nchk *ncl-script log-file*

where:

install-dir	is the directory where the DECNIS was installed.
ncl-script	is the name of the NCL script to be checked.
log-file	is the name you want to use for the log file.

1.6.3.2 Result

When you run the NCL checking utility, any errors and warnings are displayed on the screen. In addition, all NCL script directives, together with errors and warnings, are written to the named log file.

1.6.3.3 Special Requirement

The NCL checking utility does not allow any words in NCL commands to be abbreviated to fewer than three characters. If any words have fewer than three characters, the NCL checking utility issues an error message.

1.6.3.4 Example Log File

The log file produced by the NCL checking utility lists the NCL script directives in a format similar to the following:

```
! script directive 1
create modem connect

! script directive 2
create csma-cd

! script directive 3
create fddi

! script directive 4
create mop
```

When an error is logged, it is in a format similar to the following:

```
! script directive 68
set routing circuit w622-0 manual data link sdu size 4492
! ERROR - set on entity not created circuit w6229-0 manual data link sdu size 4492
```

1.6.4 CMIP Errors Logged during Loading

If there are any errors in the NCL script which have not been corrected before you attempt to load the DECNIS, they are displayed on the DECNIS console during loading.

See Section 7.5.1 for more information about errors logged on the console.

1.6.5 Location of Load Files

Refer to Appendix G for the location on all load hosts of the files loaded to the DECNIS.

1.7 Using the Configurators

1.7.1 Introduction

The following configuration utilities are provided to help you configure the DECNIS software:

Configuration Utility	Description	Supported Hosts
Load-host configurator	A text-based, menu-driven program, running on a VT220 terminal, used to provide information needed for loading. It is used in conjunction with the DECNIS text-based configurator.	DECnet-Plus for OpenVMS Alpha DECnet-Plus for OpenVMS VAX DIGITAL UNIX
DECNIS text-based configurator	A text-based menu-driven program, running on a VT220 terminal, used to provide information about the DECNIS configuration. It is used in conjunction with the load-host configurator.	DECnet-Plus for OpenVMS Alpha DECnet-Plus for OpenVMS VAX DIGITAL UNIX
clearVISN DECNIS configurator	A Windows graphical user interface (GUI) program that combines the functions of the load-host configurator and the DECNIS text-based configurator.	Windows 95/NT PCs

Note that the DECNIS text-based configurator and the clearVISN DECNIS configurator do not support identical functions and facilities. Refer to the manuals *DECNIS Installation and Configuration for OpenVMS and DIGITAL UNIX* and *clearVISN DECNIS Configurator User Guide* for details.

1.7.2 DECNIS Configurators Description

The DECNIS text-based configurator and the clearVISN DECNIS configurator enable you to supply information to define your DECNIS configuration. Based on this information, the configurators can create:

- A master NCL script.
- A CMIP file or combined file which can be loaded to the DECNIS.

1.7.3 Modifying a Configuration: DECNIS Text-Based Configurator

Follow these steps to modify a DECNIS configuration within the DECNIS text-based configurator:

1. Specify Modify from the Main Menu of the DECNIS text-based configurator, and select the name of the DECNIS to be modified.

   Result: The DECNIS will read in the existing configuration information for this DECNIS from a data file. You cannot modify this data file manually.

2. Select a section and enter additional information (for example, add a routing circuit), or change existing information (for example, change the packet size for an X.25 DTE).
3. When you have finished modifying the configuration, select Continue to New Section from any Options menu.

   Result: You will see the NCL Script screen.
4. Select Create an NCL Script.

   Result: The next screen provides the following choices:

   - Create a CMIP file or a combined file from the NCL script

     Select this if you do not want to edit the user NCL script files (see Section 1.5.6). When the DECNIS text-based configurator has created either a separate CMIP file or a combined file, you will return to the Main Menu.
   - Return to Sections Menu

     Select this if you want to check information you provided previously, or change some of your answers and create another NCL script.
   - Return to Main Menu

     Select this if you want to create or modify a configuration for another DECNIS.
   - Exit From the configurator

     Select this if you want to edit the user NCL script files. After you have edited the user NCL script files, you can enter commands to compile the master NCL script into a CMIP file and (if desired) create a combined file. Refer to Section 1.5.4 or Section 1.5.5.

1.7.4 Modifying a Configuration: clearVISN DECNIS Configurator

Follow these steps to modify a DECNIS configuration within the clearVISN DECNIS configurator:

1. Select an existing DECNIS configuration from the Browser.
2. On the Main Navigation window, click the button that takes you to the tab pages you want to change.
3. Make your changes on the tab pages, and click OK.
4. Repeat steps 2 and 3 until you have completed your changes.

5. If you want to edit the user NCL script files (see Section 1.5.6), click the **NCL Scripts** button on the Main Navigation window. You can then add NCL commands on the Extra tab pages. Note that you cannot edit the NCL commands on the Generated NCL tab page.
6. On the Main Navigation window, click the **System** button.
7. On the Load Options tab page, do the following:
   - Select a Load method under Image Loads: either **From Flash** or **From Network**.

     If you want the configurator to create a combined image/CMIP/profile file, check the **CMIP Script** box under Flash Contents.
   - Click **OK**.
   - On the Main Navigation Window, click the **Compile** button.

1.8 Security for NCL Commands

You can set up access control information for the use of NCL by creating a network management username and password.

Section 1.8.1 gives examples of the format used to enter access control information in NCL commands from various hosts.

Once you have created this username and password, you must supply them in any NCL commands you issue for the DECNIS from a host, except for SHOW commands. In addition, if you have not set up a console password, you must supply the network management password when you use Telnet to connect to the DECNIS console.

1.8.1 Example: Supplying Access Control Information in NCL Commands

You want to create a CSMA/CD circuit called SALT on the DECNIS node ORG:.SOUTH.SALES, which has a username SMITH and password SECRET. This node has the DECnet Phase IV address 56.45. Enter the following command:

- OpenVMS hosts:

```
NCL> CREATE NODE org:.south.sales"smith secret" ROUTING CIRCUIT -
_NCL> salt TYPE CSMA-CD
```

- Windows 95/NT PCs:

```
ncl> create node 26.8.8.8/smith/secret routing circuit -
_ncl> salt type csma-cd
```

- DIGITAL UNIX hosts:

```
ncl> create node org:.south.sales/smith/secret routing circuit -
_ncl> salt type csma-cd
```

1.8.2 Setting Default Security for NCL Commands

If you have set up a DECNIS as your default node, as described in Section 1.4.4, you can set default security for NCL on that DECNIS. Once you have set default to the DECNIS, issue the following command:

```
NCL> SET NCL DEFAULT ACCESS BY USER username, PASSWORD password
```

1.8.3 Setting Up NCL Access Control

Refer to Section 6.2 for more information about setting up NCL access control. Refer to Section 2.13 for information about setting up access control for console NCL.

1.9 Modifying the Combined File

MOD_FLSH is a utility that enables you to do the following on a load host:

- Add files to the combined file before it is loaded.
- Delete files from the combined file (or the double system image) before it is loaded.
- Display the contents of the combined file or system image.

For more information about the combined file, refer to Section 1.5.3. The section System Image: Definition in this chapter briefly describes the double system image; for more detailed information, refer to the DECNIS installation and configuration manuals for your load host.

1.9.1 Before You Start

Before you run MOD_FLSH, create a combined file, as described in Section 1.5.5.

1.9.2 Starting MOD_FLSH on OpenVMS or DIGITAL UNIX Hosts

To start MOD_FLSH, enter the following command(s) at the prompt:

- OpenVMS load hosts:

```
$ MOD_FLSH :== $ SYS$SYSTEM:MOD_FLSH.EXE
$ MOD_FLSH combined-file
```

- DIGITAL UNIX load hosts:

  ```
  # /usr/lib/dnet/mod_flsh combined-file
  ```

where *combined-file* is the file specification of the combined file.

1.9.3 Starting MOD_FLSH on Windows 95/NT Hosts

On a Windows 95/NT PC, you can start MOD_FLSH either from the clearVISN DECNIS configurator or from an MS–DOS window.

Starting from the clearVISN DECNIS configurator

There are two ways to start MOD_FLSH from the clearVISN DECNIS configurator:

- From the **Tools Menu**:

  1. Open a configuration.
  2. On the Menu Bar, go to the **Tools** menu, and select **Start mod_flsh**.

- After you have compiled:

  1. When you have finished configuring a DECNIS, click the **Compile** button on the Main Navigation window. The configurator will display a status window.
  2. When the configurator has finished compiling the NCL script, click the **Modify Image** button on the status window. The configurator will start MOD_FLSH.

Starting from an MS–DOS Window

To start MOD_FLSH outside the configurator, follow these steps:

1. Open an MS–DOS window.
2. At the prompt, enter the following command:

   ```
   C:\install-directory\MOD_FLSH combined-file
   ```

where *combined-file* is the file specification of the combined file, and *install-directory* is the installation directory.

1.9.4 MOD_FLSH Display

When you start MOD_FLSH, it displays information about the combined file or system image, including a list of its files. You can then enter commands at the prompt. For example:

```
   Flash Directory of file "nis1.sys"
Image Major/Minor ID 02.05
Image Name "CO_GE_SYS_ROU"
Identification "DECNIS V4.0  "
Link Date/Time  13 October 96 09:20
Flash Pages 8374, Loaded Pages 8399
V2.5/V3.0 format with two system images
Built by flash_proc 4.0

Contents of Flash Structure:

   Index     Size       Date   Time     Name
     1     3335112   29 Aug 96 15:16   nis_v4.0B
     2     1944520   29 Aug 96 13:11   nis_v4.0
     3       14460   19 Sep 96 15:24   script
     3       12210   30 Sep 96 15:24   script.test
     4        1028   19 Aug 96 14:10   mcnm_prf
     5       27242   19 Aug 96 14:10   X25l2_prf
     6       76072   19 Aug 96 14:10   X25l3_prf
Command (h for help) >
```

1.9.5 Exiting from MOD_FLSH

To exit from MOD_FLSH, enter `Quit` at the command prompt.

1.9.6 Getting Help

To display the list of MOD_FLSH commands, press h and then press Return at the prompt.

1.9.7 Adding a File

To add a file to the combined file, enter the following command:

`add` *`file-spec flash-name`*

where: *file-spec* is the specification of the file to be added

flash-name is an optional internal name for the file being added

It is strongly recommended that you supply the *flash-name*. This helps ensure that the combined file does not contain several files with the same name.

1.9.8 Deleting a File

To delete a file or image, enter the following command at the prompt:

`delete` *`index-number`*

where *index-number* is the index number of the file to be deleted. To view the index numbers, enter `list` at the prompt.

Note that you are not allowed to delete the file with the index 1. This file is the default system image.

1.9.9 Extracting a File

MOD_FLSH allows you to copy (extract) one of the files in the combined file to a disk on the load host. To do this, enter the following command:

`extract` *`index-number file-spec`*

where *index-number* is the index number of the file to be copied and *file-spec* is the file specification that you want to give to the extracted file.

Note the following:

- On OpenVMS load hosts, a file that has been extracted cannot be reloaded separately. For example, if you extract a script file from the combined file, it cannot be loaded from the load host as a separate CMIP file. It can only be loaded as part of a combined file.
- On PC and DIGITAL UNIX load hosts, this does not apply. An extracted file can be loaded as a separate CMIP file.

1.9.10 Displaying the Combined File

To redisplay the contents of the combined file or image, enter either of the following commands:

- `List`
- `Relist`

1.10 Using CMIP Over TCP

You can configure the DECNIS to use TCP as well as NSP as its transport protocol for network management. This enables you to issue NCL commands to manage the DECNIS in an IP-only environment.

NSP is used as the DECNIS transport protocol in DECnet environments.

1.10.1 What TCP Enables You to Do

Setting up TCP on the DECNIS enables you to do the following:

- Issue NCL commands to the DECNIS from a DIGITAL UNIX V3.0 host that is using TCP as its transport. The DIGITAL UNIX host does not need to be a DECnet-Plus system.
- Issue NCL commands from the DECNIS console to manage the following types of system:
  - DIGITAL UNIX V3.0 or later systems.
  - SNA Gateway V2.0 or later systems.
  - Other DECNIS V3.1 or later systems.

CMIP over TCP Not Usable From a DECnet-Plus for OpenVMS Host

You cannot manage the DECNIS over TCP from a DECnet-Plus for OpenVMS host. This is because DECnet-Plus for OpenVMS implements TCP according to RFC 1006, while the DECNIS implementation does not.

1.10.2 Managing the DECNIS Over TCP

Table 1–3 lists the tasks you need to carry out before you can manage the DECNIS from a DIGITAL UNIX system using TCP.

Table 1–3 Setting Up TCP

On the DECNIS, set up TCP.	The DECNIS configurator (text-based or Windows) automatically adds the NCL commands to set up TCP to the master NCL script. Alternatively, you can issue the following NCL commands:
	`CREATE TCP`
	`ENABLE TCP`

On the DIGITAL UNIX system, issue the following NCL command:	`SET NCL TRANSPORT=TCP`

Cannot Disable TCP

Note that there is no NCL command to disable TCP. In order to remove TCP from the DECNIS, you must do the following:

- Remove the CREATE TCP and ENABLE TCP commands from the master NCL script.
- Reload the DECNIS.

However, note that you cannot do this on Windows 95/NT load hosts, as you cannot edit the master NCL script file within the clearVISN DECNIS configurator.

1.10.3 Managing Other Systems from the DECNIS

To set up the DECNIS to manage DIGITAL UNIX and SNA Gateway systems using TCP, follow these steps:

1. Set up TCP on the DECNIS by running the DECNIS configurator. Alternatively, issue the following NCL commands:

   ```
   NCL> CREATE TCP
   NCL> ENABLE TCP
   ```

   Result: You can issue NCL commands over TCP, using an IP address to identify the node being managed.

   If you want to use IP node names as well as IP addresses to identify nodes being managed, continue to the next step.

2. Set up the DECNIS to use the IP Domain Name Server by entering the following NCL commands

   ```
   NCL> CREATE IP SERVICES RESOLVER
   NCL> CREATE IP SERVICES RESOLVER SERVER server-name IP ADDRESS ip-address
   NCL> ENABLE IP SERVICES RESOLVER
   ```

   where *server-name* is the name of an IP Domain Name Server willing to answer queries, and *ip-address* is its IP address.

3. Set TCP as the NCL transport to be used on the DECNIS:

   ```
   NCL>  SET NCL TRANSPORT TCP
   ```

1.10.3.1 Effect of Setting NCL Transport to TCP

Issuing the NCL command SET NCL TRANSPORT TCP affects the way node specifications are interpreted in NCL commands, as follows:

- If a node name is used to specify the node being managed, it is interpreted as an IP node name. If the name cannot be resolved by the IP Domain Name Server, the command will fail.
- If an IP address or DECnet Phase IV address is used to specify the node being managed, it is interpreted either as an IP address or as a DECnet address on the basis of its syntax.

1.10.3.2 Switching Back to DECnet Transport

To switch back to using DECnet transport, issue the following command:

```
NCL>  SET NCL TRANSPORT DECNET
```

1.11 Using the DECnet-Plus Naming Services

1.11.1 Introduction

On a DECnet-Plus network, node name and addressing information can be stored within the DECdns namespace or a local namespaces.

1.11.2 DECdns

DECdns servers in a network store addressing information about all the nodes in the network. DECdns clerk software running on a host system can access this addressing information. In this way, each host system does not have to maintain a database of node addressing information.

1.11.3 Local Namespace

With local namespaces, each node in a network maintains a discrete, local namespace, containing addressing information about the other nodes in the network. You can use local namespaces as well as, or instead of, using DECdns servers in a network.

1.11.4 The DECNIS and the Naming Services

The DECNIS does not use the naming services directly, as it does not contain DECdns clerk software.

However, the load-host and DECNIS text-based configurators can use the naming services to do the following:

- Build the Known Towers database on the DECNIS. This database contains specifications of all the DECnet nodes to which the DECNIS sends messages; for example, event sinks. Refer to Section 1.12 for details.
- Register the DECNIS node in the local or DECdns namespace. Refer to the manual *DECNIS Installation and Configuration for OpenVMS and DIGITAL UNIX* for details.

Note that the clearVISN DECNIS configurator does not make use of the naming services.

1.12 Translation of Node Names on the DECNIS

1.12.1 Introduction

Several management tasks, if performed using NCL commands, require you to enter the names of other nodes in the network (for example, in Section 5.4 you enter the DECdns node name of the event sink).

If the node names are those of DECnet Phase IV or DECnet-Plus nodes, then you must set up information to allow the DECNIS to translate each name to a complete specification of how the remote node can be reached. This is necessary even if the network is using DECdns and/or the local namespace to translate node names into addressing information.

The complete DECnet node specification is known as a **tower set**.

1.12.2 Tower Sets

A tower set is a list of the protocols and addresses by which the DECNIS can reach another node in the network.

You allow the DECNIS to reach another node in the network by entering the tower set of the remote node in the Known Towers database of the DECNIS.

1.13 Entering Tower Sets in the Known Towers Database

You can use either DECNIS configurator to create tower sets automatically. You can also issue NCL commands to create tower sets if you wish.

1.13.1 Entering Tower Sets Using the DECNIS Text-Based Configurator

The DECNIS text-based configurator can create tower sets in two ways:

- Using node specifications in the DECdns or local namespace.
- Using DECnet Phase IV or NSAP addresses that you supply.

1.13.1.1 Using a Naming Service

The load-host configurator and the DECNIS text-based configurator can use DECdns or the local namespace to find node specifications. In order to do this, the configurators must be running on a host which meets the following requirements:

- It must be running DECdns clerk software.
- **One** of the following must be true:
  - A suitable DECdns server is available.

- A local namespace is set up on the host.

Procedure

In the load-host configurator, you are asked whether or not you want to use a naming service to generate node specifications.

Select Yes if you want the configurator to use a naming service.

1.13.1.2 Creating Tower Sets Without Using the Naming Services

If you do not want the configurators to use DECdns or the local namespace, answer No when asked if you want to use a naming service. You will then be asked to specify NSAP or Phase IV addresses instead of node names.

1.13.1.3 Result

The DECNIS text-based configurator will construct the relevant tower sets, and will create entries in the master NCL script to add them to the Known Towers database.

1.13.2 Entering Tower Sets Using the clearVISN DECNIS Configurator

The clearVISN DECNIS configurator only creates tower sets using DECnet Phase IV or NSAP addresses that you supply. However, it allows you to construct a private database that maps node names to addresses. Once you have set up the node name/address mappings, you can enter node names where you would otherwise enter addresses. Refer to the manual *clearVISN DECNIS Configurator User Guide* for details.

1.13.3 Entering Tower Sets in the Known Towers Database Using NCL Commands

Use the following command to add node names to the Known Towers database:

```
NCL> CREATE NODE decnis SESSION CONTROL -
_NCL> KNOWN TOWER node-name TOWERS = {(tower1), (tower2), ...}
```

where:

decnis	is the name of the DECNIS.
node-name	is the name by which the remote node is known in the network.
tower1 *tower2*	are tower sets that describe the protocols and NSAP addresses used to communicate with the remote node.

You must enter one tower for each of the remote node's NSAP addresses.

1.13.4 Tower Set Structure

Each tower set has the following form:

```
([DNA_CMIP-MICE], [DNA_SESSIONCONTROLV n, NUMBER = 19],
[DNA_NSP], [DNA_OSINETWORK, NSAP-address])
```

n	is 3 if the tower set refers to a DECnet-Plus node. is 2 if the tower set refers to a Phase IV node.
NSAP-address	is the NSAP address of the node to which the tower set refers. You can specify the NSAP address in DIGITAL format or in OSI format (see Section 22.8).

1.13.5 Example: Adding a Tower Set Using NCL

You want to add the node name ORG:.NORTH.MANCHESTER.SYSTEM2 to the Known Towers database. Assume that this is a DECnet-Plus node with two NSAP addresses as follows:

37:12345:02-00:AA-02-14-78-66-11:20
37:12345:02-00:AA-02-14-78-66-10:20

Enter the following command to place this node in the Known Towers database:

```
NCL> CREATE NODE decnis SESSION CONTROL -
_NCL> KNOWN TOWER org:.north.manchester.system2 TOWERS = { -
_NCL> ([DNA_CMIP-MICE], [DNA_SESSIONCONTROLV3, NUMBER = 19], -
_NCL> [DNA_NSP], [DNA_OSINETWORK, 37:12345:02-00:AA-02-14-78-66-11:20]), -
_NCL> ([DNA_CMIP-MICE], [DNA_SESSIONCONTROLV3, NUMBER = 19], -
_NCL> [DNA_NSP], [DNA_OSINETWORK, 37:12345:02-00:AA-02-14-78-66-10:20])}
```

where *decnis* is the name of the DECNIS.

1.14 DECNIS Trace Facility

DECNIS Trace Facility (DTF) is a utility supplied with the DECNIS software. DTF traces packets as they traverse through the protocol layers within a router. DTF is an enhancement of DIGITAL's CTF (Common Trace Facility) provided in DECnet-Plus; DTF supports multiple platforms and TCP/IP networks.

Note that on Windows 95/NT PCs, you run DTF from within the clearVISN DECNIS configurator. See the configurator help and the manual *clearVISN DECNIS Configurator User Guide* for details.

For more information about DTF, refer to the document, DTF.TXT. You can find this in the following locations:

Load Host	Location
OpenVMS	SYS$EXAMPLES:DTF.TXT
Windows 95/NT PCs	*install-dir*\DTF.TXT
DIGITAL UNIX	usr/lib/dnet/dtf/files/dtf.txt

1.15 POLYCENTER Manager on Netview

The DECNIS can be managed using the POLYCENTER™ Manager on Netview® and the POLYCENTER DECnet Manager. POLYCENTER is a comprehensive system of integrated network management software.

POLYCENTER Manager on Netview lets you control the DECNIS using SNMP management.

The POLYCENTER DECnet Manager lets you monitor, manipulate and test the modules and entities of the management module in the same way as NCL, but it uses a DECwindows interface and provides graphical representations of the status of the system. You can enter the commands described in this manual using POLYCENTER DECnet Manager instead of NCL, either by entering the commands directly or by using the DECwindows™ interface.

The POLYCENTER products also provide alarm notification and performance monitoring.

If you have POLYCENTER DECnet Manager installed when you install the DECNIS software, the installation will customize the Iconic Map window to add the load-host configurator and the DECNIS text-based configurator to the Applications menu.

Refer to the POLYCENTER documentation for more details.

