4DDN Network Management Guide and Man Pages

Document Number 007-0821-030

Contents

Figures

Tables

Chapter 1

Introduction

4DDN™ is a Silicon Graphics® software option that connects IRIS® Series workstations and servers to a DECnet™ network. It provides DECnet connection and data transfer services to interactive users, task-to-task communication services for applications programming, and a suite of Network Control Program (NCP) commands, Digital Equipment Corporation's software for DECnet network management.

4DDN is an implementation of Digital Network Architecture (DNA) protocols and runs on a DECnet Phase IV network. IRIS workstations running 4DDN operate as Ethernet® end nodes in the DECnet. IRIS workstations on remote networks are reachable by means of DECnet routers.

In addition to 4DDN, your IRIS workstation might also be running Transmission Control Protocol/Internet Protocol (TCP/IP), the standard networking software shipped with the IRIS system. TCP/IP and 4DDN software can run simultaneously on your IRIS. Although an IRIS can be configured for multiple Ethernet interfaces, 4DDN can only run on one Ethernet interface at a time.

1.1 Using This Guide

The *4DDN Network Management Guide* is one of a three-volume set that documents Silicon Graphics' 4DDN product. Other members of the set are the *4DDN Programming Guide*, which explains how to write network applications for 4DDN; and the *4DDN User's Guide*, which explains how to use the interactive commands that provide user-level services.

This Guide explains the administrative tasks needed to set up and maintain 4DDN software on an IRIS workstation attached to a DECnet network. It introduces the terms and concepts needed to manage IRIS nodes in the network, explains how to configure and test 4DDN software, and describes the 4DDN commands available to monitor and control the network from an IRIS workstation.

As a convenience to readers, this guide contains some procedures that set up VAX™/VMS™ nodes to reciprocate in 4DDN connections. However, if you are not the network manager, consult with your DECnet or VMS™ system administrator before using these procedures. See Digital Equipment Corporation documentation for complete information on preparing the VMS environment.

1.1.1 About the Audience

The *4DDN Network Management Guide* assumes you are either a DECnet administrator responsible for adding and maintaining IRIS nodes on the network, or an IRIS user responsible for connecting your workstation to the DECnet and maintaining a healthy network connection. It also assumes that the IRIS is properly cabled to a functioning Ethernet local area network (LAN).

1.1.2 Chapter Contents

The *4DDN Network Management Guide* is organized into these chapters:

Chapter 1	This introduction explains the purpose of the 4DDN product and the *4DDN Network Management Guide*.
Chapter 2	''Notes to New Users'' explains the architecture of the DECnet network and defines the terms used throughout this guide. It also gives quick tips on using IRIX™, the IRIS operating system, and provides a table of command equivalents for VMS, IRIX, and 4DDN.
Chapter 3	''Configuring 4DDN Software'' contains the procedure you need to set up 4DDN software in an IRIS workstation. It also explains how to set up 4DDN proxy accounts on both IRIX and VMS systems, and how to set up a privileged account on an IRIS workstation.
Chapter 4	''Enabling the Mail and Print Utilities'' explains how to enable 4DDN mail service after software is configured. It also provides instructions for configuring the print utility on IRIS systems not running the standard print spooler.
Chapter 5	''Using the Network Control Program'' explains how to start and stop NCP, the program used to do network management. It also describes the syntax of NCP commands and command error messages.
Chapter 6	''Managing the Database'' describes the contents of the database used for network operation and how to change it.
Chapter 7	''Managing Network Nodes'' explains how to set up network user accounts on 4DDN and VMS nodes, and describes frequently used Network Control Program commands that control local and remote nodes.
Chapter 8	''Managing Circuits, Lines, and Links'' describes how to control network circuits, lines, and links using Network Control Program commands.

Chapter 9	‘‘Testing the Network’’ explains how to verify that 4DDN software is operating properly within the local node and over local-to-remote connections.
Chapter 10	‘‘NCP Command Reference’’ contains an alphabetical listing of NCP commands available with 4DDN software.
Appendix A	‘‘Troubleshooting Hints’’ explains the most common errors in 4DDN network management and how to correct them.
Appendix B	‘‘Default Parameter Values’’ lists the as-shipped settings of 4DDN parameters.
Appendix C	‘‘NCP Error Messages’’ explains the meaning of error messages from NCP commands and their probable cause.
Appendix D	‘‘Manual Pages’’ contains the IRIX manual pages relevant to 4DDN network management tasks.

1.2 Typographical Conventions

This document uses the standard UNIX® convention when referring to entries in the IRIX documentation. The entry name is followed by a section number in parentheses. For example, *cc*(1) refers to the *cc* manual entry in Section 1 of the IRIX manual pages.

In text descriptions, file names and commands are shown in *italics*. Command syntax descriptions and examples appear in `typewriter font`, and optional arguments appear in **boldface** type. Names of variables and error codes used in the VMS environment are in uppercase.

In user entries and examples, we use these conventions:

`typewriter`	**`User entries are shown in typewriter bold`** font. Enter these lines as shown, unless otherwise instructed.
UPPERCASE	VMS and NCP commands appear in uppercase; however, uppercase is optional. You entries may be in either upper or lowercase.

lowercase	IRIX commands appear in lower case. Observe case conventions for all commands processed by the IRIX operating system.

1.3 Related Documentation

Installation instructions for 4DDN software are given in the *IRIS Software Installation Guide* and the *4DDN Release Notes* that accompany the version of software you are installing. Both documents contain important information you should review before using the procedures in this guide.

The documents listed below provide supplementary information on topics covered in the *4DDN Network Management Guide*. In addition to the documents in this list, please see the *Owner's Guide* for your IRIS model.

Silicon Graphics Publications:

- *4DDN Programming Guide*
- *4DDN User's Guide*

Digital Equipment Corporation publications:

- *DECnet Digital Network Architecture*
  (Phase IV) General Description
- *Guide to Networking on VAX/VMS*
- *VAX/VMS Network Control Program Reference Manual*

1.4 Product Support

Silicon Graphics provides a comprehensive product support and maintenance program for IRIS products. For further information, contact your service organization.

Chapter 2

Notes to New Users

This chapter acquaints readers with some of the concepts and terms that apply to networking in general, or to the DECnet network, specifically. It explains how network software is organized, describes the functional components of the DECnet network, and identifies the 4DDN software components that support each network service. This chapter also provides tips for using the IRIX user interface to enter 4DDN commands and gives a table of equivalent commands for VMS, IRIX, and 4DDN.

2.1 If You Are New to Networks

A network is a configuration of computers that permits the exchange of data and sharing of resources among its members. Incompatibilities in hardware and software components in the network must be resolved before an intelligent exchange of information can take place. To resolve these differences, the government and private industry have collaborated to set standards for the development of networking products.

Under the guidance of the International Standards Organization (ISO), a set of standards have been developed in the communications industry that specify a network architecture based on layers. This architecture is known as the ISO model for Open Systems Interconnection (ISO/OSI). Each layer in the model conforms to rules that govern its structure and function. Implementations of the standards are referred to as *protocols*.

2.1.1 Network Protocols

Each protocol layer in the network operates according to specific rules to perform a function and deliver a service to the layer above it. In addition to the interface between the upper and lower layers, each layer also communicates with a peer layer running in other computers connected to the network. Since layer functions are standardized, peer layers running in different computers can be supplied by different vendors.

Two widely used protocols are Transmission Control Protocol, Internet Protocol (TCP/IP), developed by the Department of Defense, and Digital Network Architecture (DNA), developed by Digital Equipment Corporation (DEC). The structure and function of TCP/IP and DNA protocols are similar to those specified by the ISO/OSI model.