2

Using a Console Terminal on the DECNIS

2.1 Introduction

The DECNIS provides a modem-controlled asynchronous console port on the DECNIS management processor cards MPC–II or MPC–III. The console port provides the following features:

- Support for a VT100-compatible console terminal
- Full-duplex DEC Standard 052 modem control
- An implementation of the Network Control Language (NCL), for managing local and remote nodes from the console terminal
- A console break-in facility
- Password protection

2.1.1 Console Port Location

The console port is situated on the Management Processor Card MPC–II or MPC–III.

2.1.2 Requirements

The following table gives the items required to connect a terminal to the DECNIS console port, and use the console.

Item	Used to...
RS232 cable/interface	Connect the terminal to the console port
DECNIS MPC–II or MPC–III processor card	Provide the console port
VT100-compatible terminal or an MS–DOS PC running a VT100 terminal emulator program	Provide the console

2.2 Starting the Console

Table 2–1 shows how to start a console session at a terminal physically connected to the DECNIS. Refer to Section 2.4.4 to find out how to start a Telnet console session.

If you start the console at a physically connected terminal, then autobaud recognition (autobauding) is always enabled the first time you start the console. Once you have started the console, you can disable autobauding; see Section 2.7.1.

Table 2–1 Starting the Console from a Physically Connected Terminal

If...	Then do this...
Autobaud is enabled	1. Press Return twice. 2. When the character # is displayed, press Ctrl/C. 3. If # is not displayed immediately, repeat steps 1 and 2.
Autobaud is disabled	Press Ctrl/C.

2.2.1 Console Prompt

When the console is started, it will display a copyright screen and the console prompt. The console prompt will be one of the following:

- The DECNIS node name. For example:

  ```
  nis100>
  ```

- If a node name has not been set, the prompt is:

  ```
  console>
  ```

2.3 Exiting from the Console

To exit from the console, do either of the following:

- Enter the following command:

  ```
  console> Exit
  ```

- Press Ctrl/C

2.4 Using Telnet to Set Up a Remote Connection to the Console

You can use the the Telnet protocol to connect to the DECNIS console from a remote terminal. This section describes how to do this.

2.4.1 Introduction

The Telnet protocol is defined in RFC 854. It is a duplex communications facility that allows a standard interface between terminal devices and terminal processes.

The Telnet protocol runs over TCP.

Figure 2–1 shows a terminal using Telnet to connect to the DECNIS. The terminal system is the **Telnet client**; the DECNIS is the **Telnet server**.

2.4.1.1 DECNIS Telnet Implementation

The DECNIS always acts as the Telnet server.

The Telnet client software is not installed on the DECNIS. This prevents the local DECNIS console from being used to set up a remote connection via Telnet to another DECNIS.

2.4.2 Requirements for the Remote Terminal System

The remote terminal system must have:

- A Telnet client implementation installed.
- An IP network connection to the DECNIS.

2.4.3 Setting Up Telnet on the DECNIS

Table 2–2 shows the tasks you must complete on the DECNIS to enable a remote terminal to connect to it using Telnet.

Figure 2–1 Use of Telnet for Remote Connection to DECNIS

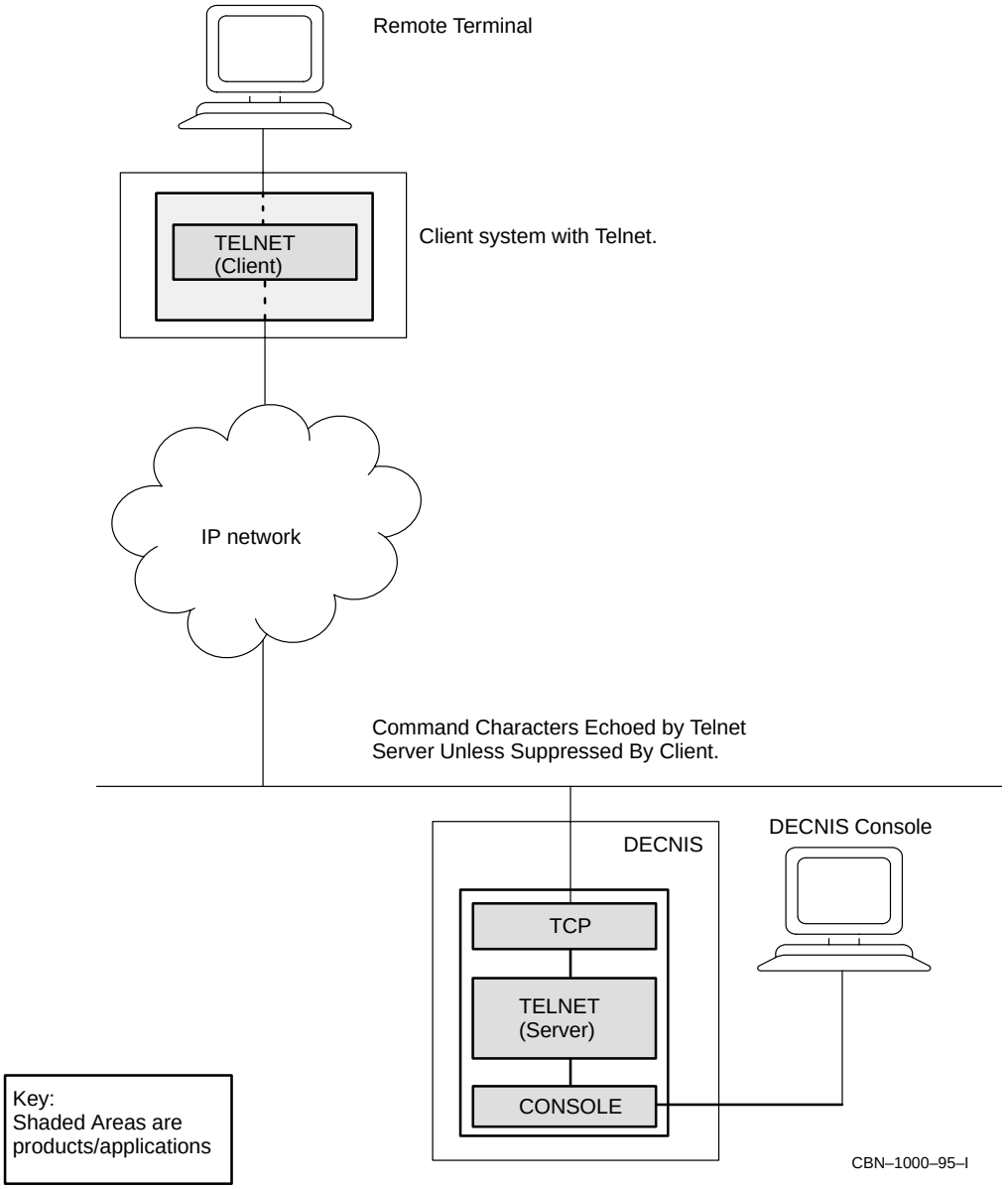


Table 2–2 Tasks Required Before Establishing a Telnet Connection

Task	Action
Enable IP routing	Run the DECNIS configurator (text-based or Windows) and request IP routing.

Create and enable TCP	Run the DECNIS configurator (text-based or Windows); this automatically sets up TCP. Alternatively, issue the following NCL commands:
	`NCL> CREATE TCP`
	`NCL> ENABLE TCP`

If Telnet has been disabled, enable it. Note that Telnet is enabled by default. However, if Telnet connections have been disabled, you must re-enable them.	From the DECNIS console, enter the following command: `console> set telnet on`

Refer to Chapter 1 for more information about the configurators.

2.4.4 Connecting to the DECNIS Console

To create a Telnet connection to the DECNIS console, follow these steps:

Step	Task	Action
1.	From the remote terminal, start Telnet	Issue the appropriate command. For example, on a DIGITAL UNIX system, enter: `# telnet`
2.	From the remote terminal, connect to the DECNIS console:	`telnet> CONNECT` *id* where *id* is the IP node name of the DECNIS or one of its IP addresses.
3.	Enter a password.	If you have set up a console password, enter it. If you have not set up a console password, enter the network management password. See Section 1.8 and Section 2.10.3
4.	Set up the character echoing option	See Section 2.4.5.

Result of Connecting to the Console

Once connected to the console, you will be asked for a password. Do one of the following:

- If you have set up a console password, as described in Section 2.10, enter this password.
- If you have not set up a console password, enter the DECNIS network management password. Refer to Section 1.8 for details.

When you have entered a password, you will see the console prompt.

2.4.5 Character Echoing Options

You can choose whether characters entered at the remote terminal are echoed locally or remotely. Remote echoing is done by the DECNIS; local echoing is done by the terminal.

2.4.5.1 Remote Echoing

It is recommended that you choose remote echoing. This offers the following advantages:

- Password characters are not echoed, as the DECNIS Telnet server can to determine what should be echoed in the current command context.
- Keyboard command control sequences and responses are processed without you having to press carriage return.

Remote echoing is sometimes referred to as character mode.

2.4.5.2 Local Echoing

It may be preferable to use local echoing if the period between entry and display of a typed character is unacceptable (for example, if the network communications are slow).

If characters are echoed locally, only commands ending with a carriage return are sent to the DECNIS console. For example, in local echoing mode, if you press Up Arrow to recall the previous command line, you then have to press carriage return to send it to the server and get the response.

Local echoing is sometimes referred to as line mode.

2.4.5.3 Procedure for Selecting Echoing Options

The commands needed to select an echoing option will vary according to the remote terminal's operating system.

Example: OpenVMS

To select remote echoing at an OpenVMS terminal, enter the following command:

```
telnet> SET MODE CHAR
```

To select local echoing at an OpenVMS terminal, enter the following command:

```
telnet> SET MODE LINE
```

2.4.5.4 Returning to the Telnet Prompt from the Console Prompt

To return to the Telnet prompt from the console prompt, do the following:

- If you are using remote echoing, type CTRL/].
- If you are using local echoing, type CTRL/] and then press Return.

2.4.6 Disabling Telnet Connections

To disable all Telnet connections to the DECNIS, enter the following command from the DECNIS console:

```
console> Set Telnet off
```

2.4.7 Specifying Allowed Connections

By default, all Telnet connections to the console are allowed. However, you can specify a list of allowed Telnet sources. If you do this, only the systems you specify will be able to Telnet to the DECNIS console.

2.4.7.1 Procedure

To specify a list of allowed terminals, enter the following command at the DECNIS console:

```
console>  Set telnet sources ip-address ...
```

where *ip-address* is either an actual IP address or an IP address with wildcard elements. The valid IP address formats are shown below:

This IP address format...	Allows these IP addresses to connect...
*	Any
130.223.14.2	130.223.14.2
130.223*	130.223.*xxx*.*yyy* where *xxx* and *yyy* are numbers in an IP address.
*.223.14.2	*xxx*.233.14.2 where *xxx* is a number in an IP address.

2.4.7.2 Changing Allowed IP Addresses

If you want to change allowed IP addresses, you must reenter all of the information. This is because the current `set telnet sources` command deletes the previous command details.

2.4.7.3 Showing Allowed Telnet Connections

To view the allowed Telnet connections, enter the following command at the DECNIS console prompt:

```
console> Show telnet
```

2.5 Using Console Commands

You can enter console commands to do the following tasks:

Task	Refer to...
Control the console	Section 2.7 to Section 2.10
Control Telnet connections	Section 2.4.3, Section 2.4.6 to Section 2.4.7.3
Do an IP Domain Name Server lookup	Section 2.11
Control the contents of flash memory	Chapter 3
Load and dump the DECNIS	Section 2.12
Start NCL	Section 2.13
Use console break-in	Section 2.14 to Section 2.15

2.5.1 Displaying Console Command Descriptions

To see a list of console commands, enter the following command:

```
console> help
```

2.5.2 Help with Console Command Parameters

To find out the valid parameters for a command, press ? and Return after a command verb or parameter.

Example

If you enter:

```
console> set ?
```

The screen displays all the attributes of the `set` command.

If you enter:

```
console> set console ?
```

The screen displays all the attributes of the `set console` command.

2.5.3 Console Command Summary

Table 2–3 lists the console commands.

Table 2–3 Console Command Summary

Command	Description	
Add *script node:filename*	Adds a configuration script to flash memory. Refer to Chapter 3	
Boot	Reruns the system self-test and boots the DECNIS	
Cls	Clears the screen	
Delete *n* where *n* is a flash index number	Marks a file in flash memory as unusable. Refer to Chapter 3	
Disconnect	Disconnects from the modem and exits	
Dump	Dumps the DECNIS and reruns the system self-test	
Exit	Exits the console	
Help	Displays help text	
Load...	Loads the DECNIS using the method specified	
Lookup	Performs an IP Domain Name Server lookup.	
Ncl	Starts the Network Control Language (NCL) utility	
Password	Sets a new console password	
Restart	Reloads the DECNIS using its default method	
Set console...	Sets console attributes, as follows:	
	Set console [no]autobaud	Disables or enables console autobauding
	Set console [no]disconnect	Disables or enables disconnecting the modem on console exit
	Set console [no]modem	Disables or enables modem control on the console port
	Set console speed	Sets the console speed
	Set console timeout	Sets the default inactivity timeout period for console sessions using a physically connected terminal

(continued on next page)

Table 2–3 (Cont.) Console Command Summary

Command	Description	
Set flash...	Sets flash memory attributes, as follows:	
	Set flash boot *n* (only valid for MPC-III cards)	Specifies the area of flash memory from which the DECNIS should be loaded.
	Set flash image *n*	Specifies the software image to be loaded from flash memory
	Set flash script *n*	Specifies the configuration script to be loaded from flash memory
	Set flash update *n* (only valid for MPC-III cards)	Specifies the area of flash memory to be cleared and updated with a new image or combined image.
Set session...	Sets attributes for the session, as follows:	
	Set session size	Sets the screen size for the session
	Set session timeout	Sets the inactivity timeout period for the current session
	Set session username	Sets the username for the session
Set Telnet...	Sets Telnet attributes, as follows:	
	Set Telnet off	Disables Telnet connections
	Set Telnet on	Enables Telnet connections
	Set Telnet sources	Sets up permitted sources for Telnet connections
	Set Telnet timeout	Sets the default inactivity timeout period for Telnet sessions
Show console	Shows current console attributes	
Show flash	Shows current flash memory attributes and contents	
Show session	Shows current session attributes	
Show Telnet	Shows current Telnet attributes	
Update *node*:*file-name*	Updates with the specified file the area of flash memory specified with set update	
Users	Displays the current users of the console	

2.6 Editing Console Commands

Table 2–4 lists the keys used to edit console commands, and recall previous commands. Note that editing is always in overstrike mode.

Table 2–4 Keys for Editing Console Commands

Use this key...	To do this...
Up arrow	Recall up to ten previous commands.
Down arrow	Re-examine command previously recalled
Right arrow	Move the cursor one character to the right
Left arrow	Move the cursor one character to the left
Ctrl/R	Redraw current line
<X or DEL	Delete the character to the left of the cursor
Ctrl/U	Delete text from the cursor position to the start of the line

Table 2–5 lists the keys used to control command output.

Table 2–5 Keys to Control Command Output

Use this key...	To do this...
Ctrl/C	Cancel the current command or output
Ctrl/O	Suspend or continue display of output to the terminal
Ctrl/X	Cancel the current line, and delete data that has been typed but not yet displayed or executed

2.7 Disabling Autobauding and Setting the Console Speed

By default, the console enables autobauding when you start the console from a physically connected terminal. This means that the console automatically tries to set the console port speed to be compatible with the terminal port speed.

If your terminal has a fixed port speed, you may want to turn off autobauding, and set the console port speed yourself. Section 2.7.1 describes how to do this.

If you turn off autobauding without setting a new port speed, then the next time you start the console, it will set it to the last speed used.

Note that autobauding does not apply to Telnet sessions.

2.7.1 Procedure: Disabling Autobauding

Follow these steps:

1. To turn off autobauding, enter the following command:

   ```
   console> set console noautobaud
   ```

2. To set the console port speed, enter the following command:

   ```
   console> set console speed n
   ```

   where *n* is one of the following speeds, in kbits/s:

   300, 600, 1200, 2400, 4800, 9600, 19200

3. If the new speed is different from the one currently used, you are asked to confirm the change. Enter Yes.

Result: The new console speed will take effect immediately.

2.7.2 Autobauding Restored When DECNIS Is Restarted

If you power up the DECNIS with the dump button pressed in, then the next time you use the console, it will use autobauding.

2.8 Setting Inactivity Timeout Periods

The console will automatically exit if there has been no console activity for a given period. This period is known as the **inactivity timeout period**. Console activity means either that you have typed something, or that the console has displayed something.

You can change inactivity timeout periods, or disable inactivity timeout altogether. Table 2–6 shows the different types of console inactivity timeout periods that you can specify.

Default Inactivity Timeout Period

All inactivity timeout periods are by default set to 300 seconds.

2.8.1 Procedure

To change the inactivity timeout period, enter the appropriate command as shown in Table 2–6:

Table 2–6 Commands to Change Inactivity Timeout

To do this...	Enter this command
Set the default inactivity timeout for sessions using a physically connected console terminal.	`console> Set console timeout` *n*
Set the default inactivity timeout for Telnet sessions	`console> Set telnet timeout` *n*
Set the inactivity timeout for the current console session (Telnet or physically connected)	`console> Set session timeout` *n*

where *n* is a decimal number in the range 10 to 65535. It specifies the number of seconds that the console is allowed to be inactive before it exits.

2.8.1.1 Disabling Inactivity Timeout

If you do not want console sessions to exit automatically after an inactive period, make the inactivity timeout period zero (0). For example, for Telnet console sessions, enter:

```
console> Set Telnet timeout 0
```

2.9 Disabling and Enabling the Modem

By default, modem control is disabled. If your terminal or PC does not have a modem, or its modem will work without console modem control, then leave modem control disabled.

However, If your terminal or PC has a modem or a null modem cable, you may enable modem control.

Note that the modem control does not apply to Telnet sessions.

2.9.1 Procedure

- To enable modem control, enter the following command:

  ```
  console> set console modem
  ```

- To disable modem control again, enter the following command:

  ```
  console> set console nomodem
  ```

2.9.2 Automatic Modem Disconnection

By default, the console port modem is automatically disconnected when the console exits. If you want the modem to remain connected when the console exits, enter the following command:

```
console> set console nodisconnect
```

2.10 Setting and Disabling a Console Password

You can limit access to the console by setting a console password. You can also disable a current password.

2.10.1 Procedure: Setting a Password

To set or change a console password, enter this command:

```
console> password
```

You will then be asked to enter the new password and to reenter it, for verification. If a console password already exists, you are asked for the current password before being allowed to change it.

Syntax

The password is an alphanumeric string of up to 8 characters.

2.10.2 Procedure: Disabling a Password

To disable an existing password, follow these steps:

1. Enter this command:

   ```
   console> password
   ```

2. Enter the current password when prompted.
3. Press Return at the prompts for `New Password` and `Verification`.

Result: You will not be asked for a password when you access the console.

2.10.3 Telnet Password Requirement

Whether or not you have set up a password, you are still required to enter a password if you connect to the console using Telnet.

If you have not set up a console password, the default password is the network management password of the DECNIS. Refer to Section 1.8 and Section 6.2.3 for more information about this password.

2.11 Looking Up IP Addresses and Node Names

You can use the lookup command to find the IP address corresponding to an IP node name, or vice versa.

2.11.1 Requirement

Before you can issue the lookup command, you must have set up the IP Domain Name Server by entering the following NCL commands

```
NCL> CREATE IP SERVICES RESOLVER
NCL> CREATE IP SERVICES RESOLVER SERVER server-name IP ADDRESS ip-address
NCL> ENABLE IP SERVICES RESOLVER
```

where *server-name* is the name of an IP Domain Name Server willing to answer queries, and *IP-address* is its IP address.

2.11.2 Procedure

To do a Domain Name Server lookup, enter the following command:

```
console> lookup node-spec
```

where *node-spec* is either an IP address or an IP node name.

Result: If you enter an IP address, the console displays the corresponding IP node name. If you enter an IP node name, the console displays the corresponding IP address.

2.12 Loading and Dumping the DECNIS

This section describes the commands used to control loading, rebooting and dumping of the DECNIS.

2.12.1 Booting the DECNIS

To reboot the DECNIS, follow these steps:

1. Enter the following command:

   ```
   console> Boot
   ```

2. You are asked to confirm that you want to go on. Enter Yes.

Result: The DECNIS will rerun the system self-test. It will then reload, using its default type of loading. This is either of the following:

- On a Windows 95/NT load host, the type of loading specified in the clearVISN DECNIS configurator.

- On any other load host, the default is the type of loading specified in the load-host configurator.

2.12.2 Dumping the DECNIS

To force the DECNIS to dump, follow these steps:

1. Enter the following command:

   ```
   console> Dump
   ```

2. You are asked to confirm that you want to go on. Enter Yes.

Result: The DECNIS will dump to a dump host. Then, once the dump completes (or times out), it will run its self-test. Finally, it will reload, using its default type of loading (as defined in Section 2.12.1).

2.12.3 Loading the DECNIS

To reload a DECNIS, and specify the type of loading, enter the following command:

```
console> load loadtype
```

where *loadtype* is one of the following:

- –FLASH to reload the image and (if present) the configuration file from flash memory.

  If there is more than one image and/or configuration file in flash memory, you can specify which one will be loaded; refer to Chapter 3 for details.

- –NETWORK to reload the software image and configuration files from a load host.

- A file name to load a specified file from a load host. Do not specify a directory or pathname.

Specifying a File Name

If you specify a file in the load command, be sure that there is a file of that name in the correct directory on the load host.

2.12.4 Restarting the DECNIS

The restart command has the same effect as the load command, except that you cannot specify the type of loading or a load file. The restart command always uses the default type of loading (as defined in Section 2.12.1).

Procedure

To restart the DECNIS, enter the following command:

```
console> restart
```

2.13 Using the Network Control Language

You can run the Network Control Language (NCL) on the console. Refer to Section 1.3 for a general description of NCL.

Note that only one user at a time can run NCL during a DECNIS console session. If a second user attempts to use NCL, a message appears asking if the current user should be logged off.

2.13.1 Starting NCL

To start NCL on the console, enter the following command:

```
console> ncl
```

2.13.2 Setting the Terminal Screen Size

Each time you start NCL, the NCL program will try to determine the screen size, using ANSI escape sequences. You can override this automatic sizing by specifying the screen size to be used during the current console session.

2.13.2.1 Procedure

To set the terminal screen size for the current session, enter the following command:

```
console> set session size row column
```

where *row* is the number of rows on the screen and *column* is the number of columns on the screen.

Example

To specify a screen size of 24 rows by 80 columns, enter the following command:

```
console> set session size 24 80
```

2.13.3 Exiting from NCL

To exit from NCL on the console, type any of the following at the NCL prompt:

- Ctrl/Z
- Exit and Return

You will return to the console prompt.