2.1.2 Message Packets

Information travels through the network in *packets*, small blocks of data prefixed with addressing and control information. Packets originate at the upper network layer and move down succeeding layers until they arrive at the transmission hardware. As each layer receives the packet, it adds header information and passes the packet down to the layer below it.

After a packet is transmitted over network hardware, it is moved up through the layers on the destination computer. Each layer of receiver software strips off header information from the packet and passes it to the layer above it. The end result of this process is a structure that the receiving machine can understand and use.

2.2 The Architecture of DECnet

DECnet, an implementation of DNA that provides connectivity among Digital Equipment systems, is composed of seven layers. In local areas of a DECnet network, the first two layers are frequently an implementation of Ethernet, a specification developed by DEC, Intel Corporation, and Xerox Corporation. DECnet layers perform these functions:

Physical Link Layer	provides the electrical and mechanical support, as well as the software device drivers for network equipment.
Data Link Layer	separates the data from noise coming in over the communication line, frames the data, and corrects transmission errors.
Routing Layer	forwards packets to their destination and controls packet traffic on the network.
End Communications Layer	controls the addressing and timing of data exchanged between communicating processes in different nodes.
Session Control Layer	adapts data received from the End Communications Layer to the specific needs of the local operating system.
Network Application Layer	controls functions required for file transfers, virtual terminal service, and remote access to files and devices on the local system.
User Layer	provides resource sharing, file access and transfers, and the interface to network management tools.

In addition to these layers, DNA also includes a Network Management Layer that spans other layers in the hierarchy. Network Management provides the services needed to plan, control, and maintain the network.

Figure 2.1 illustrates DNA network layers.

Figure 2-1 DNA Network Layers

2.3 If You Are New to DECnet

DECnet is the name given to a collection of hardware and software products that enables computers running DEC operating systems to be members of a network. Since IRIS workstations run IRIX, a Silicon Graphics implementation of the UNIX operating system, they require specialized software, 4DDN, to provide DECnet connectivity. An IRIS workstation running 4DDN is often called a *4DDN node* on the DECnet network.

2.3.1 DECnet Components

A DECnet network contains two kinds of nodes: end nodes and router nodes. An *end node* sends packets to other DECnet nodes and receives packets that bear its address. An IRIS workstation running 4DDN operates as an Ethernet end node on the DECnet.

A *router node* receives its own packets and forwards packets addressed to other nodes. When a particular router is assigned to route messages for end nodes, it is considered the *designated router* for the end nodes.

Nodes attached to the same Ethernet line are *adjacent nodes*. Nodes are said to be *reachable* when they are available for connections to other nodes. The availability of a node can be controlled by turning its *state* on or off.

The data path between a local node and a remote one is called a *circuit*. The connection between two processes that occurs over a circuit is called a *logical link*. A circuit can support simultaneous logical links between network nodes. The physical media that connects computers and support circuits are called *lines*.

Network software maintains information on events that occur at network nodes, circuits, links, and lines. Event information is saved in *counters*, which are used to track network performance and throughput. This information can be retrieved with the NCP, the user interface to DECnet management.

2.3.2 DECnet Node Identification

A node name and address identify nodes in the DECnet network. If nodes are members of a subnetwork, or *area*, the address includes an area number. Name and address information is stored in a database on each node and used to route connections to other network nodes. Node names and addresses follow these conventions:

Node Address — is a unique number assigned by the network manager; it must conform to the following format:

```
area-number.node-number
```

where:

area-number	is an integer in the 1–63 range. Area numbers must be unique within the network. If an area number is not specified, the area number defaults to the local area.
node-number	is an integer in the 1–1023 range. Node numbers must be unique to the specific network area.

Node Name	is a string of up to six alphanumeric characters (letters and numbers), and at least one character in the name must be alphabetic. Only one name is allowed per node. Node names are not case sensitive; lowercase letters are converted to uppercase. However, since IRIX is case sensitive, it is easier to manage 4DDN nodes when all letters in their names are lowercase.