2.13.4 NCL Command Format

2.13.4.1 Managing the Local DECNIS

When using NCL to manage the local DECNIS, issue the NCL command in the following format:

verb entity-name [{*argument*(*s*)/*attribute*(*s*)}]

2.13.4.2 Managing a Remote Node

When using NCL to manage a remote node, specify the node in the NCL command, as follows:

verb NODE *node entity-name* [{*argument*(*s*)/*attribute*(*s*)}]

where *node* is one of the node specifications listed in Table 2–7.

2.13.4.3 Supplying Access Control Information

To supply access control information in console NCL commands, use the following format:

verb NODE *node/username/password entity-name* [{*argument*(*s*)/*attribute*(*s*)}]

2.13.4.4 Using Default Nodes

If you wish to issue several NCL commands to the same remote node, you can set up that node as the default by entering the following command:

NCL> SET NCL DEFAULT ENTITY NODE *node/username/password*

You can then enter NCL commands without specifying the node name.

2.13.5 Setting Default Security

If you have set default access to NCL on a remote node, you can set default security for that node by issuing the following NCL command:

NCL> SET NCL DEFAULT ACCESS BY USER *username*, PASSWORD *password*

2.13.6 Managing the DECNIS and Other Nodes

You can use console NCL to manage any node that is normally manageable by NCL over a DECnet network, for example, DECNIS or DECnet-Plus nodes. (DECnet-Plus was formerly known as DECnet/OSI).

In addition, if you set up the TCP entity on the DECNIS, as described in Section 1.10, you can manage the following types of system, whether or not they are running DECnet:

- DIGITAL UNIX V3.0 or later systems
- SNA Gateway V2.0 or later systems

2.13.6.1 Specifying the Node

Table 2–7 shows the syntax for specifying a remote node in console NCL commands.

Table 2–7 Node Specifications in NCL commands

Node Specification	Example Syntax	See Also
DECnet Phase IV address	`NCL>SHOW NODE 1.100`	
DECnet Node name	`NCL>SHOW NODE nis100`	Section 2.13.6.2 and Section 2.13.6.5
IP address	`SHOW NODE 16.36.16.100`	Section 2.13.6.3
IP node name	`SHOW NODE nis100`	Section 2.13.6.4

2.13.6.2 Using DECnet Node Names in NCL Commands

To use a DECnet node name to identify a node in a console NCL command, you must do the following:

1. In the CREATE user NCL script file for the DECNIS, enter NCL commands that create a SESSION CONTROL KNOWN TOWER entity for each node you expect to manage from the DECNIS. Refer to Section 1.5.6 and Section 1.13.3 for more information.
2. Use the name of the SESSION CONTROL KNOWN TOWER entity for the node as the node name.

Example

This is an example of an NCL command to create a SESSION CONTROL KNOWN TOWER entity for the node NIS100.

```
CREATE NODE NIS100 SESSION CONTROL KNOWN TOWER -
 AA-00-04-00-DE-11-A0-AA-F9-6F-56-DE-8E-00:.nis100 TOWERS -
  {([ DNA_CMIP-MICE ], [ DNA_SessionControlV2 , Number = 25 ], -
   [ DNA_NSP ], [ DNA_OSINetwork , 49::00-01:aa-00-04-00-64-04:20 ])}
```

2.13.6.3 Using IP Addresses in NCL Commands

In order for you to use an IP address to identify a node in a console NCL command, the TCP entity must have been created and enabled on the DECNIS. This is done automatically by the DECNIS configurators.

2.13.6.4 Using IP Node Names in NCL Commands

To use an IP node name to identify a node in a console NCL command, you must follow all the steps in Section 1.10.3.

2.13.6.5 How the DECNIS Interprets Node Names and Addresses

Refer to Section 1.10.3.1 for details.

2.13.7 Help on NCL

The online help within NCL gives details of the commands, entities, attributes and characteristics supported. Later chapters in this manual give details of the NCL commands used to carry out specific tasks.

Starting NCL Help

To get NCL help on the DECNIS, follow these steps:

1. Call up the online help by entering HELP at the NCL prompt:

   ```
   ncl> HELP
   ```

2. Select the menu item DECNIS.

Exiting from NCL Help

To exit from NCL help, type Ctrl/D or Ctrl/Z.

2.13.8 Editing NCL Commands

Table 2–8 lists the keys used to edit NCL commands, and recall previous commands.

Table 2–8 Keys for Editing NCL Commands

Use this key...	To do this...
Up arrow	Recall a previous command
Down arrow	After recalling a previous command, recall the next command in the series
Left arrow	Move the cursor one character to the left
Right arrow	Move the cursor one character to the right
Enter or Return	Enter a command you have typed
F14 or Insert	Toggle between inserting characters and typing over characters
F12 or Ctrl/A	Move the cursor to the beginning of the line

(continued on next page)

Table 2–8 (Cont.) Keys for Editing NCL Commands

Use this key...	To do this...
Ctrl/E	Move the cursor to the end of the line
Ctrl/R	Redraw current line
Ctrl/U	Delete text from the cursor to the beginning of the line
<X or DEL	Delete the character to the left of the cursor
F13 (not on all keyboards)	Delete the word to the left of the cursor
-	Type a hyphen at the end of a line to continue typing a command on the next line
^ *string* Return or *string* Find Return	Recall the last NCL command that starts with the characters in *string*

2.13.8.1 Line Length

The maximum length of a line in an NCL command is 1024 characters. The effect of this is as follows:

- If you are typing in an NCL command, when you reach the righthand side of the screen, the cursor automatically moves to the next line.
- If you have typed 1024 characters or less, you can at any point go back to previous lines (using the left arrow key) and edit the command.
- Once you type more than 1024 characters, the cursor jumps to the next line. At this stage, you cannot go back and edit previous lines. You must press Enter to enter the command.

2.13.8.2 Command Length

The maximum length of an NCL command (as opposed to a line) is 2048 characters. To type in a command longer than 1024 characters, use a hyphen as a continuation character.

You can enter the hyphen at any point before you have typed 1024 characters.

2.13.9 Displaying NCL Output

You use NCL SHOW commands to display information about your system. The console provides a UNIX-style **More** function to control the display of NCL command output.

2.13.9.1 Controlling the Display of NCL Output

When you enter an NCL SHOW command, the console displays the first screenful of NCL output. Table 2–9 shows the keys you can then use to control the display of NCL command output.

Table 2–9 Keys to Control NCL Output

Use this key...	To do this...
Space bar	Display the next screen of NCL output.
Return	Scroll one line of output.
q or Q	Terminate the output and return to the NCL prompt.
b or Ctrl/B	Display the previous screen of output.
? or h	Display help text on the More utility.
s or S	Scroll continuously to the end of the output.

2.13.9.2 The More Prompt

When you enter an NCL SHOW command, the console displays the following prompt at the bottom of the screen:

```
--More--(x)
```

where *x* is the screen number of this screen of output.

The screen number is relative to the other screens of output for this command. For example, if an NCL command produces three screens of output, and you are looking at the second screen, the prompt is:

```
--More--(2)
```

If you page back to a previous screen of output, the prompt will contain an arrow. For example, if you page back to the first screen of output, the prompt will be:

```
--More->(1)
```

If you page back until you are looking at the output from a previous command, the prompt will show a negative number. For example:

```
--More->(-3)
```

When the last saved screen is displayed, you will see the prompt:

```
No more screens saved
```

2.13.10 Restrictions

This section describes restrictions in the use of NCL at the console.

2.13.10.1 SNAPSHOT Command

The NCL SNAPSHOT command is not supported in console NCL.

2.13.10.2 NCL File Generation

You cannot create or use NCL script files, or generate log files from the console.

2.13.10.3 DECdns Names in NCL Commands

A restriction applies to NCL commands containing attributes which are DECdns names. If you enter an NCL command of this type at the console, the DECdns name must have the following format:

NSCTS:.*dns-name*

where: *NSCTS* is the namespace creation timestamp of the namespace in which the full name is registered.

dnsname is the DECdns full name of the node.

Definition

The NSCTS is a unique identifier for the namespace in which the DECdns full name is registered. The NSCTS is automatically created when the namespace is created. It consists of 14 hexadecimal digits.

Finding the NSCTS

There are two ways to find the NSCTS for a node. These are described in the following sections.

Finding the NSCTS: Using the Phase IV Address

If the node has a DECnet Phase IV address, follow these steps:

1. Find the DECnet Phase IV address of the node.
2. Enter the following NCL command at the console:

   ncl> SHOW NODE *phase-iv-address* NAME

   where *phase-iv-address* is the Phase IV address of the node.
3. In the display, the hexadecimal digits appearing before the colon (:) make up the NSCTS.

Finding the NSCTS: Using the Node Name

If the node does not have a DECnet Phase IV address, follow these steps:

1. From a DECnet-Plus node, enter the following NCL command:

   ```
   ncl> SHOW NODE node-name DNS CLERK KNOWN NAMESPACE * NSCTS
   ```

   where *node-name* is the DNS name of the node whose NSCTS you are attempting to find.

2. The NSCTS is displayed next to the label NSCTS.

Example Commands

This section gives example NCL commands to which this restriction applies. In the examples:

- The node name to be used as an attribute is:

  shrub.leaf

- The NSCTS is:

  00-12-23-56-77-A0-A1-A2-A3-A4-A5-A6-A7-18

Example 1

To define **shrub.leaf** as an event sink for a router called PEACH, enter this command:

```
ncl> SET NODE peach EVENT DISPATCHER SINK NODE -
_ncl> {00-12-23-56-77-A0-A1-A2-A3-A4-A5-A6-A7-18:.shrub.leaf}
```

Example 2

To set up X.25 security for outgoing calls for the node **shrub.leaf**, enter this command:

```
ncl> SET X25 SERVER SECURITY NODES security_out NODES -
_ncl> {00-12-23-56-77-A0-A1-A2-A3-A4-A5-A6-A7-18:.shrub.leaf}
```

2.14 Using Console Break-in

The console provides an emergency break-in function. This allows you to use a limited set of commands even if the DECNIS is not responding to normal console commands, or commands over the network.

2.14.1 Procedure

To use the break-in function, follow the steps in the following table:

If...	Then do this at the console...
The DECNIS is loading or dumping	1. Press Ctrl/P and then press Return. 2. The ROM console program will start; see Section 2.15 for details.
The DECNIS is not loading or dumping	1. Press Ctrl/P. 2. If a console password is set, you are asked to enter it. If no password is set, you go directly to step 3. 3. You are prompted to enter a single letter of the break-in command to be executed: `Choose one [D]ump, [H]alt, [L]oad, [Q]uit ?`

2.14.2 The Break-In Commands

Table 2–10 describes the break-in commands.

Table 2–10 Break-in Commands

Command	Description
Dump	Dump the DECNIS. This is the same as the console command **dump**.
Halt	Stops the DECNIS and the normal console software. Refer to Section 2.15.
Load	Reload the DECNIS. This is the same as the console command **restart**.
Quit	Return to the normal console prompt.

Note that normal console output is suspended while the break-in prompts are active.

2.15 The ROM Console Program

The DECNIS provides a ROM-based console program which you can access even if the DECNIS is not loaded or the software console program has stopped. The ROM console program provides a limited set of console commands, which you can use for diagnostic purposes or to reload the DECNIS.

2.15.1 Starting the ROM Console Program

The ROM console program starts if you do either of the following:

- Select Halt when using the break-in function.
- Press Ctrl/P and then Return when loading or dumping the DECNIS.

2.15.2 ROM Console Commands

Table 2–11 lists the ROM console commands.

Table 2–11 ROM Console Command Summary

Command	Description
Boot	As in Table 2–3
Dump	Dumps one or more areas of DECNIS memory
Help	Displays a list of ROM console commands
Load	As in Table 2–3
Set console modem	Sets the modem signal power up state, and enables or disables modem control
Set console speed	As in Table 2–3
Set flash boot	As in Table 2–3
Set flash image	As in Table 2–3
Set flash script	As in Table 2–3
Set flash update	As in Table 2–3
Set self_test	Specifies a quick or a full self-test
Show all	Shows console parameters
Show console	As in Table 2–3
Show flash	As in Table 2–3
Show memory	Show memory size
Show self_test	Shows the type of self-test set
Zero [BBRAM]	Empties DECNIS BBRAM (memory in battery-backed RAM). Note that this command does not empty flash memory.

2.15.3 Dumping the DECNIS

To force the DECNIS to dump, enter the following command:

```
dump dumptype
```

where *dumptype* can have any of the following values:

One or more of the following:	
MPC	Dumps memory on processor card
POOL	Dumps pool memory on memory card
CARDS	Dumps Network Interface Card memory
ARE	Dumps ARE memory on memory card
DEFAULT or no parameter	Dumps the DECNIS memory
NONE	Dumps a minimum area of memory. Use this to test that the dump command works.

Example

Enter the following to dump MPC memory and pool memory:

```
dump {mpc, pool}
```

Recommendation

It is strongly recommended that you enter the dump command without parameters, to create a full dump. Normally, a full dump is required to analyze problems correctly.

2.15.4 Enabling Modem Control

By default, modem control is disabled. To enable modem control from the ROM console, enter the following command:

```
set console modem on
```

For more information, refer to Section 2.9.

2.15.5 Enabling Modem Control

To set the state of the modem signals when the DECNIS powers up, enter the following command:

```
set console modem signaltype
```

where *signaltype* can have any of the following values:

DTR
RTS
DSRS

ALL

2.15.6 Set and Show Self-Test

You can choose whether you want the DECNIS to run a full or a quick self-test on starting up.

- For a full self-test, enter this command:

  ```
  set self_test full
  ```

  The full self-test takes about five minutes.

- For quick self-test, enter this command:

  ```
  set self_test quick
  ```

  The quick self-test takes about 30 seconds.

2.15.6.1 Show All Command

The show all command displays useful information about the console parameters set. For example, it might show the following:

```
Model                              : DECNIS 600-VE
Node id                            : 08-00-2B-A1-12-80
Self-test mode                     : Quick
Console speed                      : 19200
Console modem signal powerup state : DTR-on   RTS-on   DSRS-off
Secure console mode                : On
MPC memory size (Mbytes)           : 16
Configured flash RAM size (Mbytes) :
Image index                        :
Script index                       :
Flash contents                     : Index   Name            Size
                                   : 1       sysncl          2260940
                                   : 2       sysnoncl        1911244
```

3

Configuring Nonvolatile Memory Dynamically

3.1 Introduction

This chapter describes how to modify the contents of nonvolatile (flash) memory after the image file or combined file has been loaded.

- Section 3.2 describes the commands and procedures for controlling the content of nonvolatile memory on the MPC–II. The same commands are also applicable to the MPC–III.
- Section 3.3 describes the additional commands required for working with an MPC–III, and illustrates these commands with a worked example.

3.1.1 Dynamic Updates to Nonvolatile Memory

You can use the DECNIS console to add new CMIP files (scripts) to nonvolatile memory dynamically, and to select which script is to be used at the next reboot.

This feature saves having to reload the DECNIS from a host when testing new configuration scripts.

Management Processor Cards MPC–II and MPC–III

The DECNIS console is only supported on the management processor cards MPC–II and MPC–III. The MPC–II has 4 MB of flash memory and the MPC–III has 8 MB.

You can use the console to dynamically add, delete, update and view files in nonvolatile memory, and designate the file to be loaded. The console commands are the same for either card, but there are some additional considerations for the MPC–III, since on this card flash memory is divided into two distinct areas.

3.1.2 Requirements

The ability to dynamically modify the contents of flash memory is only useful if the DECNIS is configured so that it loads from flash memory. Chapter 1 describes how to prepare and configure a load file for the DECNIS; Chapter 8 describes the basic procedure for setting up the DECNIS to load from flash memory.

3.2 Controlling Nonvolatile Memory on the MPC–II

The commands described in this section also apply to the MPC–III. However, before you can use them on the MPC–III, you need to do some additional configuration; refer to Section 3.3 for details.

3.2.1 Adding a File

You can add script files and profiles to flash memory using the ADD console command, as follows:

```
console> add flash-name host:file-name
```

where:

flash-name	is the name to be used for the script or profile file in flash memory.
host	is the IP address or name of the remote host.
file-name	is the specification of the file to be added.

You can keep adding scripts and profiles until flash memory is full. It may be possible to add as many as 32 different scripts into flash memory, depending on the size of the scripts.

3.2.1.1 Script File Names

If you want to be able to use a script file as the default script, its flash name should be of the form `script` or `script.`*yy*. The extension *yy* can be any string. Script file names do not have to be unique.

3.2.1.2 Profile File Names

The flash name used for a profile file must be the correct name for the profile (`mcnm_prf`, `x25l2_prf` or `x25l3_prf`). Delete any older versions from flash memory and from the host.

3.2.1.3 Remote Host System Requirements

The remote host system is the system where the scripts are held. It need not be the load host used for the load image, but it must meet the following requirements:

- It must have a TFTP server.
- The load files and the directories in which they are held must be accessible by the TFTP server. For example, on OpenVMS and DIGITAL UNIX systems, they must be world readable.

Special Considerations

Refer to the release notes for information about problems found when using particular TFTP servers.

For example, problems have been found with the TFTP server currently provided by UCX software on OpenVMS hosts. The DECNIS release notes describe the problems and solutions.

3.2.2 Selecting the Script or Image to Be Loaded

If there are several configuration scripts in flash memory, you can specify which should be loaded on the next load from flash memory. If you do not specify the script to be loaded, then by default, the first file with the name `script` or `script.`*`yy`* is loaded.

You specify the script to be loaded by giving its **index number**. The next sections explain how to find the script index number and how to specify the script.

Section 3.2.3.2 explains how to specify the image to be loaded

3.2.2.1 Finding the Flash Index Numbers

Each new file is added to flash memory in sequence and an index number is assigned to it. To find the flash index number enter the command:

```
console> show flash
```

This displays the images and configuration scripts in flash memory. For example:

```
Image Index           0
Script Index          7
Flash Contents        Index   Name            Size      Date
  sumchk ok  02133    1       nis_v4.0-1B     3322310   1996-08-27-15:37:32.380
  sumchk ok  32415    2       script          7416      1996-08-27-16:21:57.308
  sumchk ok  42836    3       ospf            5824      1996-08-27-16:34:24.308
  sumchk ok  28463    4       script.1        8420      1996-08-27-16:54:17.367
  sumchk ok  24778    5       mcnm_prf        1028      1996-08-27-17:12:46.335
  sumchk ok  32449    6       x25l2_prf       18650     1996-08-27-17:35:08.153
  sumchk ok  17232    7       x25l3_prf       23826     1996-08-27-17:54:21.302
```

3.2.2.2 Using the Default Script

The default image is the first or only file in the displayed list.

The default script is the first file called `script` or `script.`*yy*, where *yy* can be any string.

For example, you may have two script files named `script`, one with the index number 2 and one with the index number 3. The script file with the index number 2 will be loaded by default.

3.2.3 Selecting the File to Be Loaded from Flash Memory

This section describes how to override the default script by specifying another script to be loaded.

3.2.3.1 Specifying the Script to Be Loaded

To specify a configuration script to be the next script loaded, enter the following command:

```
console> set flash script index-number
```

where *index-number* is the index number of the image or script.

If you want to go back to the default script, enter 0 (zero) for the *index-number*.

Note that if you clear the contents of NVRAM (for example, by holding the dump button in when you power up the DECNIS), this value is reset to the default (0).

Example

In the example display in Section 3.2.2.1, you wish to select the file called `script.1` as the script file to load. Enter the following command:

```
console> set flash script 4
```

The file `script.1` will be loaded when you next reboot the DECNIS.

3.2.3.2 Specifying the Image to Be Loaded

If you have loaded more than one image into flash memory, you can specify which should be loaded next. Although you would not normally load more than one image into flash memory, you may wish to do this for test purposes.

Procedure

The default image is the first image in the displayed list. To specify another image, enter the following command:

```
console> set flash image index-number
```

where *index-number* is the index number of the image. To go back to loading the default image, enter 0 (zero) for the *index-number*.

3.2.3.3 Entering Incorrect Index Numbers

If you specify a nonexistent or inappropriate index number, and then try to load the DECNIS from flash memory, the illegal number will be ignored. The DECNIS will instead try to load the default image or script.

An inappropriate index number is one that is wrong for the type of file, for example, if you specify as the script number the index number of a profile file.

3.2.4 Deleting an Entry

You can remove a script or other file entry from the flash index.

Procedure

To remove a file, enter the command:

```
console> delete index
```

where *index* is the index number of the file being deleted.

This marks the file as deleted. It does not physically remove the file from flash memory, it simply removes the entry for the file from the flash index. The other index entries are not affected.

Example

In the example display in Section 3.2.2.1, you wish to delete the file called `script`. You would enter the following command:

```
console> delete 2
 File index 2 will be marked as deleted,
 but the space cannot be recovered, Confirm ? Y
```

To see the result, use the SHOW FLASH command:

```
console> show flash
Image Index           0
Script Index          7
Flash Contents        Index   Name            Size      Date
  sumchk ok  02133    1       nis_v4.0-1B     3322310   1996-08-27-15:37:32.380
  sumchk ok  42836    3       ospf            5824      1996-08-27-16:34:24.308
  sumchk ok  28463    4       script.1        8420      1996-08-27-16:54:17.367
  sumchk ok  24778    5       mcnm_prf        1028      1996-08-27-17:12:46.335
  sumchk ok  32449    6       x25l2_prf       18650     1996-08-27-17:35:08.153
  sumchk ok  17232    7       x25l3_prf       23826     1996-08-27-17:54:21.302
```

Now that `script` has been deleted from the index, `script.1` becomes the default script.

Memory Allocation

Note that although the DELETE command marks a file as deleted and its entry is removed from the index, the space allocated to the file is not recovered.

To recover the allocated space you must delete the file from the combined image file on the load host using the MOD_FLSH command. Then you can clear flash and reload the amended image from the load host using the UPDATE command, as described in Section 3.2.5.

3.2.5 Updating Nonvolatile Memory

You can erase the contents of flash memory and load a new image or combined file into flash memory. To do this, enter the following command:

```
console> update host:file-name
```

where:

host	is the IP address or name of the remote host.
file-name	is the specification of the file to be loaded.

For example:

```
console> update 16.36.0.138:/usr/users/flash/nis_nis116.sys
 About to overwrite flash area, Confirm ? y
Is it ok to clear the flash first ? y
```

The first question is asking for confirmation that you want to proceed. Answer "Y" to continue, or "N" to abort the command.

The second question is asking if it should clear flash memory first.

If you answer "N" the system will check that the file exists and is of the correct format, before it erases flash memory and starts to load the image. You may need to adjust the TFTP timeout interval because the time taken to erase the flash memory exceeds the default TFTP timeout value. If TFTP times out the connection is lost by the time flash memory is cleared and the transfer fails.

If you answer "Y" the system clears flash memory immediately, before it makes a connection to the load host.

Note that flash memory is cleared before the load starts. If there is a failure during the transfer (for example, if there is a power failure), then the contents of flash memory is lost. If you have an MPC-II card the DECNIS has to load a new image from the load host before it can reboot. If you have an MPC-III card the effect is not so serious because only the Flash Update Area is erased. See Section 3.3 for more information about the use of flash memory on the MPC-III.

3.3 Controlling Nonvolatile Memory On the MPC–III

You can use the same commands on the MPC–III as on the MPC–III. However, before you can use them on the MPC–III, you need to do some additional configuration, as described in the next sections.

3.3.1 Nonvolatile Memory Areas

The MPC–III has 8 MB of flash memory, as against 4 MB on the MPC–II. These 8 MB are divided into two areas, as shown in Figure 3–1.

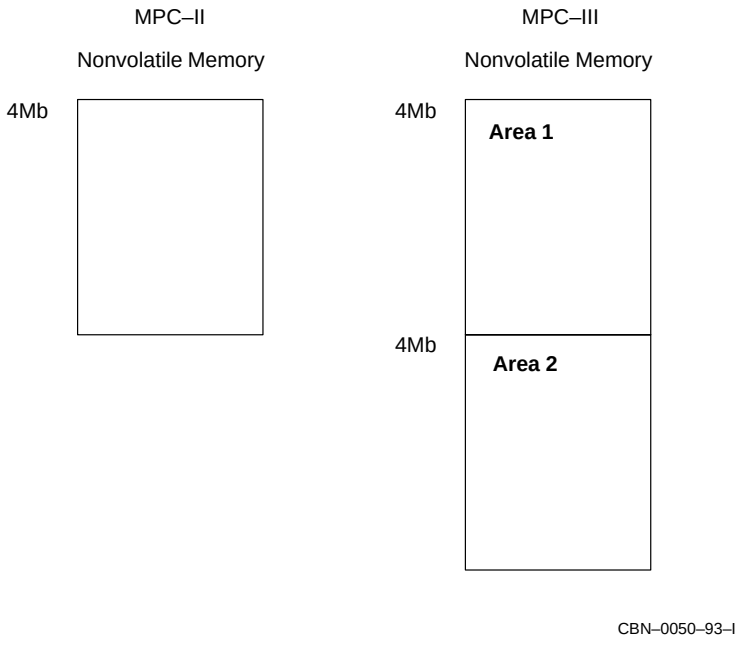
On an MPC–III, you designate the way these areas are to be used, as follows:

- You can designate one of the areas of flash memory to be the Flash Boot Area. This will be the area that the DECNIS will load from at startup.
- If you want to control files in flash memory, you must first designate an area to be used as the Flash Update Area. The commands to add, delete, set or update files only operate in the Flash Update Area.

  Once you have designated the Flash Update Area, you can use the ADD, DELETE, SET and UPDATE commands, as for the MPC–II; refer to Section 3.2.

The Flash Boot Area and Flash Update Area are set independently. They may both refer to the same area of flash memory or to different areas.

Figure 3–1 Flash Memory on MPC–II and MPC–III



3.3.2 Selecting the Flash Boot Area

With the MPC–III, the DECNIS boots from the flash memory area which has been designated as the Flash Boot Area. By default, the DECNIS will use flash Area 1 as the Flash Boot Area.

3.3.2.1 Changing the Flash Boot Area

To change the Flash Boot Area, enter the following console command:

```
console> set flash boot area-number
```

where *area-number* is the number of the area which contains the combined image you want the DECNIS to use the next time it is started.

There is an equivalent ROM console command which you can use to set the Flash Boot Area, if an error occurs during loading; refer to Section 2.15.2.

Note that, if the load from the Flash Boot Area fails, the DECNIS does **not** access the other area. If an error occurs in loading the DECNIS attempts to load from a load host.

3.3.3 Selecting the Flash Update Area

By default, the Flash Update Area is not defined.

Commands which affect the contents of flash memory only operate on the area which has been nominated as the Flash Update Area. This means that the ADD, DELETE, SET and UPDATE console commands will not work on the MPC–III until you have selected the Flash Update Area.

3.3.3.1 Designating the Flash Update Area

To designate the Flash Update Area, enter the following console command:

```
console> set flash update area-number
```

where *area-number* is the number of the area on which you want the commands to operate.

If you want to preserve the contents of the Flash Boot Area until you have set up the new image and script files, set the Flash Update Area to use the other area of flash memory.

3.3.4 Loading from the Network

When you load a CMIP file from a network load host, the image is loaded into the Flash Update Area. If the Flash Update Area is not defined, the image is loaded into Area 1.

Note that the first time you boot the DECNIS the Flash Update Area is not defined, so the combined image is loaded into Area 1, which, by default, is the Flash Boot Area.

3.3.5 Example

In this example, you wish to load a new image and configuration script into flash memory, as the existing ones are out of date. However, you also wish to keep the existing image and script in flash memory, while you test the new ones.

The original image and scripts were loaded into Area 1, which is identified as the Flash Boot Area.

To update the DECNIS you should do the following:

1. Set Area 2 to be the Flash Update Area,
2. Use the UPDATE command to load the new image and scripts into Area 2.
3. When this area has been fully configured, set Area 2 to be the Flash Boot Area.

4. Restart the DECNIS.

The original image is still safe in Area 1, so if there is any problem with the new configuration you can set the Flash Boot Area back to Area 1 and continue to re-configure Area 2.

The following steps go through this procedure in more detail:

Step 1. Viewing the Initial Configuration

The first time the DECNIS is booted from flash memory the Flash Boot Area is Area 1 by default, and the Flash Update Area is not defined. The SHOW FLASH command display is:

```
console> show flash
Flash Boot Area       1
Flash Update Area     not setup

Status of 1st Flash Area
Image Index           0
Script Index          0

Flash Contents        Index   Name            Size       Date
  sumchk ok  02133    1       nis_v4.0-1B     3322310   1996-08-27-15:29
  Flash free  850432 bytes
```

Step 2. Updating Flash With a New Image

Before you can use the Update command you must designate an area to be used as the Flash Update Area. Enter the following command:

```
console> set flash update 2
```

You can now do a network load or use the UPDATE command to load a new image or combined file:

```
console> update 16.36.0.138:/usr/users/flash/nis_nis116.sys
 About to overwrite flash area 2, Confirm ? y
Is it ok to clear the flash first ? y
Requested Flash Update
 Flash modify is in progress; it cannot be stopped. You can either observe
 the progress or type return to exit. The status can be checked using the
 Show Flash command and, if in progress, Update with no parameter returns
 to this point:

 Flash Update : erasing flash.
 Flash Update : waiting for file transfer
 Flash Update : transferring and writing to flash.............
 Flash Update : skipping the MPC-I image.........
 Flash Update : transferring and writing to flash.
 Flash Update : operation completed ok
```

Step 3. Viewing Updated Flash Memory

When the update has completed you can see that the new files have been loaded into the Flash Update Area by using the SHOW FLASH command:

```
console> show flash
Flash Boot Area       1
Flash Update Area     2
Flash Status : operation completed ok

Status of 1st Flash Area
Image Index           0
Script Index          0

Flash Contents        Index   Name            Size       Date
  sumchk ok  02133    1       nis_v4.0-1B     3322310   1996-08-27-15:29
  Flash free  850432 bytes

Status of 2nd Flash Area
Image Index           0
Script Index          0

Flash Contents        Index   Name            Size       Date
  sumchk ok  02133    1       nis_v4.0-1B     3322310   1996-08-27-15:29
  sumchk ok  32415    2       script          7416      1996-08-27-21:42
  sumchk ok  04497    3       test_script     1024      1996-07-09-12:42
  Flash free  841216 bytes
```

Step 4. Adding Profiles and Scripts

When the new image has been copied you can use the ADD command to fetch additional profiles and script files into the Flash Update Area:

```
console> add mcnm_prf node.reo.dec.com:/usr/users/flash/mcnm_prf
Requested Flash Update
 Flash modify is in progress; it cannot be stopped. You can either observe
 the progress or type return to exit. The status can be checked using the
 Show Flash command and, if in progress, Update with no parameter returns
 to this point:

 Flash Update : waiting for file transfer
 Flash Update : operation completed ok
console>
```

You can use the SHOW FLASH command to verify that the new file has been added:

```
console> show flash
Flash Boot Area       1
Flash Update Area     2
Flash Status : operation completed ok

Status of 1st Flash Area
Image Index           0
Script Index          0
```

```
Flash Contents        Index   Name            Size       Date
  sumchk ok  02133    1       nis_v4.0-1B     3322310   1996-08-27-15:29
  Flash free  850432 bytes

Status of 2nd Flash Area
Image Index           0
Script Index          0

Flash Contents        Index   Name            Size       Date
  sumchk ok  02133    1       nis_v4.0-1B     3322310   1996-08-27-15:29
  sumchk ok  32415    2       script          7416      1996-08-27-21:42
  sumchk ok  04497    3       test_script     1024      1996-07-09-12:42
  sumchk ok  24778    4       mcnm_prf        1028      1582-10-15-00:28
  Flash free  839680 bytes
console>
```

Step 5. Using the Updated Image

When the Flash Update Area has been configured with the required scripts and profiles you can set the DECNIS to boot from this flash area using this console command:

```
console> set flash boot 2
console>
```

You can now restart the DECNIS using the new configuration. For example:

```
console> load -F
Confirm, reload the system ? y

%MOPFW, attempting program load from FLASH RAM...

%MOPFW, l o a d i n g from FLASH RAM...

%MOPFW, load completed
 time           : 0 seconds
 total size     : 41 blocks
 transfer       : 0x800


%FLASH, FLASH BOOT Program
%FLASH, decompressing image from flash

%FLASH, using flash area 2
 ....................
```

If there is no valid image in the Flash Boot Area, the DECNIS will load an image from the network and save it in the current Flash Boot Area.

Step 6. Reviewing Flash Memory

After the system has reloaded, the Flash Boot Area will be set to 2, but the Flash Update Area will not be defined. In this example the first flash area still contains the original image, since it was never cleared, and the second flash area contains the new configuration which was used to reload the system.

```
console> show flash
Flash Boot Area       2
Flash Update Area     not setup

Status of 1st Flash Area
Image Index           0
Script Index          0

Flash Contents        Index   Name            Size       Date
  sumchk ok  02133    1       nis_v4.0-1B     3322310   1996-08-27-15:29
  Flash free  850432 bytes

Status of 2nd Flash Area
Image Index           0
Script Index          0

Flash Contents        Index   Name            Size       Date
  sumchk ok  02133    1       nis_v4.0-1B     3322310   1996-08-27-15:29
  sumchk ok  32415    2       script          7416      1996-08-27-21:42
  sumchk ok  04497    3       test_script     1024      1996-07-09-12:42
  sumchk ok  24778    4       mcnm_prf        1028      1582-10-15-00:28
  Flash free  839680 bytes
```

Step 7. Deleting An Index Entry From Flash Memory

We want to delete the dummy entry test_script from the index. Since the Flash Update Area has not been defined, the DELETE command will not work yet.

```
console> delete 3
Flash Update Area must be set first
```

To delete an entry, you must enter the following commands:

```
console> set flash update 2
console> delete 3
 File index 3 in flash area 2 will be marked as deleted,
 but the space cannot be recovered, Confirm ? y
Requested Flash Update

 Flash Update : operation completed ok
```

The SHOW FLASH command reveals that Flash Update Area is Area 2, and that the entry has been deleted from the index in the 2nd Flash Area.

Remember that the memory has not been recovered.

```
console> show flash
Flash Boot Area       2
Flash Update Area     2
Flash Status : operation completed ok

Status of 1st Flash Area
Image Index           0
Script Index          0

Flash Contents        Index   Name            Size       Date
  sumchk ok  02133    1       nis_v4.0-1B     3322310   1996-08-27-15:29
  Flash free  850432 bytes

Status of 2nd Flash Area
Image Index           0
Script Index          0

Flash Contents        Index   Name            Size       Date
  sumchk ok  02133    1       nis_v4.0-1B     3322310   1996-08-27-15:29
  sumchk ok  32415    2       script          7416      1996-08-27-21:42
  sumchk ok  24778    4       mcnm_prf        1028      1582-10-15-00:28
  Flash free  839680 bytes
console>
```

Step 8. Restoring The Original Image

If you want to return to using the original image, there is no need to load it from the load host, since it is still in flash memory. Simply set the Flash Boot Area to be the original area (Area 1).

```
console> set flash boot 1
```

4

Configuring SNMP on the DECNIS

4.1 Introduction

Any node in the same IP network as the DECNIS can use SNMP (Simple Network Management Protocol) to manage the DECNIS, provided the node complies with the SNMP standard as specified in RFC 1157. A node being used in this way is called a **network management station**.

The network management station can send SNMP Set requests to alter the configuration of the DECNIS, and Get requests to monitor its status.

4.2 MIBs

4.2.1 MIB Definition

An SNMP-compliant network device such as the DECNIS supports one or more Management Information Bases (MIBs). Each MIB is a collection of **objects**.

The values of these objects determine the behavior of corresponding aspects of the network device. The values can be read or altered by the network management station.

Example: The Bridge MIB is a collection of objects relevant to the bridging function of a network device. A network device which does not perform bridging would not need to support the Bridge MIB.

4.2.2 Groups and Objects

The objects in a MIB are divided into groups. Each group corresponds to a particular aspect of the network device.

Example: MIB-II has a System group, which contains objects relating to the network device as a whole, such as its physical location, the functions it performs, and so on.

A network device does not have to support all groups in a MIB. Generally, the MIB specifies which of its groups are mandatory.

4.3 Overview

Figure 4–1 shows a network management station being used to manage a DECNIS across an IP network.

Network Management Station

This is the system (host) from which network management instructions are sent to the DECNIS.

Network Management Software

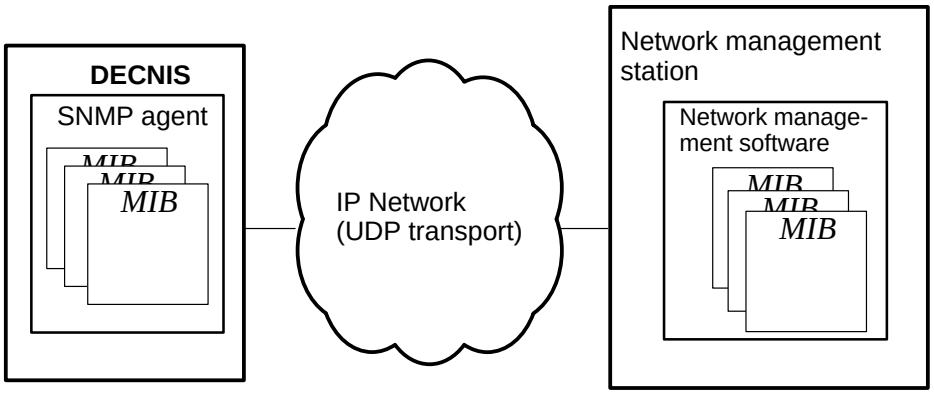
This is the SNMP-based management application running on the network management station.

Note that the MIBs supported by the DECNIS can be compiled into the management application: see the documentation for the management application for details of how to do this.

SNMP Agent

Any network device that has been configured to be manageable using SNMP must have an SNMP agent residing on it. In this case, the SNMP agent resides on the DECNIS.

Figure 4–1 Managing the DECNIS Using SNMP



CBN–0069–93–I

4.4 Access Control

4.4.1 Community Names

Each SNMP request sent to a network device contains a community name. Each network device is preconfigured with one or more community names, and the sort of access allowed for each (read-only, read-write, or no access).

If a network device receives an SNMP request containing an unknown community name, or a community name that is associated with the wrong sort of access, the network device will not respond.

4.4.2 Example

Network managers have configured a DECNIS with two community names, PUBLIC and PRIVATE. They have associated SNMP read-only access with PUBLIC, and read-write access with PRIVATE.

If the DECNIS receives an SNMP **read** request that contains the community name PRIVATE or PUBLIC, the request succeeds. If the DECNIS receives an SNMP read request that contains any other community name, the request fails.

If the DECNIS receives an SNMP **write** request that contains the community name PRIVATE, the request succeeds. If the DECNIS receives an SNMP write request that contains the community name PUBLIC, or any other community name, the request fails.

4.5 MIBs Supported by the DECNIS

4.5.1 MIBs and MIB Groups

The DECNIS supports the following MIBs and MIB groups:

- MIB-II (RFC 1213)

  MIB groups that the DECNIS supports in this MIB are:

  – system
  – interfaces
  – at
  – ip
  – icmp
  – udp
  – egp

  - transmission
  - snmp
- Bridge MIB (RFC 1493)

  MIB groups that the DECNIS supports in this MIB are:
  - dot1dBase
  - dot1dStp
  - dot1dTp
  - dot1dStatic
- FDDI MIB (RFC 1285 - draft)

  MIB groups that the DECNIS supports in this MIB are:
  - snmpFddiSMT
  - snmpFddiMAC
  - snmpFddiPORT
  - snmpFddiATTACHMENT
- DS3 MIB (RFC 1407)

  MIB groups that the DECNIS supports in this MIB are:
  - dsx3ConfigTable
  - dsx3CurrentTable
  - dsx3IntervalTable
  - dsx3TotalTable

  Note: The DECNIS provides Read-Only access to the dsx3ConfigTable
- DEC Vendor MIB elanext V2.7

  MIB groups that the DECNIS supports in this MIB are:
  - efddi

    efddiSMT
    efddiMAC
    efddiPORT
    efddiFDX
  - ebridge
  - eauth

4.5.2 Location of MIBs

The DEC Vendor MIB elanext V2.7 is supplied with the DECNIS kit. Table 4–1 gives the file name and location for each type of host on which the kit can be installed.

Table 4–1 Location of DEC Vendor MIB

Host	File Specification
OpenVMS	SYS$HELP:DEC_ELAN_MIB.V27_TXT
DIGITAL UNIX	/usr/lib/dnet/dec_elan_mib.v27_txt
Windows 95/NT PCs	C:\ *install-dir* \ DOCS \ ELANMIB.TXT

where *install-dir* is the directory where the PC kit is installed.

You can access MIB-II, Bridge MIB, and FDDI MIB through ftp from the repository at nic.ddn.mil in the rfc directory. Check the rfc-index file for details of the MIBs available.

4.6 Traps

4.6.1 Definition

SNMP traps are messages that are sent by a network device when one of a prescribed list of changes in its operation occurs. The traps can be received by one or more systems in the same IP network as the originating device.

4.6.2 Traps Supported by the DECNIS

The DECNIS sends the traps specified in Table 4–2. Note that all traps include the IP address of the DECNIS.

Table 4–2 Traps Sent by the DECNIS

Trap	Meaning
coldStart	The DECNIS has rebooted.
warmStart	The SNMP entity of the DECNIS has been enabled.
linkUp	A data link has changed to state "up". The trap includes the SNMP MIB-II index number of the interface.

(continued on next page)

Table 4–2 (Cont.) Traps Sent by the DECNIS

Trap	Meaning
linkDown	A data link has changed to state "down". The trap includes the SNMP MIB-II index number of the interface.
authenticationFailure	The DECNIS has received an SNMP request that specifies an incorrect community name.
egpNeighborLoss	The DECNIS has marked an EGP peer relationship down. The trap includes the IP address of the EGP neighbor.

4.7 Configuring the DECNIS to Be Manageable Using SNMP

4.7.1 Introduction

This section describes the NCL commands required to allow the DECNIS to respond to SNMP requests from a management station.

It also describes how to use NCL and SNMP to configure the DECNIS to send traps.

You can issue these commands on a running system. You can also use the DECNIS configurator (text-based or Windows) to set up SNMP management. In this case, you must reboot the DECNIS before the changes will take effect.

4.7.2 Configuring the DECNIS to Respond to SNMP Requests

Follow these steps to configure the DECNIS to respond to SNMP requests from a remote management station:

1. Enable the SNMP entity:

   ```
   NCL> ENABLE NODE decnis SNMP
   ```

   (The SNMP entity is created automatically when the DECNIS is booted.)

2. Create COMMUNITY entities to control SNMP access to the DECNIS.

   The name of each COMMUNITY entity forms (part of) the community name. Each COMMUNITY entity specifies the type of SNMP access allowed to management stations sending that community name.

   Create COMMUNITY entities as follows:

   – Create the entity:

     ```
     NCL> CREATE NODE decnis SNMP COMMUNITY name
     ```

name is the name you use to refer to this entity. It also forms the first part of the community name that the DECNIS will try to validate when it receives an SNMP request.

Note that this name is case-sensitive; also, although nonalphanumeric characters are permitted in this name, DIGITAL recommends that you use only alphanumeric characters. (Note also that you cannot use a wildcard (*) in an NCL command to show all COMMUNITY entities on the DECNIS, since * is a valid name for a COMMUNITY entity.)

– If you wish, set a second (nonreadable) part of the community name for this COMMUNITY entity by entering the following command:

```
NCL> SET NODE decnis SNMP COMMUNITY name -
_NCL> PRIVATE NAME private-name
```

private-name, when appended to the name of the COMMUNITY entity, forms the community name that the DECNIS will try to validate when it receives an SNMP request. Note that the PRIVATE NAME characteristic is write-only, and can contain nonalphanumeric characters. It is case-sensitive.

If you do not set this characteristic, it remains at its default (null).

– Specify the type of access for this community:

```
NCL> SET NODE decnis SNMP COMMUNITY name ACCESS access
```

access is READ-ONLY or READ-WRITE.

3. Enable each COMMUNITY entity:

```
NCL> ENABLE NODE decnis SNMP COMMUNITY name
```

4.7.3 Example

When the DECNIS receives an SNMP request, it checks the community name specified in the request, and permits or denies access accordingly.

You want to configure the DECNIS with two community names: one for read-only access, and one for read-write access.