2.3.3 4DDN Software Components

A network connection requires communication between two processes: a *client* process running in the node where the request is entered and a *server* process running in the responding node. Unless each counterpart is running, logical links cannot be established between network nodes.

The master server process, *dnserver*, is responsible for network connections to 4DDN nodes. *dnserver* is a continuously running IRIX process that invokes other servers to handle requests for service. For each server *dnserver* invokes, a corresponding client process resides on other DECnet nodes. Any errors that might occur when *dnserver* is invoking servers is logged in the IRIX file */usr/adm/SYSLOG*.

dnserver invokes these other 4DDN servers to handle incoming requests:

FAL	The File Access Listener (FAL) server handles incoming requests that require access to files on its node. It provides the local file system with network file access functions, such as file copies and directory listings. FAL uses Data Access Protocol (DAP) to communicate with processes on other nodes.
NML	The Network Management Listener (NML) server supports NCP, which handles network management tasks. Together with NCP, NML enables users to determine the status of a network, zero lines and circuit counters, and perform other network control operations. NML executes Network Information and Control Exchange (NICE) protocol messages that it receives from NCP. The responses to the protocol messages are returned to the appropriate NCP for display.

sethostd	The virtual terminal server *sethostd* enables a user on one node to log into a remote node on the network. The *SET HOST* client and *sethostd* server use the CTERM protocol to communicate during virtual terminal sessions.
dnMaild	The mail server, *dnMaild*, handles incoming mail deliveries to the 4DDN node where it runs. It passes the mail messages it receives from remote nodes to the local delivery service. *dnMaild* uses a subset of the VMS MAIL protocol.

2.4 If You Are New to IRIX

IRIX is the operating system supplied with all IRIS workstations. It is a version of the UNIX System V operating system (originally developed at Bell Laboratories) and also includes some BSD UNIX enhancements (features developed by the University of California at Berkeley). The IRIX *kernel* handles system-level tasks such as managing hardware, and the *shell* is the command interpreter for user entries.

The IRIS workstation offers two user interfaces to IRIX: a visual interface called the *WorkSpace*™, which you use by selecting icons; and a shell interface, which you use by typing commands at the IRIX prompt. 4DDN commands must be used from an IRIX shell. (See Appendix A of the *Owner's Guide* for your IRIS for more introductory information on IRIX.)

2.4.1 Opening an IRIX Shell

To open an IRIX shell, follow this procedure:

1. Click on the *Tools* toolbox.
2. Select *Shell* from the menu.
3. Position the shell and click it in place.

2.4.2 Using Command Equivalents

Table 2-1 shows you command equivalents in the VMS, IRIX, and 4DDN environments. IRIX, like other UNIX systems, is case sensitive; your commands must be entered in lower or uppercase, as shown.

VMS	IRIX	4DDN
COPY	cp	dncp
DELETE	rm	dnrm
DIRECTORY	ls	dnls
RENAME	mv	dnmv
PRINT	lp	dnlp
SET HOST	rlogin	sethost
SUBMIT	sh	dnex
TYPE	more	(no equivalent)
MAIL	mail, Mail	dnMail

Table 2-1 4DDN Command Equivalents

Chapter 3

Configuring 4DDN Software

The procedures in this chapter explain how to configure 4DDN software after it has been installed on an IRIS system. They describe how to prepare your system for 4DDN configuration, explain the information you need before you begin, then guide you through the configuration procedure.

To do the procedures in this chapter, you should be familiar with the IRIS screen interface and know how to locate files on an IRIS system. You should also be able to use a UNIX editor such as *vi*, *ed*, or *emacs*.

3.1 Preparing Your System

The 4DDN configuration procedure assumes that your IRIS hardware and software meet 4DDN operating requirements. To prepare your system, follow these steps:

1. **Complete hardware and software upgrades.**

   First, read the ''Configuration Information'' section of your *4DDN Release Notes* for 4DDN hardware and software requirements. If any system upgrades are advised in this information, be sure to complete them before doing any procedures in this chapter.