1. Enable the SNMP entity:

```
NCL> ENABLE NODE decnis SNMP
```

2. Create a COMMUNITY entity for read-only access, called PUBLIC:

```
NCL> CREATE NODE decnis SNMP COMMUNITY public
```

3. Set the access for this community to read-only:

```
NCL> SET NODE decnis SNMP COMMUNITY public -
_NCL> ACCESS read-only
```

4. Enable the entity:

```
NCL> ENABLE NODE decnis SNMP COMMUNITY public
```

5. Now create a community entity for read-write access, called PRIVATE:

```
NCL> CREATE NODE decnis SNMP COMMUNITY private
```

6. For security, set a second, nonreadable part of the community name:

```
NCL> SET NODE decnis SNMP COMMUNITY private-
_NCL> PRIVATE NAME _secret
```

7. Set the access for this community to read-write:

```
NCL> SET NODE decnis SNMP COMMUNITY private -
_NCL> ACCESS read-write
```

8. Enable the entity:

```
NCL> ENABLE NODE decnis SNMP COMMUNITY private
```

4.7.3.1 Result

If the DECNIS receives an SNMP read request containing the community name PUBLIC or PRIVATE_SECRET, access is allowed. A read request bearing any other community name will not succeed.

If the DECNIS receives an SNMP write request containing the community name PRIVATE_SECRET, access is allowed. A write request bearing any other community name will not succeed.

4.7.4 Using NCL to Configure the DECNIS to Send Traps

Note that use of NCL and SNMP are equivalent, and affect the same internal DECNIS database. If you set up trap destinations using NCL, you can use SNMP to show these destinations.

Follow these steps to configure the DECNIS to send traps:

1. Ensure that the SNMP entity is enabled:

```
NCL> ENABLE NODE decnis SNMP
```

(The SNMP entity is created automatically when the DECNIS is booted.)

2. Specify the IP addresses of the systems to which the DECNIS will send traps, together with the community name to be included in traps:

```
NCL> SET NODE decnis SNMP TRAP SINKS -
_NCL> {(DESTINATION= a.b.c.d, COMMUNITY = name), ...}
```

a.b.c.d is the IP address of the system to which the DECNIS will send traps, and *name* is the community name to be included in all traps sent by the DECNIS. Note that this community name will be the same irrespective of the trap destination. If you enter different community names in the above command, only the last one you enter will be included.

You may enter up to four IP address/community name combinations, but all community names except the last will be ignored.

If you do not wish traps from the DECNIS to include a community name, specify a null community name in the final IP address:

```
NCL> SET NODE decnis SNMP TRAP SINKS -
_NCL> {..., (DESTINATION= a.b.c.d, COMMUNITY = "")}
```

You can use the ADD NCL command to add an IP address/community name entry to an existing set; the REMOVE command removes an IP address/community name entry from an existing set.

3. If you wish the DECNIS to send authenticationFailure traps, you must issue the following command:

```
NCL> SET NODE decnis SNMP -
_NCL> GENERATE AUTHENTICATION FAILURE EVENTS true
```

By default, the DECNIS does not send authenticationFailure traps.

4.7.5 Using SNMP to Configure the DECNIS to Send Traps

Note that use of NCL and SNMP are equivalent, and affect the same internal DECNIS database. If you set up trap destinations using SNMP, you can use NCL to show these destinations.

Table 4–3 shows the MIB variables involved in configuring the DECNIS to send traps. How you read and set these variables depends on the management application software running on the network management station.

Table 4–3 SNMP Traps: MIB Variables

MIB Group	MIB Table	MIB Variable	Comments
		DEC Vendor MIB elanext V2.7	
eauth	(scalar)	eauthTrapCommunity	Set the value of TrapCommunity_0 to the community name to be included in the traps sent by the DECNIS.
eauth	TrapUserTable	eauthTrapUserAddr	Create a TrapUserAddr_*a.b.c.d*, with a value of *a.b.c.d*, where *a.b.c.d* is the IP address to which traps are to be sent.
		eauthTrapUserStatus	Set this to "invalid" to delete the relevant trap destination.
		MIB-II	
SNMP	(scalar)	snmpEnableAuthenTraps	Set this to "enabled" if you want the DECNIS to generate authentication-failure traps.

4.8 Using SNMP to Manage the DECNIS

Section 11.20 describes the use of SNMP to manage data links on the DECNIS.

Section 17.26 describes the use of SNMP to manage the IP routing functions of the DECNIS.

Section 27.18 describes the use of SNMP to manage the bridge functions of the DECNIS.

Note

The above sections do not describe in detail how to use SNMP. They assume that you are familiar with and have some experience of using SNMP.

For more information, refer to the relevant RFCs, and the documentation for the network management software you are using.

4.9 clearVISN

clearVISN is DIGITAL's policy-based element manager. It consists of a suite of SNMP-based network management applications for managing devices within a network.

Because of its extensive SNMP and MIB support, the DECNIS can be managed using the clearVISN Router Manager. The clearVISN Router Manager is an SNMP router management application with multivendor support for routers. Its features include:

- **Device discovery**. When given an IP address, clearVISN discovers the appropriate IP device.
- **Path Tracing**. You can trace paths between IP devices in the network, to help find problems in network links.
- **Performance Monitoring**. clearVISN uses MIB information to determine bandwidth utilization on a per interface or per protocol basis, and automatically create a utilization graph.
- **Alarms and Reports**. You can set thresholds for automatic alarms and to show network events, for example, percentage line utilization thresholds. You can also generate periodic reports according to predetermined criteria.
- **Configuration Manager**. This provides management of multiple device configuration files, including DECNIS configuration and image files.
- **Telnet capability**. This allows you to connect to the DECNIS console and the NCL command line interface to manage running DECNIS systems.
- **Policy-based management**. This allows procedures to be set up to simplify the management of complex network devices and technologies.

Refer to the clearVISN documentation for more information.

5

Setting Up Event Logging on the DECNIS

5.1 Event Logging on the DECNIS

This chapter describes how to set up event logging on the DECNIS, using NCL commands.

You can use the clearVISN DECNIS configurator to set up event logging. Refer to the configurator on line help, and the *clearVISN DECNIS Configurator User Guide* for more information.

5.1.1 Event Messages

A list of the event messages sent by the DECNIS is supplied on line in a text file. The locations of the event message files are as follows:

Load Host	Location
Windows 95 and Windows NT PCs	*install-directory* \ DOCS \ EVMSG41.TXT
OpenVMS	SYS$HELP:NIS$EVENTS.TXT
DIGITAL UNIX	/usr/lib/dnet/nis_event.txt

5.1.2 NCL Commands

All tasks performed using NCL commands assume that you have:

- Logged on to a suitable host system or logged on to a console terminal.
- Started NCL and, if managing a remote DECNIS, set default to the DECNIS as described in Section 1.4.4 or Section 2.13.5

5.1.2.1 Entering Event Logging Commands in the User NCL Script Files

Note that you must insert the event dispatcher commands PASS, BLOCK, IGNORE and SET into the SET user NCL script file.

An example of such a command is:

```
PASS EVENT DISPATCHER OUTBOUND STREAM stream_1
```

5.2 Outbound Event Streams

Outbound event streams allow events to be passed from the DECNIS to an event sink.

5.2.1 Example

Figure 5–1 illustrates a DECNIS on which three event streams have been set up, to allow events to be passed to two event sinks.

5.3 Event Sinks

5.3.1 Requirement to Set Up Event Sinks

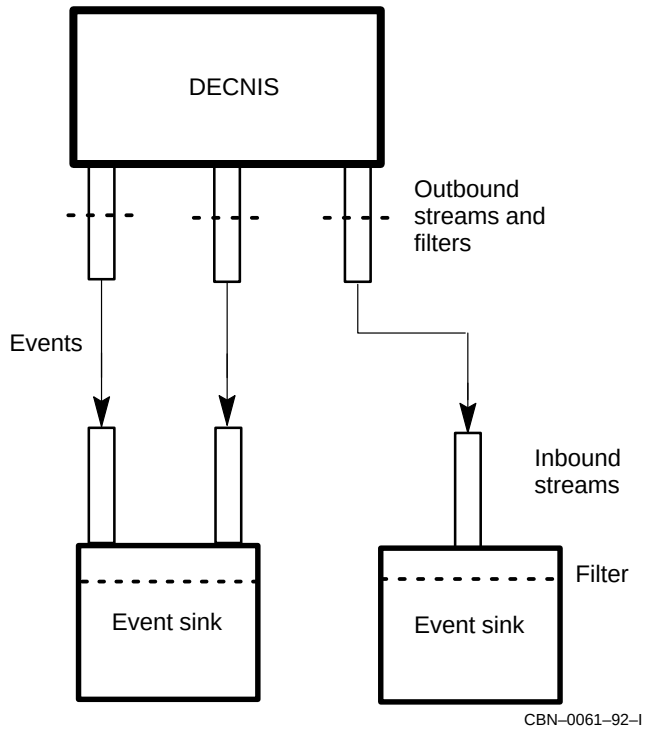
You must set up the event sink on the sink node before you can use event logging. Appendix C contains the NCL commands required to set up event sinks. For more information, refer to the documentation for your sink node.

5.3.2 Event Sink Nodes

You can use any of the following types of node as an event sink:

- A DECnet Phase IV system.
- A DECnet-Plus system.
- A Windows 95/NT system with the clearVISN DECNIS configurator installed. This system can be set up as an IP sink node.

Figure 5–1 Event Streams on the DECNIS



5.3.3 Structure of Tower Set for an Event Sink

If you are using a DECnet Phase IV or DECnet-Plus system as an event sink, then you must provide the tower set information that specifies how the DECNIS can reach the event sink. You can do this in two ways:

- By entering the node name of the event sink in the Known Towers database, as described in Section 1.12.
- By specifying the event sink as a tower set, as described in Section 5.3.3.1.

5.3.3.1 Specifying the Tower Set

.

A tower set consists of one or more towers. An event sink tower is:

```
([DNA_CMIP-MEN], [DNA_SESSIONCONTROLVn, NUMBER = 82],
[DNA_NSP], [DNA_OSINETWORK, NSAP-address])
```

n	is 3 if the event sink is a DECnet-Plus (formerly known as DECnet /OSI) node. is 2 if the event sink is a Phase IV node.
NSAP-address	is the NSAP address of the event sink.

5.3.3.2 Example

Suppose the DECnet-Plus node that is to act as the event sink for the DECNIS has the following two NSAP addresses:

37:12345:02-00:AA-02-14-78-66-11:20
37:12345:02-00:AA-02-14-78-66-10:20

The tower set representing this event sink consists of two towers, as follows:

```
SET NODE decnis EVENT DISPATCHER -
OUTBOUND STREAM stream-name SINK ADDRESS -
{([DNA_CMIP-MEN], [DNA_SESSIONCONTROLV3, NUMBER = 82], -
[DNA_NSP], -
[DNA_OSINETWORK, 37:12345:02-00:AA-02-14-78-66-11:20]), -
([DNA_CMIP-MEN], [DNA_SESSIONCONTROLV3, NUMBER = 82], -
[DNA_NSP], [DNA_OSINETWORK, 37:12345:02-00:AA-02-14-78-66-10:20]) -
}
```

where *decnis* is the name of the DECNIS.

5.4 Setting Up Event Logging

This section describes how to set up an outbound event stream on the DECNIS.

5.4.1 Procedure

Follow these steps to set up event logging on the DECNIS.

1. Ensure that the EVENT DISPATCHER entity exists and is enabled:

   ```
   NCL> CREATE NODE decnis EVENT DISPATCHER
   NCL> ENABLE NODE decnis EVENT DISPATCHER
   ```

   where *decnis* is the name of the DECNIS.

2. Create the outbound event stream:

```
NCL> CREATE NODE decnis EVENT DISPATCHER -
_NCL> OUTBOUND STREAM stream-name
```

where *stream-name* is the name of the outbound event stream.

3. Specify the sink to be used to log the events from this event stream.

If the sink is a DECnet system, you can specify it as a node name or a tower set. If the sink is an IP system, you specify it as an IP address in the required format.

As a Node Name

To specify the sink as a node name, issue the following command:

```
NCL> SET NODE decnis EVENT DISPATCHER -
_NCL> OUTBOUND STREAM stream-name SINK NODE sink-node
```

where *sink-node* is the node name of the sink node as defined in the Known Towers database (see Section 1.12).

As a Tower Set

To specify the sink as a tower set, issue the following command:

```
NCL> SET NODE decnis EVENT DISPATCHER -
_NCL> OUTBOUND STREAM stream-name SINK ADDRESS {(tower1), (tower2), ...}
```

where *tower1* and *tower2* are towers that describe the protocols and NSAP addresses used to communicate with the event sink.

As an IP Address

To specify the sink as an IP address, issue the following command:

```
NCL> SET EVENT DISPATCHER OUTBOUND STREAM stream-name -
_NCL> SINK NODE "\ip-address"
```

where *ip-address* is the IP address of the sink node. Note that you must not supply the SINK ADDRESS attribute for IP sink nodes.

Example

To set up a node with the IP address 63.18.124.12 as a sink node, you could enter the following command:

```
NCL> SET EVENT DISPATCHER OUTBOUND STREAM evstream1
_NCL> SINK NODE "\63.18.124.12"
```

4. Enable the OUTBOUND STREAM entity:

```
NCL> ENABLE NODE decnis EVENT DISPATCHER -
_NCL> OUTBOUND STREAM stream-name
```

5.5 Disconnecting the DECNIS from the Event Sink

5.5.1 Introduction

This section describes how to break the connection between the outbound event stream and the event sink.

5.5.2 Breaking the Connection Immediately

Issue the following command to destroy immediately the connection between an outbound event stream and its event sink:

```
NCL> DISCONNECT NODE decnis EVENT DISPATCHER OUTBOUND STREAM -
_NCL> stream-name
```

This will cause events in transit to be lost. You should issue this command if you have problems with your sink node and wish to specify a new sink node.

5.5.3 Breaking the Connection in an Orderly Manner

Issue the following command to achieve an orderly shutdown of the connection between an outbound event stream and its event sink:

```
NCL> SHUTDOWN NODE decnis EVENT DISPATCHER OUTBOUND STREAM stream-name
```

This breaks the connection for the outbound stream once all events in transit have been received. However, if the connection is faulty, the command may take a long time to execute.

5.5.4 Reestablishing the Connection

The DECNIS will attempt to reestablish the connection automatically when the CONNECT RETRY TIMER expires (see Section 5.7).

You can manually reestablish the connection by issuing the following command:

```
NCL> CONNECT NODE decnis EVENT DISPATCHER OUTBOUND STREAM stream-name
```

5.6 Disabling Event Streams

5.6.1 Introduction

Disable an event stream to prevent any events from being sent from the event stream to the event sink. The DECNIS can still send events to an event sink by using another event stream.

5.6.2 Procedure

Issue the following command to disable an event stream:

```
NCL> DISABLE NODE decnis EVENT DISPATCHER -
_NCL> OUTBOUND STREAM stream-name METHOD = method
```

where *method* is one of:

- ABORT

  The connection is destroyed at once, and all events in transit are lost. This is similar to using the DISCONNECT command (see Section 5.5.2).

- ORDERLY

  This is the default method of disabling an event stream. It allows all events in transit to be received by the sink before the connection is broken (see the SHUTDOWN command in Section 5.5.3). However, this directive may take a very long time to execute, particularly if the event sink is faulty.

To prevent immediately the DECNIS from sending any events to any event sink, use the following command:

```
NCL> DISABLE NODE decnis EVENT DISPATCHER -
_NCL> OUTBOUND STREAM * METHOD = ABORT
```

5.7 Connection Timers

5.7.1 Introduction

The connection between the event source and the sink is controlled by two timers, whose values can be changed as required.

5.7.2 Connect Retry Timer

This controls how often the outbound stream attempts to make connections to the sink.

Note

The CONNECT TIMER ENABLED characteristic of the OUTBOUND STREAM entity must be set to TRUE for this timer to operate. If it is set to FALSE, the outbound event stream will not make connection retries.

5.7.2.1 Changing the Connect Retry Timer

Set the connect retry timer as follows:

```
NCL> SET NODE decnis EVENT DISPATCHER OUTBOUND STREAM stream-name -
_NCL> CONNECT RETRY TIMER n
```

where:

decnis	is the name of the DECNIS.
n	is a decimal number between 1 and 65,535, and specifies the number of seconds to wait between attempts to create connections. The default is 120 seconds.

5.7.3 Disconnect Timer

This controls the length of time to wait before disconnecting idle connections.

5.7.3.1 Setting the Disconnect Timer

Set the disconnect timer as follows:

```
NCL> SET NODE decnis EVENT DISPATCHER OUTBOUND STREAM stream-name -
_NCL> DISCONNECT TIMER n
```

where:

decnis	is the name of the DECNIS.
n	is a decimal number greater than or equal to 1. If no events are logged during this number of seconds, the connection is dropped. Disable this timer by using a value of 0; the connection will then not be dropped automatically.

5.8 Event Filtering on the DECNIS

5.8.1 Introduction

You can set up specific, global, and catchall filters to determine which events generated by the DECNIS get sent to the event sink.

5.8.2 Specific Filters

Specific filters specify the following:

- A specific named entity (for example, a specific routing circuit on the DECNIS).
- An event that can be generated by that entity.
- Whether the event is to be passed, blocked or ignored.

5.8.3 Global Filters

Global filters specify the following:

- An entity class (for example, all routing circuits on the DECNIS).
- An event that can be generated by that class of entity.
- Whether the event is to be passed, blocked or ignored.

5.8.4 Catchall Filter

A catchall filter specifies whether the events not specified in a specific or global filter should be passed or blocked.

5.8.5 Operation of Event Filters

Figure 5–2 illustrates how event filters operate on the DECNIS.

5.8.6 Recommendation

You can also set up filters on event sinks. However, DIGITAL recommends that you set up event sinks so that all events are received, and all filtering is done on the outbound stream. This reduces unnecessary network traffic and wasted resources for events that will be discarded at the sink.

5.9 Events Blocked by Default

5.9.1 List of Blocked Events

By default, global filters are set to block the events shown in Table 5–1 on all DECNIS outbound streams.

Figure 5–2 Operation of Event Filters in the Outbound Stream

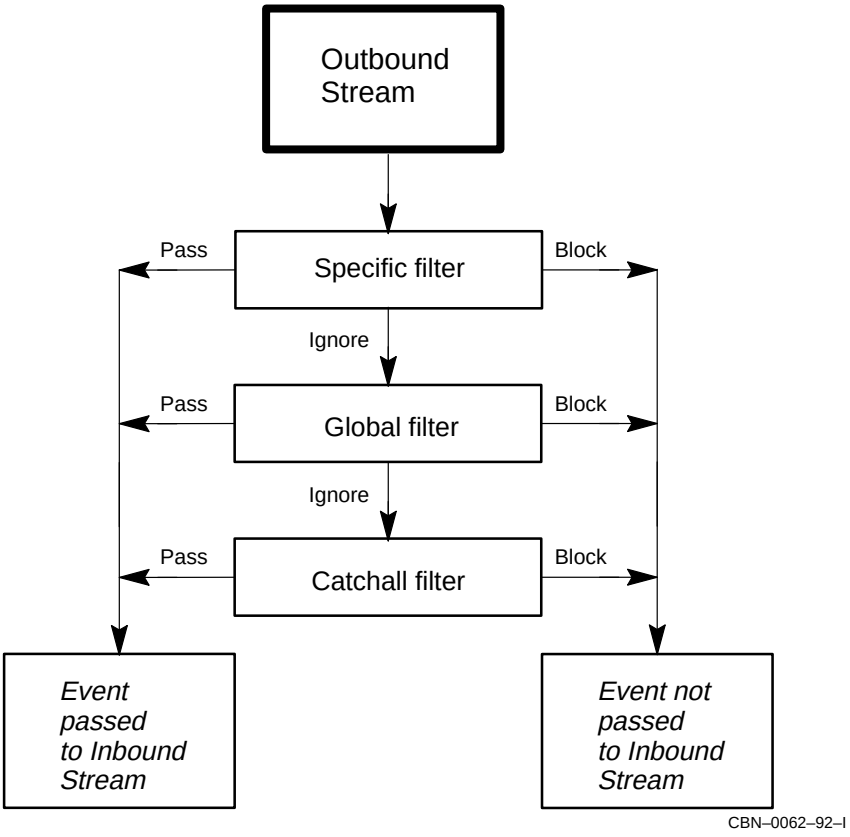


Table 5–1 DECNIS Events Blocked by Default

Entity Class	Event Name
ROUTING	Address Unreachable PDU Discard
	Aged PDU Discard
	IP Address Unreachable Packet Discard
	IP Source Address Error Packet Discard
	PhaseIV Translation Failure
ROUTING CIRCUIT	Adjacency State Change
	ID Reachability Change
	Initialization Failure
CSMA-CD STATION	Unrecognized Multicast Destination PDU
FDDI STATION	Unrecognized Multicast PDU Destination
NSP LOCAL NSAP REMOTE NSAP	Remote Protocol Error

5.9.2 Overriding Default Blocking

If you want to pass some or all of these events, insert the appropriate NCL commands in the NIS_*client-name*_EXTRA_SET.NCL file.

You can override the blocks entirely by setting up global filters, as described in Section 5.11.4, or you can override the blocks for specific instances of the entity, as described in Section 5.10.4.

5.10 Setting Up Specific Event Filters

5.10.1 Introduction

This section describes how to set up specific filters for events. A specific filter passes or blocks events from a particular entity instance, for example, a particular routing circuit.

5.10.2 To Block Events

Issue this command to set up a specific filter to block events:

```
NCL> BLOCK NODE decnis EVENT DISPATCHER OUTBOUND STREAM -
_NCL> stream-name SPECIFIC FILTER = -
_NCL> ((entity-instance), event)
```

where:

decnis	is the name of the DECNIS.
entity-instance	is the name of the entity from which events are to be blocked.
event	is the name of the event to be blocked.

5.10.3 Example: Blocking

To prevent the Circuit Change event on routing circuit Pepper from being sent to the sink, irrespective of any global or catchall filters, issue the following command:

```
NCL> BLOCK NODE decnis EVENT DISPATCHER OUTBOUND STREAM -
_NCL> stream-name SPECIFIC FILTER = -
_NCL> ((NODE decnis ROUTING CIRCUIT pepper), circuit change)
```

5.10.4 To Pass Events

Issue this command to set up a specific filter to pass events:

```
NCL> PASS NODE decnis EVENT DISPATCHER OUTBOUND STREAM -
_NCL> stream-name SPECIFIC FILTER = -
_NCL> ((entity-instance), event)
```

where:

decnis	is the name of the DECNIS.
entity-instance	is the name of the entity from which events are to be passed.
event	is the name of the event to be passed.

5.10.5 Example: Passing

To allow the Circuit Change event on routing circuit Pepper to be sent to the sink, irrespective of any global or catchall filters, issue the following command:

```
NCL> PASS NODE decnis EVENT DISPATCHER OUTBOUND STREAM -
_NCL> stream-name SPECIFIC FILTER = -
_NCL> ((NODE decnis ROUTING CIRCUIT pepper), circuit change)
```

5.11 Setting Up Global Event Filters

5.11.1 Introduction

This section describes how to set up global filters for events. A global filter passes or blocks events from all instances of a particular entity class, for example, all routing circuits.

5.11.2 To Block Events

Issue this command to set up a global filter to block events:

```
NCL> BLOCK NODE decnis EVENT DISPATCHER OUTBOUND STREAM -
_NCL> stream-name GLOBAL FILTER = -
_NCL> ((entity-class), event)
```

where:

decnis	is the name of the DECNIS.
entity-class	is the name of the entity class from which events are to be blocked.
event	is the name of the event to be blocked.

5.11.3 Example: Blocking

To prevent the Circuit Change event on all routing circuits from being sent to the sink, irrespective of any catchall filter, issue the following command:

```
NCL> BLOCK NODE decnis EVENT DISPATCHER OUTBOUND STREAM -
_NCL> stream-name GLOBAL FILTER = -
_NCL> ((NODE, ROUTING, CIRCUIT), circuit change)
```

Note that Circuit Change events would still be logged for any routing circuits on which a specific filter had been set to pass Circuit Change events.

5.11.4 To Pass Events

Issue this command to set up a global filter to pass events:

```
NCL> PASS NODE decnis EVENT DISPATCHER OUTBOUND STREAM -
_NCL> stream-name GLOBAL FILTER = -
_NCL> ((entity-class), event)
```

where:

decnis	is the name of the DECNIS.
entity-class	is the name of the entity class from which events are to be passed.
event	is the name of the event to be passed.

5.11.5 Example: Passing

To allow the Circuit Change event on all routing circuits to be sent to the sink, irrespective of any catchall filter, issue the following command:

```
NCL> PASS NODE decnis EVENT DISPATCHER OUTBOUND STREAM -
_NCL> stream-name GLOBAL FILTER = -
_NCL> ((NODE, ROUTING, CIRCUIT), circuit change)
```

Note that Circuit Change events would not be logged for any routing circuits on which a specific filter had been set to block Circuit Change events.

5.12 Setting Up a Catchall Event Filter

5.12.1 Introduction

This section describes how to set up a catchall filter for events. A catchall filter passes or blocks all events from a particular outbound event stream.

5.12.2 To Block Events

Issue this command to set up a catchall filter to block all events:

```
NCL> SET NODE decnis EVENT DISPATCHER -
_NCL> OUTBOUND STREAM stream-name CATCH ALL FILTER BLOCK
```

where:

decnis	is the name of the DECNIS.
stream-name	is the name of the outbound event stream.

Note that events would still be logged if any specific or global filters had been set to pass them.

5.12.3 To Pass Events

Issue this command to set up a catchall filter to pass events:

```
NCL> SET NODE decnis EVENT DISPATCHER -
_NCL> OUTBOUND STREAM stream-name CATCH ALL FILTER PASS
```

where:

decnis	is the name of the DECNIS.
stream-name	is the name of the outbound event stream.

Note that events would still be blocked if any specific or global filters had been set to block them.

5.13 Removing Specific and Global Event Filters

5.13.1 Introduction

This section describes how to use the IGNORE command to remove specific or global event filters.

5.13.2 Removing Specific Event Filters

Issue the following command to remove a specific event filter:

```
NCL> IGNORE NODE decnis EVENT DISPATCHER OUTBOUND STREAM -
_NCL> stream-name SPECIFIC FILTER = -
_NCL> ((entity-instance), event)
```

where:

decnis	is the name of the DECNIS.
entity-instance	is the name of the entity specified in the specific filter.
event	is the name of the event specified in the specific filter.

Example

If you have created a specific filter that blocks the Adjacency State Change event from routing circuit Pepper, use the following command to delete this specific filter:

```
NCL> IGNORE NODE decnis EVENT DISPATCHER OUTBOUND STREAM -
_NCL> stream-name SPECIFIC FILTER = -
_NCL> ((NODE decnis ROUTING CIRCUIT pepper), adjacency state change)
```

Events from routing circuit Pepper will now be passed or blocked according to any global filtering for routing circuits (see Section 5.8.3).

5.13.3 Removing Global Event Filters

Issue the following command to remove a global event filter:

```
NCL> IGNORE NODE decnis EVENT DISPATCHER OUTBOUND STREAM -
_NCL> stream-name GLOBAL FILTER = -
_NCL> ((entity-class), event)
```

where:

decnis	is the name of the DECNIS.
entity-class	is the name of the entity class specified in the global filter.
event	is the name of the event specified in the global filter.

Example

If you have created a global filter that blocks the Adjacency State Change event from all routing circuits, use the following command to delete this global filter:

```
NCL> IGNORE NODE decnis EVENT DISPATCHER OUTBOUND STREAM -
_NCL> stream-name GLOBAL FILTER = -
_NCL> ((NODE, ROUTING, CIRCUIT), adjacency state change)
```

Events from routing circuit Pepper will now be passed or blocked according to the catchall filter for this event stream (see Section 5.8.4), unless there is an appropriate specific filter.

For example, if there is a specific filter that passes the Adjacency State Change event on routing circuit Pepper, this event will always be passed, irrespective of any global or catchall filtering.

5.14 Testing Event Streams and Filters

5.14.1 Introduction: TESTEVENT Command

Use the TESTEVENT command to check whether a particular event from a particular entity will be passed or blocked, and to see the filter type used to pass or block it.

5.14.2 Command Structure

The TESTEVENT command is as follows:

```
NCL> TESTEVENT NODE decnis EVENT DISPATCHER -
_NCL> OUTBOUND STREAM stream-name EVENT = -
_NCL> ((entity-instance), event)
```

where:

decnis	is the name of the DECNIS.
entity-instance	is the name of the entity to be tested.
event	is the name of the event to be tested.

5.14.3 Example

Issue the following command to check what happens to the Circuit Change event on routing circuit Pepper:

```
NCL> TESTEVENT NODE decnis EVENT DISPATCHER -
_NCL> OUTBOUND STREAM stream-name EVENT = -
_NCL> ((NODE decnis ROUTING CIRCUIT pepper), circuit change)
```

If this event is blocked by a specific filter, the command will return the following information:

```
Action = BLOCK
Type = SPECIFIC
```

6

System-Level Management

6.1 Introduction

This chapter describes those aspects of the DECNIS that relate to the system as a whole. These are:

- System-level security
- Assigning a name to the DECNIS
- Setting the system time on the DECNIS
- The system group of MIB-II

6.2 Security for NCL

6.2.1 Need for Security

Security for the use of NCL is required to prevent unauthorized changes to the DECNIS.

6.2.2 NCL Security Mechanism

In order to restrict the use of NCL commands that alter the system configuration, you need to create a user name and password on the management listener, OBJ_19. These are sometimes referred to as the network management user name and password.

Note that you cannot restrict the use of the NCL SHOW command.

6.2.3 Setting Up Security Within the Configurators

The DECNIS text-based configurator requires you to set up the network management user name and password in the Configuration Options section.

The clearVISN DECNIS configurator allows, but does not require you to set up a network management user name and password. If you wish, you can set up the user name and password on the Security tab page under the **System** button on the Main Navigation window.

6.2.4 Setting Up the User Name and Password using NCL Commands

To set up or change the user name and password using NCL commands, enter the following:

```
NCL> SET NODE decnis SESSION CONTROL APPLICATION OBJ_19 -
_NCL> USER NAME username, PASSWORD password
```

where: *decnis* is the name of the DECNIS and *username* and *password* are the user name and password that users must supply when they enter NCL commands on the DECNIS.

6.2.5 Result of Creating the NCL User Name and Password

To use NCL on the DECNIS, users will need to enter the user name and password in the usual way for their operating system. Refer to Section 1.8.1 for examples.

In addition, if you have not set up a console password, you must supply the network management password when you use Telnet to connect to the DECNIS console.

6.3 Security for the Common Trace Facility

6.3.1 Need for Security

Security for the Common Trace Facility (CTF) is required to prevent unauthorized users from monitoring the data passing through the DECNIS.

6.3.2 CTF Security Mechanism

Access to the Common Trace Facility (CTF) is restricted by creating a user name and password on the CTF listener, OBJ_54. You supply this user name and password when you use the DECNIS configurator (text-based or Windows) to configure the DECNIS.

6.3.3 Changing the User Name and Password for CTF

To set up or change the user name and password on the CTF listener, using NCL commands, enter the following:

```
NCL> SET NODE decnis SESSION CONTROL APPLICATION obj_54 -
_NCL> USER NAME username, PASSWORD password
```

where: *decnis* is the name of the DECNIS and *username* and *password* are the user name and password that users must quote when they use CTF on the DECNIS.

Result

To use CTF on the DECNIS, users will need to quote the user name and password in the usual way for their operating system.

6.3.4 Example

To start CTF on a node called ORG:.SOUTH.SALES, which has a user name SMITH and password SECRET, specifying a tracepoint "HDLC LINK THIS-LINK", enter:

```
$ TRACE START org:.south.sales"smith secret"::"HDLC LINK this-link"
```

6.4 Preventing Unauthorized Loading

6.4.1 Need for Security

By default, any user on an adjacent system can send a LOAD command to the DECNIS, optionally specifying a new system image.

You can set up a verification value (password) on the DECNIS. This will ensure that only those users who supply this verification will be permitted to issue a LOAD command to the DECNIS.

6.4.2 Setting a MOP Verification Value

Set up a MOP verification value as follows:

NCL> SET NODE *decnis* MOP VERIFICATION *value*

where *decnis* is the name of the DECNIS, and *value* is an octet string of 8 bytes. An example verification value is %X8877665544332211.

6.4.3 Do Not Include Commands in NCL Script File

The MOP verification value is stored in nonvolatile memory on the DECNIS, and does not change when the DECNIS is rebooted. You should not add this command to NIS_*mop-client-name*_EXTRA_SET.NCL.

Clearing Nonvolatile Memory

To clear the nonvolatile memory of the DECNIS, power up the hardware with the Dump button held in. See the *Installation and Service Manual*.

6.5 Assigning a Name to the DECNIS

By default, when the DECNIS first boots, it does not know its name. Any events logged from the DECNIS will appear as logged from node 0.

After the DECNIS is first booted, enter the following command to place the DECNIS name in its nonvolatile memory:

```
NCL> RENAME NODE phase-iv-name NEW NAME decnis
```

where *phase-iv-name* is the Phase IV name for the DECNIS, and *decnis* is its DECdns name (including the namespace name) or local namespace name.

6.6 Security for SNMP

6.6.1 Security for SNMP Requests

You can use any of the following to set up community names to validate SNMP Get and Set commands sent to the DECNIS:

- The DECNIS text-based configurator
- The clearVISN DECNIS configurator
- NCL commands

The DECNIS will not respond to SNMP requests that contain an incorrect community name.

Section 4.7 describes how to set up community names.

6.6.2 Security for Traps

You can use any of the following to set up a community name that will be sent in traps from the DECNIS:

- The DECNIS text-based configurator
- The clearVISN DECNIS configurator
- NCL commands
- SNMP requests

The management stations to which the DECNIS traps are sent may then use this community name to reject or accept the trap.

Section 4.7 describes how to set up a community name in traps.

6.7 Secure Connections

Secure Connections is a utility that enables you to set up filtering to permit or deny network connect requests to or between network domains. Secure Connections can filter connect requests for IP, DECnet, and OSI.

Secure Connections works by enabling you to:

- Combine nodes, circuits, applications, and users into groups that share the same security requirements.
- Define access rules that apply to each group.

Refer to the *DECNIS Introduction and Glossary* for a more detailed description of Secure Connections.

6.7.1 Setting Up Secure Connections

You can only set up Secure Connections access rules in the clearVISN DECNIS configurator. Follow these steps:

1. Run the clearVISN DECNIS configurator.
2. Click the **Secure Connections** button on the Main Navigation window.
3. Enter Secure Connections information.

Refer to the *clearVISN DECNIS Configurator User Guide* for more information about Secure Connections.

6.8 Setting the System Time on the DECNIS

You can check the system time of the DECNIS by checking the output of NCL SHOW commands, checking the events that are logged, or by issuing the following command:

```
NCL> SHOW NODE decnis DTSS CURRENT TIME
```

where *decnis* is the name of the DECNIS.

If the system time of the DECNIS is incorrect, use the following command to set it to the correct value:

```
NCL> UPDATE NODE decnis DTSS TIME yyyy-mm-dd-hh:mm:ss+a:b
```

where *yyyy-mm-dd-hh:mm:ss* is the year, month, day, and local time in hours, minutes and seconds. You must specify all parts of the date and time.

The variable *+a:b* is the TDF (time differential factor), and is the number of hours:minutes by which UTC time (coordinated universal time) differs from local time. Use a + sign if local time is ahead of UTC; use a – sign if local time is behind UTC.

Example

To set the system time of the DECNIS to a local time of 10.30 am, August 17, 1994, when local time is 5 hours behind UTC time, issue the following command:

```
NCL> UPDATE NODE decnis DTSS TIME 1994-08-17-10:30:00-5:0
```

6.9 MIB-II: System Group

You can use SNMP to set the following objects in the system group of MIB-II:

MIB Variable	Comments
	MIB-II, system group
sysContact	Specifies the contact person for the DECNIS.
sysName	Specifies the name by which the DECNIS will be known. If you use DECdns, it is recommended that you use the DECdns node name.
sysLocation	Specifies the physical location of the DECNIS.

Part II
Loading

This part describes how to load the DECNIS software onto the DECNIS hardware, and how to set up various types of loading for the DECNIS.

It contains the following chapters:

- Chapter 7 describes how to load the configured software to the DECNIS.
- Chapter 8 describes nonvolatile memory loading.
- Chapter 9 describes how to set up the DECNIS as a MOP proxy load host.
- Chapter 10 describes how to set up the DECNIS as a BOOTP gateway.

7
Loading a DECNIS

7.1 Introduction

This chapter describes the following:

- Loading the DECNIS from a load host.
- Reloading the DECNIS after the initial load.
- Restricting loading and dumping on the DECNIS.
- How the DECNIS loads and dumps its software.

The following chapters describe other aspects of loading and dumping:

- Chapter 8 describes how to set up nonvolatile memory loading (also known as flash memory loading).
- Chapter 9 describes how to set up the DECNIS as a proxy load host.
- Chapter 10 describes how to set up the DECNIS as a BOOTP gateway.

7.1.1 MOP and BOOTP Loading

You can downline load the DECNIS using MOP (Maintenance Operations Protocol), BOOTP/TFTP or both.

- MOP is a DIGITAL-specific protocol used for loading and dumping.
- BOOTP and TFTP are protocols used for loading and dumping, defined in RFCs 783 and 951. BOOTP determines the IP address of a device being loaded, and the names of load files. TFTP is the protocol used for loading.

7.1.1.1 Types of Connection for Loading

A load host can be connected to the DECNIS in either of two ways:

- Directly. The load host is on the same LAN as the DECNIS, or is directly connected to it by a synchronous line. Note that loading over a synchronous line is not supported by all load hosts.
- Indirectly. The load host is connected through another DECNIS acting as a proxy load host or BOOTP gateway (see Chapter 9 and Chapter 10).

7.2 Loading the DECNIS for the First Time

To load a DECNIS that has not been loaded previously, follow these steps:

1. Connect the DECNIS hardware unit to the LAN.
2. Plug the unit into the power supply and power up.

The DECNIS hardware unit then follows the steps described in Section 7.11.

7.3 Updating the DECNIS

If you have a management processor card (MPC-II or MPC-III) you can use the console to make dynamic updates to the contents of flash memory. You can add new script files, mark old script files for deletion, or clear flash memory and load a new image using the console commands. Refer to Chapter 3 for more information about controlling the contents of flash memory using the console.

7.4 Reloading the DECNIS

If the DECNIS has been loaded previously, you can use the following methods to reload it:

- Entering NCL commands.
- Powering up the DECNIS.
- Entering console commands. Refer to Chapter 2 for more information about the DECNIS console.

Section 7.4.1 to Section 7.4.3 describe various methods of reloading.

For more information about the NCL commands described here, refer to the NCL reference manual for your operating system, or NCL help. Note that on Windows 95/NT PCs, NCL help is available in a Windows help file.

7.4.1 Reloading Using the Default Type of Loading

The default type of loading is the type of loading set up for the DECNIS during load-host configuration: full nonvolatile memory loading, partial nonvolatile memory loading, or load host loading.

7.4.1.1 Entering the NCL LOAD Command

To reload the DECNIS using its default method, enter the following command:

- On a Windows 95/NT load host:

  ```
  NCL> LOAD NODE decnis/username/password DEVICE UNIT MP*
  ```

- On an OpenVMS load host:

  ```
  NCL> LOAD NODE decnis"username password" DEVICE UNIT MP*
  ```

- On a DIGITAL UNIX load host:

  ```
  NCL> LOAD NODE decnis/username/password DEVICE UNIT MP*
  ```

where: *decnis* is the node name of the DECNIS. On a Windows 95/NT PC load host, the node name is an IP address or IP node name.

username and *password* are the user name and password to enter when using NCL commands to manage the DECNIS.

7.4.1.2 Powering Up

If you power up the DECNIS, as described in Section 7.2, this will have the same effect as using the LOAD command in Section 7.4.1.1.

7.4.1.3 Using the Console

Enter the following command at the DECNIS console:

```
console> restart
```

This will have the same effect as the LOAD command in Section 7.4.1.1.

7.4.2 Reloading from a Load Host

Section 7.4.2.1 to Section 7.4.2.3 explain how to cause the DECNIS to reload from a load host, rather than from nonvolatile (flash) memory.

7.4.2.1 Entering NCL Commands

To cause the DECNIS to reload from a load host, enter the commands shown below. In each case, the first command tells the DECNIS to load from a load host the next time it reloads, and the second command reloads the DECNIS.

- On a Windows 95/NT PC load host:

```
NCL> SET NODE decnis/username/password HARDWARE -
_NCL> DEBUG FLAGS 1073741952

NCL> LOAD NODE decnis/username/password DEVICE UNIT MP*
```

- On an OpenVMS load host:

```
NCL> SET NODE decnis"username password" HARDWARE -
_NCL> DEBUG FLAGS 1073741952

NCL> LOAD NODE decnis"username password" DEVICE UNIT MP*
```

- On a DIGITAL UNIX load host:

```
NCL> SET NODE decnis/username/password HARDWARE -
_NCL> DEBUG FLAGS 1073741952

NCL> LOAD NODE decnis/username/password DEVICE UNIT MP*
```

where *decnis*, *username* and *password* are defined as in Section 7.4.1.1.

Note that these commands do not specify which load host will be used.

7.4.2.2 Powering Up

If you power up the DECNIS with the dump button pressed in, it will have the same effect as issuing the commands in Section 7.4.2.1, that is, it will attempt to load from the load host.

7.4.2.3 Using the Console

Enter the following command at the DECNIS console:

```
console> load -NETWORK
```

7.4.3 Reloading from a Specified MOP Load Host

You can cause the DECNIS to reload from a particular MOP load host, provided that the load host is reachable from the DECNIS.

To reload the DECNIS from a particular MOP load host, enter the following NCL command:

- On an OpenVMS load host:

```
NCL> LOAD NODE load-host"username password" MOP CLIENT client-name
```

- On a DIGITAL UNIX load host:

  ```
  NCL> LOAD NODE load-host/username/password MOP CLIENT client-name
  ```

Although MOP is not supported on Windows 95/NT load hosts, you can still issue this command from these load hosts to specify another load host that does support MOP.

where: *load-host* is the DECnet node name of the MOP load host.

username and *password* are the user name and password to enter when using NCL commands to manage the DECNIS.

client-name is the DECNIS MOP client name, as set up on the MOP load host.

7.5 Errors While Loading

If there are any errors in the NCL script files which have not been detected during CMIP conversion, then the DECNIS does the following:

- Logs errors to the console terminal, as described in Section 7.5.1.
- Logs script exception events, provided that the errors do not prevent communication with the event sink. Refer to the online manual *DECNIS Problem Solving* for a description of exception events.

7.5.1 NCL Script Errors Logged to the Console Terminal

If there is a console terminal connected to the DECNIS, the console will display NCL script errors in the following format:

```
Error processing script directive n, code m
```

where *n* is the NCL script directive number for the command that is in error, and *m* is a hexadecimal number that DIGITAL can use to analyze the error.

The NCL Checking Utility

To help avoid NCL errors, you should run the NCL checking utility before loading the DECNIS. This utility checks the DECNIS NCL script, and produces a log file. The log file shows the directive numbers of the commands in your NCL script, together with errors and warnings.

The NCL checking utility may not find all errors. However, the log file it produces is useful in any case, because it lists each NCL script directive together with its directive number. This enables you to match the directive numbers in console error messages against the actual commands in the NCL script, so that you can pinpoint the commands that are wrong.

In addition, the log file can help in interpreting script exception events, as such events may include directive numbers.

Refer to Section 1.6 for more information about the NCL checking utility.

7.6 Disabling and Restoring Loading from a MOP Load Host

This section describes how to disable loading from one or more MOP load hosts, and restore it after it has been disabled.

Note that Windows 95/NT load hosts do not support MOP loading, so these commands do not apply to them.

7.6.1 Disabling MOP Loading

To prevent loading from a particular MOP load host, use NCL to delete the MOP Client entity for the DECNIS on that load host.

7.6.2 Restoring MOP Loading

If you have deleted one or more DECNIS MOP Client entities, you can use the load-host configurator Restore option to recreate these MOP client entities. There are two ways to use Restore:

- Use the `Restore a router` option in the load-host configurator to restore an individual DECNIS.
- Use the automatic Restore procedure to restore all DECNIS systems set up by the load-host configurator.

Automatic Restore

To use automatic Restore, enter the command appropriate for your load host:

- OpenVMS load hosts:

  ```
  $ @SYS$MANAGER:NIS$HOST_CONFIG RESTORE
  ```

- DIGITAL UNIX load hosts:

  ```
  # /usr/lib/dnet/nis_host_config -r
  ```

7.7 Enabling Dumping

The DECNIS will by default prevent dumping on all connections. This allows the DECNIS to restart quickly after failure; typically, the DECNIS takes about two minutes to restart if it does not dump, as compared to 30 minutes if it does dump.

Section 7.7.1 and Section 7.7.2 describe how to enable dumping on the DECNIS.

7.7.1 Enabling Dumping Temporarily

You may want the DECNIS to dump the next time it fails, but to revert to not dumping thereafter. To do this, issue the following command interactively:

```
NCL> SET HARDWARE DUMP CONTROL dump-type
```

where *dump-type* is one of the following:

FULL DUMP
SYSTEM PROCESSOR DUMP

7.7.2 Enabling Dumping Permanently

To enable dumping permanently, enter the SET HARDWARE DUMP CONTROL command described in Section 7.7.1 in the user SET NCL script.