2. **Install 4DDN software on the IRIS workstation.**

   If 4DDN software has not already been installed on your IRIS system, install it before doing the procedures in this chapter. Follow the general instructions in the *IRIS Software Installation Guide* and the specific installation information given in your copy of the *4DDN Release Notes*.

3.2 Collecting the Information You Need

During the configuration procedure, you will be expected to supply or confirm information about the DECnet network where your workstation will be connected, as well as information contained in TCP/IP files (standard networking software shipped with IRIS workstations). Your DECnet or TCP/IP network administrator should provide this information.

Be ready to supply this information during the configuration procedure:

- **The names and addresses of DECnet nodes you want to reach.**

  As administrator of a 4DDN node, you must supply 4DDN software the name and address of each DECnet node you plan to connect to. Node names and addresses are entered in a file called */usr/etc/dn/nodes*, which you build during the configuration procedure.

  If you get a list of node names and addresses from a network administrator, be sure you can identify the name and address of your IRIS workstation from the list. If you plan to assign a new node name to your IRIS during configuration, be sure the name you select is unique. The address of your node must also be unique. Node addresses are usually assigned by a central source, such as a network administrator responsible for coordinating address assignments throughout the DECnet.

- **The host name for your workstation.**

  If your IRIS is new or has never been configured for network operation, you will be required to assign it a unique host name during 4DDN configuration. This host name enables IRIS network hardware to recognize the network messages destined for your system. Host names are assigned by two TCP/IP files, */etc/sys_id* and */etc/hosts*. When an IRIS is first shipped, these files contain the host name ‘‘IRIS.’’ However, you replace this generic name with a unique host name when you configure the workstation for TCP/IP operation.

  Recall that the name of a node in the DECnet network is assigned in the *nodes* file (see first bulleted item above). Although it is possible to assign one name to your host on the DECnet network and a different name on the TCP/IP network, it is easier to remember and maintain the same name in both DECnet and TCP/IP environments. Node names on a DECnet network are limited to six characters. If you choose a longer name for your station, it will be truncated to the first six characters for DECnet operation.

- **The Internet address of your station.**

  Your IRIS workstation is shipped with a test Internet address of 192.0.2.1 entered in the */etc/hosts* file. The test address must be changed whenever an IRIS workstation is configured for network operation. If the Internet address of your IRIS is still set to its test value, you will be required to assign it a valid Internet address during 4DDN configuration.

  Caution: The test Internet address 192.0.2.1 inhibits operation of the Ethernet board in your IRIS, making network connections impossible. For this reason, it is mandatory that you change the address and reboot your system when configuring an IRIS as a network station.

- **The superuser password to the IRIS, if needed.**

  To do the configuration procedure, you must access the *root* account on the IRIS system. Because this account grants superuser privileges, workstation owners can assign it password protection to prevent others from accessing it. If you are configuring 4DDN on an IRIS with a password-protected *root* account, be sure you know the password before beginning the configuration procedure.

3.3 The Configuration Procedure

Once your system is ready and you have the information you need, you can begin the configuration procedure. The configuration procedure has five parts. Each part is listed below and described in detail in the remainder of this chapter.

1. Assign the name and address to your station.
2. Edit the *nodes* file.
3. Run the configuration scripts.
4. Set the access path to 4DDN software.
5. Create a default user account on the IRIS.

3.3.1 Assigning a Host Name and Address

To operate as a member of a network, your workstation needs a unique host name entered in the */etc/sys_id* file and needs its host name and a valid Internet address entered in the */etc/hosts* file. (See Section 3.2, ''Collecting the Information You Need.'') The first part of the configuration procedure is to edit the *sys_id* and *hosts* files to enter your workstation's name and Internet address.