7.7.3 Dumping Using the Dump Button

If you press the red dump button on the DECNIS hardware unit, the DECNIS will dump, regardless of whether you have enabled dumping.

7.8 Restricting Connections Used for Loading and Dumping

7.8.1 Introduction

By default, the DECNIS does the following:

- Attempts to load over all connections.
- Prevents dumping over all connections.

This section and Section 7.9 describe how to:

- Selectively restrict loading.
- Selectively restrict dumping, provided that you have first explicitly enabled dumping on the DECNIS (see Section 7.7).

7.8.2 Types of Restriction

You can restrict DECNIS loading and dumping by specifying either or both of the following:

- The Network Interface Cards that are not allowed to be used for loading and/or dumping
- The load protocols that are not allowed to be used for loading and/or dumping

Example

1. You can ensure that the DECNIS will only load through one of its CSMA/CD connections. This would avoid delays in attempting to load through synchronous connections.
2. You can prevent the DECNIS from using BOOTP to perform dumps, ensuring that a MOP load host will always receive the dump file.

7.8.3 Cannot Restrict Individual Connections

Although you can restrict the cards used for loading and dumping, all ports on a particular card must have the same ability to load and dump . For example, you cannot specify that one port on a DEC WANcontroller 622 card can be used for loading and dumping, but the other cannot. If the card is enabled for loading and dumping, then all ports can be used for loading and dumping.

7.8.4 Restrictions on Protocols Used for Loading and Dumping

The only protocols supported for loading and dumping over serial lines are DEC HDLC and PPP.

For this reason, you must disable loading and dumping on any WAN card that has lines configured to use other protocols (for example, X.25). To disable loading and dumping, use the ADD HARDWARE SLOT FUNCTIONS DISABLE command. See Section 7.9.3 for an example.

7.8.5 Restrictions on Cards Used for Loading and Dumping

You cannot use the following Cards for loading or dumping:

- DECNIS ATMcontroller 631
- DECNIS HSSIcontroller 641

7.9 Commands to Manage Loading and Dumping Restrictions

This section describes the FUNCTIONS DISABLED commands used to restrict, or remove restrictions from, loading and dumping.

7.9.1 Entity and Attributes Used for Restricting Loading and Dumping

To restrict loading and dumping on a card, you use the NCL entity HARDWARE SLOT, with its attribute FUNCTIONS DISABLED.

7.9.1.1 HARDWARE SLOT Entity

Each card on a DECNIS has an associated HARDWARE SLOT entity. The name of this entity is the number of the slot in which the card is inserted.

7.9.1.2 FUNCTIONS DISABLED Attribute of the HARDWARE SLOT Entity

The FUNCTIONS DISABLED attribute determines whether the DECNIS can load and dump over the connections on a card, and the protocols it can use.

Possible values are shown in Table 7–1.

Table 7–1 Values of FUNCTIONS DISABLED Attribute

Value	Description
MOP LOAD REQUESTER	Loading using MOP over all lines associated with the backplane slot is prohibited
MOP DUMP REQUESTER	Dumping using MOP over all lines associated with the backplane slot is prohibited
IP LOAD REQUESTER	Loading using BOOTP/TFTP over all lines associated with the backplane slot is prohibited
IP DUMP REQUESTER	Dumping using BOOTP/TFTP over all lines associated with the backplane slot is prohibited

7.9.2 Commands Used to Restrict Loading and Dumping

This section lists the commands you can use to restrict loading and dumping, or change previous restrictions on loading and dumping.

Do Not Include Commands in NCL Script File

The value of the FUNCTIONS DISABLED command for each slot is stored in nonvolatile memory on the DECNIS, and does not change when the DECNIS is rebooted. For this reason, do not add any FUNCTIONS DISABLED commands to the user NCL script files.

You can override the FUNCTIONS DISABLED values in nonvolatile memory by entering FUNCTIONS DISABLED commands interactively.

7.9.3 ADD Command

Use this command to add loading and/or dumping to the functions that are disabled for the hardware slot.

ADD Command Example

```
NCL> ADD NODE decnis HARDWARE SLOT 3 FUNCTIONS DISABLED -
_NCL>{MOP LOAD REQUESTER, MOP DUMP REQUESTER, -
_NCL> IP LOAD REQUESTER, IP DUMP REQUESTER}
```

Result: This prevents lines associated with the card in slot 3 of the backplane from being used to load and dump the DECNIS.

7.9.4 REMOVE Command

Use this command to remove loading and/or dumping from the functions that are disabled for the hardware slot.

REMOVE Command Example

```
NCL> REMOVE NODE decnis HARDWARE SLOT 3 FUNCTIONS DISABLED -
_NCL> {MOP DUMP REQUESTER}
```

Result: This now allows the lines associated with the card in slot 3 of the backplane to use MOP for sending upline dumps. They still cannot use BOOTP /TFTP for dumping, and they cannot load using either MOP or BOOTP/TFTP.

7.9.5 SHOW Command

Use this command to see what functions (if any) are disabled for a particular hardware slot.

SHOW Command Example

```
NCL> SHOW NODE decnis HARDWARE SLOT 3 FUNCTIONS DISABLED
```

Clearing Nonvolatile Memory

You can clear the nonvolatile memory of the DECNIS by powering up the hardware with the dump button held in: see the *Installation and Service Manual* for your hardware.

7.10 Moving a DECNIS

If your DECNIS is moved to a new site, or a Management Processor Card is moved to a different DECNIS, you must hold the dump button in when you power up the DECNIS, to clear the contents of nonvolatile memory. This will force the DECNIS to load from a load host.

For more information, refer to the *Installation and Service Manual* for your hardware unit.

7.11 How the DECNIS Loads Its Software

When the DECNIS hardware unit is powered up, it does the following:

1. Runs diagnostic tests on the DECNIS system.
2. Checks whether there is a valid image in DECNIS nonvolatile memory. (If you have an MPC-III card, it checks the Boot Area of nonvolatile memory.)
3. If there is no valid image, sends out a request to downline load the software, as described in steps 12 to 14.
4. If there is a valid image, checks its load instructions to see if it should load that image, or if it should load an image from a load host.
5. If the load instructions are to load from a load host, sends out a request to downline load the software, as described in steps 12 to 14.
6. If the load instructions are to load the image from nonvolatile memory, it does so.
7. Checks whether there is a valid CMIP file in nonvolatile memory.
8. If there is no valid CMIP file, loads the CMIP file from a load host.
9. Checks to see whether the profile files are required and, if so, are present.
10. If there are no profile files, loads them from the load host. The load is then complete.
11. If the profile files are present, the load is then complete.
12. When requesting a load from a load host, the DECNIS issues the request on the connection over which it was last loaded, using the protocol with which it was last loaded.
13. If there is no response, or if it has not been previously loaded, it broadcasts MOP and BOOTP load requests on all available connections.
14. One of the load hosts (either BOOTP or MOP) responds to this request:
    - If a MOP load host is the first to respond, the DECNIS sends it a load request. The MOP load host then loads the required files.
    - If a BOOTP load host is the first to respond, it sends the DECNIS a BOOTP message, containing the IP address of the relevant DECNIS interface, and the name and location of the load files. The DECNIS then sends it a TFTP load request, and the BOOTP load host loads the required files.

7.12 How the DECNIS Dumps Its Software

7.12.1 Load Hosts and Dumping

Each load host you configure using the load-host configurator or the clearVISN DECNIS configurator can also act as a dump sink.

7.12.2 How Dumping Works

When you press the dump button on the DECNIS hardware unit, the DECNIS sends a request for a load host to accept a dump of its memory:

1. It issues a request for a host system to accept the dump on the connection over which it was last dumped.

   If there is no response, or if it has not previously dumped, it broadcasts the request on all available connections.

2. One of the load hosts (MOP or BOOTP) responds by sending a message to the DECNIS. The DECNIS then sends the dump to this load host.

8

Nonvolatile (Flash) Memory Loading

8.1 Introduction

This chapter describes how to set up a DECNIS to load from its own nonvolatile (flash) memory.

8.1.1 Methods for Setting Up Flash Memory Loading

The recommended method is to do either of the following:

- On OpenVMS or DIGITAL UNIX use the load-host configurator, as described in the manual *DECNIS Installation and Configuration for OpenVMS and DIGITAL UNIX*.
- On Windows 95/NT hosts, use the clearVISN DECNIS configurator, as described in *clearVISN DECNIS Configurator User Guide*.

However, you may need to set up nonvolatile loading using NCL commands. Section 8.3 describes how to do this.

8.1.2 How Flash Memory Loading Works

The term nonvolatile, or flash, memory refers to an area of DECNIS memory which can be used to store:

- The DECNIS software image only.
- A combined file containing the DECNIS software image, configuration file(s) and any profile files.
- The DECNIS software image and one or more separate configuration files.

When the DECNIS Loads From Flash Memory

The DECNIS initially loads its image and configuration files from a load host. After that, it will reload from flash memory if the following apply:

- The following NCL command is in the DECNIS configuration file.

  `SET NODE` *`decnis`* `HARDWARE DEBUG FLAGS 0`

The DECNIS configurator automatically inserts this command in the DECNIS NCL script if you select nonvolatile or flash memory loading.

- The required file or files have been loaded into flash memory.

  The software image is always loaded into flash memory when you initially load the DECNIS from a host.

  On the next load, if there is only an image in flash memory, the DECNIS will load it, and load the CMIP/profile files from the load host. If the CMIP /profile files are in flash, the DECNIS will load them as well.

  The CMIP/profile files can be placed in flash memory in two ways:

  – By creating and loading a combined file.

  – By inserting individual CMIP files into flash memory dynamically, as described in Chapter 3.

However, note that there are some circumstances in which the DECNIS automatically loads from a load host; see Section 8.4 for details.

8.2 Modifying Flash Memory Dynamically

The rest of this chapter assumes that you wish to load a combined file into flash memory. However, you can, if you wish, use DECNIS console commands to add one or more separate CMIP files to flash memory dynamically, and designate the one to be loaded. Refer to Chapter 3 for details.

8.3 Setting Up Flash Memory Loading Using Commands

Section 8.3.1 to Section 8.3.2 describe how to set up flash loading without using the configurator. Note that you can only do this on OpenVMS and DIGITAL UNIX load hosts. On Windows 95/NT hosts, you can only set up loading information using the clearVISN DECNIS configurator.

This section assumes that you have already created a CMIP file, either in the configurator or as described in Section 1.5.4.

8.3.1 Method for Setting Up Flash Memory Loading

To set up the DECNIS for flash memory loading, follow these steps:

1. Create a combined file, as described in Section 1.5.5.
2. Issue the command to load from the load host, as described in Section 8.3.2. This is required to load the new combined file to the DECNIS.

8.3.2 Issuing the Command to Load from a Load Host

Issue the following NCL comand to tell the DECNIS to load from a load host:

- OpenVMS load hosts:

  ```
  NCL> SET NODE decnis"username password" HARDWARE DEBUG FLAGS 1073741952
  ```

- DIGITAL UNIX load hosts:

  ```
  ncl> SET NODE decnis/username/password HARDWARE DEBUG FLAGS 1073741952
  ```

where: *decnis* is the node name of the DECNIS.

username and *password* are the user name and password required to use NCL commands to manage the DECNIS.

Issuing the Network Loading Console Command

If the DECNIS has previously been loaded, you can use the DECNIS console to reload it from a load host. Issue the following console command:

```
console> load -flash
```

8.3.3 Example: Setting Up Flash Memory Loading

8.3.3.1 Before You Begin

In this example, you have already done the following:

- Run the load-host configurator and selected `Load host for both CMIP and image`.
- Run the DECNIS text-based configurator and created an NCL script.

You have not created a CMIP file.

8.3.3.2 Available Information

In this example, the following information is available:

Type of load host	DIGITAL UNIX MOP
Load client name of the DECNIS	load_decnis1
Node name of the DECNIS	paris_decnis1
Username	Rosencrantz
Password	Guildenstern

8.3.3.3 Procedure

To set up paris_decnis1 for flash memory loading, follow these steps:

1. Create a CMIP file by entering the command:

   ```
   # /usr/lib/dnet/nis_script_compile nis_load_decnis1
   ```

2. Create a combined file by entering the command:

   ```
   # /usr/lib/dnet/nis_combine nis041 nis_load_decnis1
   ```

3. Instruct the DECNIS to load from a load host by entering the command:

   ```
   NCL> SET NODE paris_decnis1/rosencrantz/guildenstern -
   _NCL> HARDWARE DEBUG FLAGS 1073741952
   ```

4. Reload the DECNIS. The new combined file will be loaded from the load host.

5. The next time you reboot the DECNIS, it will load from flash memory. To check that it has loaded successfully, enter the following command:

   ```
   NCL> SHOW NODE paris_decnis1/rosencrantz/guildenstern LAST REBOOT REASON
   ```

   The reason displayed should be either **Flash updated successfully** or code 12. If there is any other reason, refer to the online *DECNIS Problem Solving* manual.

8.4 Loading a New Image or Configuration File

In order to load new software versions and configuration files, you need to reload the DECNIS from a load host. The configurators automatically force the next load to be from the load host in the following circumstances:

- If you use the load-host configurator Update option to update one or more DECNIS systems to Version 2.1 or higher.
- If you use the automatic Update procedure to update all DECNIS systems.
- If you use the load-host configurator Add option to set up a new DECNIS.
- If you create a new CMIP file, or a new combined file, within the DECNIS configurator.

8.5 Forcing the DECNIS to Load from the Load Host

You may sometimes need to force the DECNIS to load from a load host, without using the configurator. Section 8.5.1 lists the circumstances in which you will need to do this.

8.5.1 When to Force a Load from the Load Host

You will need to force the DECNIS to load from the load host if:

- You have created a new combined file from the command line, as described in Section 1.5.5.1.
- The system on which you are configuring is on a different network from the load host from which you are loading.
- The HARDWARE DEBUG FLAGS command issued by the DECNIS configurator fails. This will happen if either of the following are true:
  - The DECNIS is not connected to the network and powered up.
  - The DECNIS is not reachable from the system on which the command is issued.

8.5.2 Methods of Forcing a Load from the Load Host

Refer to Section 7.4.2 and Section 7.4.3.

8.6 Version 7-07 ROMs and Flash Memory Loading

If the DECNIS attempts to load from a load host, and no load has been obtained within one hour, what happens next will depend on the version of Management Processor ROMs fitted to the DECNIS.

- With ROM versions earlier than V7-07, the DECNIS will reset and run the self-test after one hour. It will not attempt to load from flash memory.
- With ROM versions V7-07 or later, the DECNIS will attempt to load from flash memory after one hour. If there is a software image in flash memory, it will be loaded.

  On future loads, the DECNIS will again attempt to load from the load host.

8.6.1 Finding the ROM Version Used by the DECNIS

To check what version of Management Processor ROM your DECNIS is using, enter the following command:

```
NCL> SHOW NODE decnis DEVICE UNIT MP* FIRMWARE ID
```

The screen will display the firmware identifier, which includes the version number. For example:

```
  Status
   Firmware Identifier = "0 2-3.6 V7-04 2.7"
```

In this example, the version number is V7-04.

9

Using the DECNIS as a Proxy Load Host

9.1 Introduction

This chapter describes how to set up the DECNIS as a proxy load host.

9.1.1 Definition of Proxy Load Host

A **proxy load host** is a system that can load another system or receive dumps from it, but which does not itself store the load or dump files. Instead, the load and dump files are stored on a **real load host**. They are sent to and from the proxy load host as required.

9.1.2 Using the DECNIS as a Proxy Load Host

When acting as a proxy load host, the DECNIS can get the load files from the real load host in either of two different ways:

- Using the Data Access Protocol (DAP). This is a DECnet protocol used for file access and transfer.
- Using the TFTP protocol. This is an IP protocol used for loading and dumping.

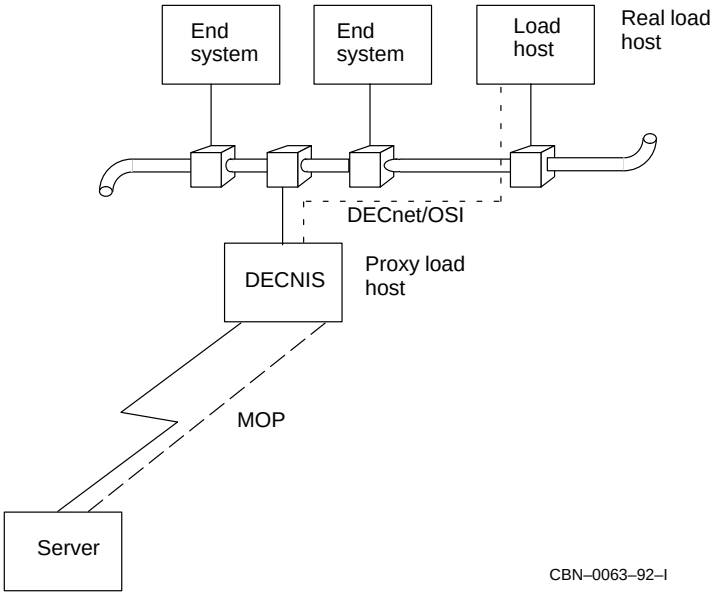
Once the DECNIS has obtained the files, it uses MOP over an HDLC or PPP link to load the target systems.

9.1.3 Example: Proxy Load Host

Figure 9–1 shows the DECNIS acting as a proxy load host.

The DECNIS obtains the load files from the real load host and downline loads them to the target system.

Figure 9–1 DECNIS Acting as a Proxy Load Host



9.2 Setting Up Proxy Loading: DECnet and MOP

This section describes how to set up the DECNIS so that it can do the following:

- Receive files from a real load host, using DECnet, and downline load them to other systems, using MOP.
- Receive dumps from other systems, using MOP, send them to the real load host, using DECnet.

9.2.1 Requirements for the Real Load Host

The real load host, which stores the files to be downline loaded, must meet the following requirements:

- It must be a DECnet system that supports DAP.
- It must be reachable from the proxy load host, using DECnet protocols.

9.2.2 Requirements for the DECNIS Proxy Load Host

9.2.2.1 Supported Data Link

The DECNIS proxy load host can use the following types of data link to connect to the target systems:

- HDLC
- PPP

9.2.2.2 Configuration: Proxy Load Host

Before you set up a DECNIS as a proxy load host, do the following:

1. Run the load-host configurator and add load information for the DECNIS proxy load host.
2. Run the DECNIS text-based configurator and configure the DECNIS proxy load host. You should:
   - In the `Network Interface Cards` section, set up a WANcontroller card, for example, a W618 card.
   - In the `Lines` section, set up an HDLC or PPP circuit on one of the lines on the WANcontroller card.
   - In the `Create NCL Script` section, create a CMIP file or a combined file.

9.2.3 Requirements for Target Systems

If the target system is a DECNIS, configure it, following the steps in Section 9.2.2.2.

If the target system is not a DECNIS, configure it and generate a CMIP or combined file as described in the documentation for the target system.

If you configure the target system on a system other than the real load host, copy the load files to the real load host.

9.2.4 Information Required

You will need to know the following:

- The node name of the real load host. This is the same as the name of the KNOWN TOWER entity for the load host. Note: The node name must have the correct format for a full DECdns node name, even if you are not using DECdns on your network.

  Refer to Section 1.13 for a full explanation of how to create the KNOWN TOWER entity.

- The location on the real load host of the files to be loaded to the target system: either the combined file or the software image and CMIP and profile files.

Refer to Appendix G for the standard location of load files on various load hosts.

9.2.5 Procedure

Follow these steps to set up the DECNIS as a proxy load host:

1. If necessary, create a MOP circuit that uses the HDLC or PPP data link.

   The DECNIS text-based configurator sets up a MOP circuit for each HDLC data link. The MOP circuit has the same name as the HDLC line. For example, if the line name is W618-4-2, the MOP circuit name is also W618-4-2.

   The DECNIS text-based configurator does not set up a MOP circuit for PPP data links. To set up a MOP circuit on a PPP link, enter the following NCL commands, or insert them into the user NCL scripts:

   ```
   NCL> CREATE NODE decnis MOP CIRCUIT circuit-name TYPE HDLC
   NCL> SET NODE decnis MOP CIRCUIT circuit-name -
   _NCL> LINK NAME PPP LINK link-name
   ```

   Note that the MOP circuit is of type HDLC.

2. Enable the LOAD SERVER and DUMP SERVER functions on the MOP circuit:

   ```
   NCL> ENABLE NODE decnis MOP CIRCUIT circuit-name FUNCTION -
   _NCL> {LOAD SERVER, DUMP SERVER}
   ```

   This command will not affect any other functions currently enabled on the MOP circuit.

3. Create a MOP client to represent the target system:

   ```
   NCL> CREATE NODE decnis MOP CLIENT client-name
   ```

   where: *decnis* is the node name of the DECNIS proxy load host.
   client-name is the MOP client name to identify the target system.

   Note that MOP client names are also referred to as load client names in this manual.

4. Specify the MOP circuit to be used to load to the target system (the MOP client name identifies the target system).

   ```
   NCL> SET NODE decnis MOP CLIENT client-name -
   _NCL>  CIRCUIT circuit-name
   ```

5. Create an entry for the real load host in the Known Towers database:

   ```
   NCL> CREATE NODE decnis SESSION CONTROL -
   _NCL> KNOWN TOWER host TOWERS {(tower1), (tower2), ...}
   ```

   where: *host* is the node name of the real load host.
   tower1 and *tower2* are towers that describe the protocols and NSAP addresses used to communicate with the remote node.

6. Specify the image and load files to be loaded to the target system:

   - If the target system will use nonvolatile memory loading, specify the name of the combined file:

     ```
     NCL> SET NODE decnis MOP CLIENT client-name -
     _NCL> SYSTEM IMAGE {"host::filespec"}
     ```

     where: *host* is the node name of the real load host. This is the name of the KNOWN TOWER entity created in step 4.
     filespec is the specification of the combined file on the real load host.

   - If the target system will reload from the load host, follow these steps:

     a. Specify the software image to be loaded to the target system:

        ```
        NCL> SET NODE decnis MOP CLIENT client-name -
        _NCL> SYSTEM IMAGE {"host::image"}
        ```

b. Specify the CMIP (script) file to be loaded to the target system:

```
NCL> SET NODE decnis MOP CLIENT client-name -
_NCL> SCRIPT FILE {"host::cmip"}
```

c. Specify the directory containing the profile files to be loaded to the target system. Enter the name of the CMIP script file, as in the previous step:

```
NCL> SET NODE decnis MOP CLIENT client-name MANAGEMENT IMAGE -
_NCL> {"host::cmip"})
```

where: *host* is the node name of the real load host. This is the name of the KNOWN TOWER entity created in step 4.
image is the file specification of the image on the real load host.
cmip is the specification of the CMIP file on the real load host.

7. Specify the name of the dump file for the target system:

```
NCL> SET NODE decnis MOP CLIENT client-name -
_NCL> DUMP FILE {"host::filespec"}
```

where: *host* is the node name of the real load host.
filespec is the specification of the file to which the dump will be written on the real load host.

8. Ensure that the DECNIS has read access to the load files and write access to the directory where the dump files will be written on the real load host. Do this in one of the following ways:

   - Set the file protection on the load and dump files on the real load host to allow access by the default DECnet account.
   - Set up a proxy account on the real load host, to allow access to these files by *proxy-name*::LES$MOP, where *proxy-name* is the node name of the DECNIS.

These steps will set up the DECNIS to load another DECNIS. However, if you want the DECNIS to load to other types of target system, you may need to specify additional MOP client characteristics for the target system.

9.2.5.1 Enter Commands in the User NCL Script Files

It is recommended that you add the NCL commands in Section 9.2.5 to the appropriate user NCL script files (refer to Section 1.5.6). If you do not do this, you will have to reenter the commands each time the proxy load host is rebooted.

9.2.6 Example NCL Commands: OpenVMS Real Load Host

This section gives NCL commands to configure a DECNIS to act as a proxy load host for another DECNIS. You have the following information:

Name of OpenVMS real load host	ORG:.NORTH.REAL
Name of DECNIS proxy load host	DECNIS_PROXY
MOP Client Name of DECNIS target	DECNIS_TARGET
Type of loading specified for target	All files from load host

The commands are:

```
create node DECNIS_PROXY mop circuit hdlc_circ1 type hdlc
set node DECNIS_PROXY mop circuit hdlc_circ1 link name hdlc link -
   hdlc_link1 logical station hdlc_stat1
enable node DECNIS_PROXY mop circuit hdlc_circ1 function
   {load server,dump server}

create node DECNIS_PROXY mop client DECNIS_TARGET
set node DECNIS_PROXY mop client DECNIS_TARGET circuit hdlc_circ1

create node DECNIS_PROXY session control known tower org:.north.real -
   towers {([DNA_CMIP-MICE], [DNA_SESSIONCONTROLV3, NUMBER = 19], -
   [DNA_NSP], [DNA_OSINETWORK, 37:12345:02-00:08-2B-14-78-66-11:20])}

set node DECNIS_PROXY mop client DECNIS_TARGET system image
   {"org:.north.real::mom$system:nis041.sys"}
set node DECNIS_PROXY mop client DECNIS_TARGET script file
   {"org:.north.real::mom$system:nis_DECNIS_TARGET.cmip"}
set node DECNIS_PROXY mop client DECNIS_TARGET management image
   {"org:.north.real::mom$system:nis_DECNIS_TARGET.cmip"}
set node DECNIS_PROXY mop client DECNIS_TARGET dump file
   {"org:.north.real::mom$system:nis_DECNIS_TARGET.dmp"}
```

9.2.7 Example NCL Commands: DIGITAL UNIX Real Load Host

This section gives NCL commands to configure a DECNIS to act as a proxy load host for another DECNIS. You have the following information:

Name of DECnet/OSI for DIGITAL UNIX real load host	ORG:.NORTH.REAL
Name of DECNIS proxy load host	DECNIS_PROXY
MOP Client Name of DECNIS target	DECNIS_TARGET
Type of loading specified for target	All files from load host

The commands are:

```
create node DECNIS_PROXY mop circuit hdlc_circ1 type hdlc
set node DECNIS_PROXY mop circuit hdlc_circ1 link name hdlc link -
   hdlc_link1 logical station hdlc_stat1
enable node DECNIS_PROXY mop circuit hdlc_circ1 function
   {load server,dump server}

create node DECNIS_PROXY mop client DECNIS_TARGET
set node DECNIS_PROXY mop client DECNIS_TARGET circuit hdlc_circ1

create node DECNIS_PROXY session control known tower ORG:.NORTH.REAL -
   towers {([DNA_CMIP-MICE], [DNA_SESSIONCONTROLV3, NUMBER = 19], -
   [DNA_NSP], [DNA_OSINETWORK, 37:12345:02-00:08-2B-14-78-66-11:20])}

set node DECNIS_PROXY mop client DECNIS_TARGET system image
   {"org:.north.real::/usr/lib/mop/nis041.sys"}
set node DECNIS_PROXY mop client DECNIS_TARGET script file
   {"org:.north.real::/usr/lib/mop/nis_DECNIS_TARGET.cmip"}
set node DECNIS_PROXY mop client DECNIS_TARGET management image
   {"org:.north.real::/usr/lib/mop/nis_DECNIS_TARGET.cmip"}
set node DECNIS_PROXY mop client DECNIS_TARGET dump file
   {"org:.north.real::/usr/lib/mop/nis_DECNIS_TARGET.dmp"}
```

9.3 Setting Up Proxy Loading: TFTP and MOP

This section describes how to set up the DECNIS so that it can:

- Receive files from a real load host using TFTP and downline load them to other systems, using MOP.
- Receive dumps from other systems using MOP and send them to the real load host, using TFTP.

9.3.1 Requirements for the Real Load Host

The real load host, which stores the files to be downline loaded, must meet the following requirements:

- It must be running the TFTP daemon.
- It must be reachable from the proxy load host.

9.3.2 Requirements for the DECNIS Proxy Load Host

The requirements for the DECNIS proxy load host are as described in Section 9.2.2.1 and Section 9.2.2.2.

9.3.3 Requirements for Target Systems

If the target system is a DECNIS, configure it, following the steps in Section 9.2.2.2.

If the target system is not a DECNIS, configure it and generate a CMIP or combined file as described in the documentation for the target system.

If you configure the target system on a system other than the real load host, copy the load files to the real load host.

9.3.4 Information Required

You will need to know the following:

- The IP address of the real load host.
- The location on the real load host of the files to be loaded to the target system: either the combined file or the software image and CMIP and profile files.

  Refer to Appendix G for the standard location of load files on various load hosts.

9.3.5 Procedure

Follow these steps to set up the DECNIS as a proxy load host:

1. If necessary, create a MOP circuit that uses the HDLC or PPP data link.

   The DECNIS text-based configurator sets up a MOP circuit for each HDLC data link. The MOP circuit has the same name as the HDLC line. For example, if the line name is W618-4-2, the MOP circuit name is also W618-4-2.

   The DECNIS text-based configurator does not set up a MOP circuit for each PPP data link. See Section 9.2.5, step 1 for the NCL commands required to set up a MOP circuit on a PPP link.

2. Enable the LOAD SERVER and DUMP SERVER functions on the MOP circuit:

   ```
   NCL> ENABLE NODE decnis MOP CIRCUIT circuit-name FUNCTION -
   _NCL> {LOAD SERVER, DUMP SERVER}
   ```

   This command will not affect any other functions currently enabled on the MOP circuit.

3. Create a MOP client to represent the target system:

   ```
   NCL> CREATE NODE decnis MOP CLIENT client-name
   ```

   where: *decnis* is the node name of the DECNIS proxy load host.
   client-name is a MOP client name to identify the target system.

4. Specify the MOP circuit to be used to load to the target system (the MOP client name identifies the target system):

   ```
   NCL> SET NODE decnis MOP CLIENT client-name -
   _NCL>  CIRCUIT circuit-name
   ```

5. Specify the image and load files to be loaded to the target system:

   - If the target system will use nonvolatile memory loading, specify the name of the combined file:

     ```
     NCL> SET NODE decnis MOP CLIENT client-name -
     _NCL> SYSTEM IMAGE {"host-ip-address:filespec"}
     ```

     where: *host-ip-address* is the host IP address of the real load host.
     filespec is the specification of the combined file on the real load host.

- If the target system will reload from the load host, follow these steps:

  a. Specify the software image to be loaded to the target system:

  ```
  NCL> SET NODE decnis MOP CLIENT client-name -
  _NCL> SYSTEM IMAGE {"host-IP-address:image"}
  ```

  b. Specify the CMIP (script) file to be loaded to the target system:

  ```
  NCL> SET NODE decnis MOP CLIENT client-name -
  _NCL> SCRIPT FILE {"host-IP-address:cmip"}
  ```

  c. Specify the directory containing the profile files to be loaded to the target system. Enter the name of the CMIP script file, as in the previous step:

  ```
  NCL> SET NODE decnis MOP CLIENT client-name MANAGEMENT IMAGE -
  _NCL> {"host-IP-address:cmip"})
  ```

  where: *host-ip-address* is the host IP address of the real load host.
  image is the file specification of the image on the real load host.
  cmip is the specification of the CMIP file on the real load host.

6. Specify the name of the dump file for the target system:

```
NCL> SET NODE decnis MOP CLIENT client-name -
_NCL> DUMP FILE {"host-ip-address:filespec"}
```

where: *host* is the host IP address of the real load host.
filespec is the specification of the file to which the dump will be written on the real load host.

These steps will set up the DECNIS to load another DECNIS. However, if you want the DECNIS to load to other types of target system, you may need to specify additional MOP client characteristics for the target system.

9.3.5.1 Enter Commands in the User NCL Script Files

It is recommended that you add the NCL commands in Section 9.3.5 to the appropriate user NCL script files (refer to Section 1.5.6). If you do not do this, you will have to reenter the commands each time the proxy load host is rebooted.

9.3.6 Example NCL Commands: DIGITAL UNIX Real Load Host

This section gives NCL commands to configure a DECNIS to act as a proxy load host for another DECNIS. You have the following information:

IP address of DIGITAL UNIX real load host	23.24.32.78
Name of DECNIS proxy load host	DECNIS_PROXY
MOP client name of DECNIS target	DECNIS_TARGET
Type of loading specified for target	All files from load host

The commands are:

```
create node DECNIS_PROXY mop circuit hdlc_circ1 type hdlc
set node DECNIS_PROXY mop circuit hdlc_circ1 link name hdlc link -
   hdlc_link1 logical station hdlc_stat1
enable node DECNIS_PROXY mop circuit hdlc_circ1 function
   {load server,dump server}

create node DECNIS_PROXY mop client DECNIS_TARGET
set node DECNIS_PROXY mop client DECNIS_TARGET circuit hdlc_circ1

set node DECNIS_PROXY mop client DECNIS_TARGET system image
   {"23.24.32.78:/usr/lib/mop/nis041.sys"}
set node DECNIS_PROXY mop client DECNIS_TARGET script file
   {"23.24.32.78:/usr/lib/mop/nis_DECNIS_TARGET.cmip"}

set node DECNIS_PROXY mop client DECNIS_TARGET management image
   {"23.24.32.78:/usr/lib/mop/nis_DECNIS_TARGET.cmip"}

set node DECNIS_PROXY mop client DECNIS_TARGET dump file
   {"23.24.32.78:/usr/lib/mop/nis_DECNIS_TARGET.dmp"}
```

10

Using the DECNIS as a BOOTP Gateway

10.1 Introduction

This chapter describes how to set up the DECNIS as a BOOTP gateway.

10.1.1 Definition of BOOTP Gateway

A BOOTP gateway can do the following:

- Downline load files located on **BOOTP servers** to **BOOTP clients**,
- Receive dumps from BOOTP clients and relay them to BOOTP servers.

These functions are known as BOOTP relay functions.

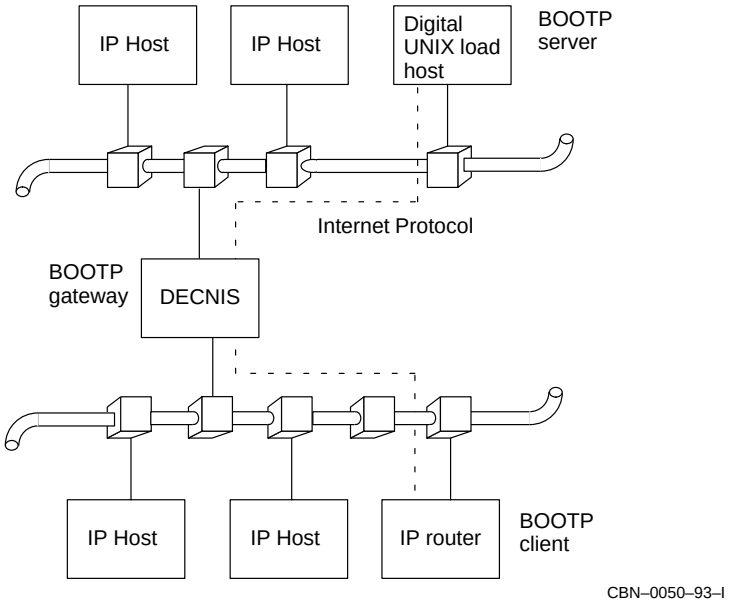
The BOOTP server is the real load host, and stores the load and dump files. The BOOTP gateway does not store load and dump files; rather, it relays BOOTP load and dump requests from the BOOTP client to the BOOTP server, and BOOTP responses from the BOOTP server to the BOOTP client.

A DECNIS BOOTP gateway can send load requests directly to one or more BOOTP servers, or can forward them to another BOOTP gateway.

10.1.2 Example: BOOTP Relay

Figure 10–1 shows a DECNIS BOOTP gateway. The DECNIS BOOTP gateway obtains the load files from the BOOTP server, and downline loads them to the BOOTP client.

Figure 10–1 DECNIS Acting as a BOOTP Gateway



10.2 Setting Up the DECNIS as a BOOTP Gateway

This section describes how to set up a DECNIS so that it can load BOOTP clients, and receive dumps from them.

10.2.1 Requirements for BOOTP Servers

You need to specify the BOOTP servers to which the DECNIS will forward BOOTP messages. You can specify multiple BOOTP servers on each routing circuit configured for BOOTP relay.

10.2.1.1 Type of System

A BOOTP server can be any of the following:

- A Windows NT or Windows 95 PC, with the Microsoft TCP/IP network protocol stack installed (included in Windows NT/95).
- A DIGITAL UNIX host, configured for BOOTP/TFTP, as described in the manual *DECNIS Installation and Configuration for OpenVMS and DIGITAL UNIX*.

- A non-DIGITAL UNIX® load host configured for BOOTP, as described in the manual *DECNIS Installation and Configuration for OpenVMS and DIGITAL UNIX*.

10.2.1.2 BOOTP Load File Locations

On a DIGITAL UNIX BOOTP server, the directory and file names for the load files must be set up so that the BOOTP server can respond correctly to load and dump requests. For more information, refer to Section G.2.

You can set up load information for all BOOTP clients, and locate the files correctly, by running one of the following on the BOOTP server:

- The load-host configurator, if the BOOTP server is a DIGITAL UNIX system.
- The clearVISN DECNIS configurator, if the BOOTP server is a Windows NT or Windows 95 system.

10.2.2 Requirements for BOOTP Gateways

10.2.2.1 Type of Data Link

The DECNIS can perform BOOTP gateway functions on all types of circuits except X25 DA.

10.2.2.2 Configuration of BOOTP Gateway on DIGITAL UNIX systems

Before you enter the NCL commands to set up DECNIS as a BOOTP gateway, you make sure that you have done the following in the DECNIS text-based configurator:

1. Selected Yes for IP routing in the Configuration Options section,
2. In the Lines section, set up the routing circuit(s) to be used for BOOTP gateway functions.

10.2.2.3 Configuration of BOOTP Gateway on Windows NT/95 PCs

Before you enter the NCL commands to set up DECNIS as a BOOTP gateway, you make sure that you have done the following in the clearVISN DECNIS configurator:

1. Tick the IP box on the General tab page, under the **System** button.
2. On the Circuits tab page in the IP routing section, configure the routing circuit(s) to be used for BOOTP gateway functions.

If you are using a non-DIGITAL UNIX system as a BOOTP server, you will have to copy the required files to it, as described in the manual *DECNIS Installation and Configuration for OpenVMS and DIGITAL UNIX*.

10.2.3 Requirements for BOOTP Clients

10.2.3.1 Configuration: DIGITAL UNIX Systems

If the BOOTP client is a DECNIS, do the following:

1. Run the load-host configurator and add load information. Select either BOOTP or MOP or BOOTP as the method to be used for loading.
2. Run the DECNIS text-based configurator. The requirements are the same as in Section 10.2.2.2.

10.2.3.2 Configuration: Windows NT or Windows 95 Systems

If the BOOTP client is a DECNIS, it needs to meet the requirements described in Section 10.2.2.3.

10.2.3.3 Configuration: Other Systems

If the BOOTP client is a type of system other than those in Section 10.2.3.1 and Section 10.2.3.2, configure it and generate a CMIP file as described in the documentation for the BOOTP client.

10.2.4 Information Required for BOOTP Relay Configuration

You will need to know the following:

- The IP address of the BOOTP server(s).
- The names of the routing circuits to be used for BOOTP relay functions on the BOOTP gateways.

10.2.5 Procedure

Follow these steps to set up DECNIS systems as BOOTP gateways:

1. Check that the BOOTP server or servers are configured so that they can reach the BOOTP gateway and clients.
2. If the routing circuit(s) to be used for BOOTP are enabled, disable them. For example:

   ```
   NCL> DISABLE ROUTING CIRCUIT circuit-name
   ```

3. On each DECNIS BOOTP gateway, specify the circuit(s) to be used for BOOTP functions; that is, to receive BOOTP requests and carry out the load. This circuit must be connected to the same LAN to which the BOOTP client is connected.

```
ncl> SET NODE decnis ROUTING CIRCUIT circuit-name - )
_ncl>  BOOTP SERVERS {a.a.a.a.,b.b.b.b}
```

where: *decnis* is the node name of the DECNIS BOOTP gateway.
circuit-name is the name of the routing circuit.
a.a.a.a,b.b.b.b are IP addresses of BOOTP servers, or of other BOOTP gateways through which the DECNIS is obtaining the files.

10.2.5.1 Enter Commands in the User NCL Script Files

It is recommended that you add the NCL command in Section 10.2.5 to the SET user NCL script file (refer to Section 1.5.6). If you do not do this, you will have to reenter the commands each time the DECNIS BOOTP gateway is rebooted.

10.2.6 Example

Figure 10–2 shows systems to be set up for BOOTP relay.

10.2.6.1 Available Information

In this example, the following information is available:

System	Node Name	IP Address	Routing Circuit
BOOTP gateways	nis_peach		L602-3-1
	nis_mango		L602-6-0
BOOTP server	host north	28.34.26.10	

10.2.6.2 Procedure

You wish to configure nis_peach and nis_mango as BOOTP gateways so that they can load BOOTP clients from the BOOTP server host_north. Follow these steps:

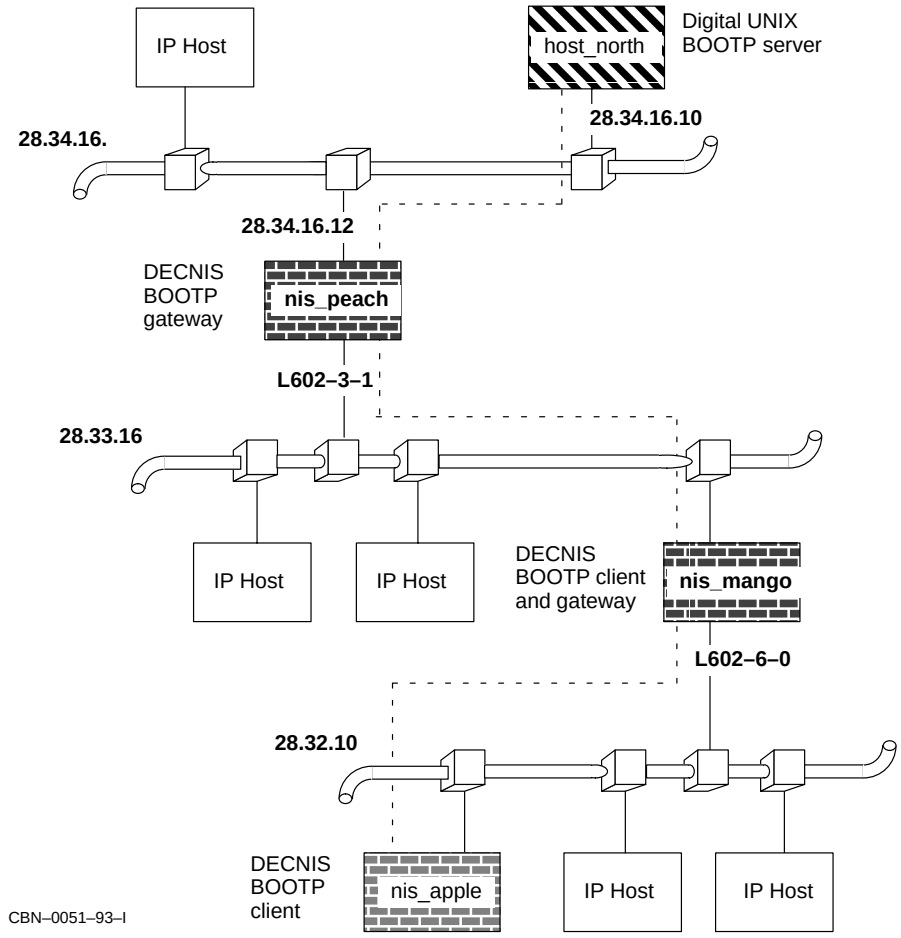
1. Disable the routing circuit that nis_peach will use (L601-3-1) to forward load requests to the BOOTP server (IP address 28.34.26.10), and downline load files from the BOOTP server.

   ```
   ncl> DISABLE NODE nis_peach ROUTING CIRCUIT L601-3-1
   ```

2. Specify the routing circuit L601-3-1 as a BOOTP RELAY circuit:

   ```
   ncl> SET NODE nis_peach ROUTING CIRCUIT L601-3-1 -
   _ncl> BOOTP RELAY 28.34.16.10
   ```

Figure 10–2 BOOTP Relay Example



3. Enable the routing circuit L601-3-1:

   ```
   ncl> ENABLE NODE nis_peach ROUTING CIRCUIT L601-3-1
   ```

4. Disable the routing circuit that nis_mango will use (L602-6-0) to forward load requests to the BOOTP server (IP address 28.34.26.10), and downline load files from the BOOTP server.

   ```
   ncl> DISABLE NODE nis_mango ROUTING CIRCUIT L602-6-0
   ```

5. Specify the routing circuit L602-6-0 as a BOOTP RELAY circuit

```
ncl> SET NODE nis_mango ROUTING CIRCUIT L602-6-0 -
_ncl> BOOTP RELAY 28.34.16.10
```

6. Enable the routing circuit L602-6-0:

```
ncl> ENABLE NODE nis_peach ROUTING CIRCUIT L602-6-0
```

Index

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

W