1. **Log into the *root* account on the IRIS.**

   If you have not yet logged into the IRIS system, log into the *root* account using the standard IRIS login procedure. If you have already logged into the IRIS on a different account, switch your user ID and account to *root* with the command shown below. (If you have not assigned a password to the *root* account, the password prompt is not displayed.) After your entry, the superuser prompt is displayed.

   ```
   % su
   Password:       (password not echoed)
   #
   ```

2. **Change to the */etc* directory.**

   The *sys_id* and *hosts* files are stored in the */etc* directory. Change to this directory with the command shown below:

   ```
   # cd /etc
   ```

3. **Edit the */etc/sys_id* file.**

   Use a UNIX editor to open */etc/sys_id*. Delete the current entry (this is ''IRIS'' unless you changed it earlier) and enter the new network name for your workstation. Your entry should be in lowercase letters.

   Figures 3-1 and 3-2 illustrate the *sys_id* file before and after editing.

   ```
   IRIS
   ```

   Figure 3-1 The *sys_id* File Before Editing

   ```
   newnode
   ```

   Figure 3-2 The *sys_id* File After Editing

4. **Edit the */etc/hosts* file.**

   Edit the */etc/hosts* file for your system. Delete the test Internet address (190.0.2.1) and enter the valid Internet address for your workstation. Also delete the name listed beside the test Internet address and enter the new network name for your workstation (this should be the same as your entry in *sys_id*).

   Figures 3-3 and 3-4 illustrate the *hosts* file before and after editing.

   Caution: Do not remove the entry ''127.0.0.1'' from this file. It is required by the IRIS window system.

```
127.0.0.1            localhost

#This network number, '192.0.2' is the officially blessed
                     'test' network.
#   It should be changed immediately to your own official
                     network number.
192.0.2.1           IRIS
```

Figure 3-3 The *hosts* File Before Editing

```
127.1            localhost

#This network number, '192.0.2' is the officially blessed
                     'test' network.
#   It should be changed immediately to your own official
                     network number.
192.10.10.1      newnode
```

Figure 3-4 The *hosts* File After Editing

5. **Reboot your system.**

   You must reboot your IRIS system for the name and address changes to take effect. Set the verbose option to display startup information, and reboot your system as shown below:

   ```
   # /etc/chkconfig verbose on
   # reboot
   ```

   As the startup information is displayed, check your display screen for a message like the sample shown below. It indicates that IRIS network hardware (the Ethernet controller) has been enabled and your TCP/IP name and address are assigned. The message in this sample means that Ethernet controller 0 is enabled and the workstation is assigned the name *newnode*.

   ```
   Configuring ec0 as newnode
   ```

If you did not get a success message during system reboot, or if you saw a message indicating your system rebooted for standalone operation, repeat all instructions in Section 3.3.1 to correct any errors in *sys_id* or *hosts*, then reboot your system again.

Once you have successfully completed the instructions in this section, the new name and address are assigned to your station and you can go on to the instructions in Section 3.3.2, ''Editing the *nodes* File.''

3.3.2 Editing the *nodes* File

The purpose of the *nodes* file is to initialize the database that describes the DECnet network to 4DDN software. *nodes* contains commands to set DECnet node names and addresses. When you run the 4DDN configuration scripts (Section 3.3.3), the information in *nodes* is used to establish a permanent database.

It is not necessary to include the name and address of all DECnet nodes in the *nodes* file. Once 4DDN is running, you can add node names and addresses to the database using NCP tools (instructions in Chapter 6).

The *nodes* file shipped with 4DDN software contains dummy values. In this part of the configuration procedure, you will replace the dummy values in */usr/etc/dn/nodes* with valid DECnet node names and addresses given to you by your network administrator.

1. **Log into the *root* account on the IRIS.**

   If you haven't yet logged into the IRIS system, log into the *root* account using the standard IRIS login procedure. If you have already logged into the IRIS on a different account, switch to the superuser account with the command shown below. After your entry, the superuser prompt is displayed.

   ```
   % su
   Password:        (password not echoed)
   #
   ```

2. **Change to the */usr/etc/dn* directory.**

   The *nodes* file is stored in the */usr/etc/dn* directory. Change to this directory with command shown below:

   ```
   # cd /usr/etc/dn
   ```

3. **Enter your list of node names and addresses in *nodes*.**

   Use a UNIX editor to open *nodes*. Replace the dummy names and addresses with valid DECnet node names and addresses in your network; do not replace the final line of the original entries, however. Node names must be no longer than six characters (longer names are invalid) and must be entered in lowercase letters.

   Figures 3-5 and 3-6 illustrate a *nodes* file before and after editing. 4DDN can use the *nodes* file as soon as your edits are made; no reboot is necessary. When your edits are complete, go on to the instructions in Section 3.3.3, ''Running the Configuration Scripts.''

```
!4DDN Sample NCP Host Database                 $Revision: 1.5

!Edit this file to contain your node's name and address.
!Contact the network administrator for a node address if you
!are connecting to an existing network.

!Node names can be at most six alphanumeric characters.
!Node addresses are composed of area (1 to 63) and
                  node (1 to 1023) numbers.

set node 1.2 name topcat
set node 1.4 name pluto
set node 1.5 name romula
set node 1.6 name vulcan
set node 1.7 name saturn
define known nodes all
```

Figure 3-5 */usr/etc/dn/nodes* Before Editing

```
!4DDN Sample NCP Host Database                 $Revision: 1.5

!Edit this file to contain your node's name and address.
!Contact the network administrator for a node address if you
!are connecting to an existing network.

!Node names can be at most six alphanumeric characters.
!Node addresses are composed of area (1 to 63) and
                   node (1 to 1023) numbers.

set node 23.90 name newnod
set node 23.38 name vms1
set node 23.81 name vax1
set node 23.57 name iris6
define known nodes all
```

Figure 3-6 *\/usr/etc/dn/nodes* After Editing

3.3.3 Running the Configuration Scripts

The configuration scripts *dninstall.sh* and *dnstart.sh* set up the permanent and volatile DECnet databases in your workstation and start the DECnet server process, *dnserver*. Once these scripts are run, you can set 4DDN to start automatically whenever the IRIS is rebooted. In this part of the configuration procedure, you will run the configuration scripts and set 4DDN to start automatically (optional).

1. **Make sure your directory is */usr/etc/dn*.**

   The directory */usr/etc/dn* must be the current directory when you run the *dninstall.sh* and *dnstart.sh* scripts. You can verify the current directory by entering the command below and checking its output:

   ```
   # pwd
   /usr/etc/dn
   ```

2. **Run *dninstall.sh*.**

   To run the *dninstall* script, enter this command at the superuser prompt:

   ```
   # sh dninstall.sh
   ```

After your entry, the script posts advisory information along with a prompt that looks like the sample below:

```
Is "newnod" used as this node’s name in the nodes file? [y]
```

The node name it posts is your entry in the *sys_id* file. If your entry is longer than six characters, only the first six are shown. Answer y[es] if the node name shown in the prompt is the name you entered in the *nodes* file.

If you entered a different name in *nodes* from the one shown in the prompt, enter `n`, and type the node name entry from your *nodes* file when you are prompted. (A different name is not recommended. See Section 3.2, ‘‘Collecting the Information You Need’’ for details.)

After your entry, *dninstall* builds the permanent 4DDN database in your station and posts an ‘‘All done.’’ message when it is finished. For large networks, this process takes a few minutes.

3. **Run *dnstart.sh*.**

   The *dnstart* script loads the volatile database, turns on 4DDN for networking, and starts *dnserver*, the 4DDN master server process. To run the *dnstart* script, enter this command at the superuser prompt:

   ```
   # sh dnstart.sh
   ```

4. **Confirm that 4DDN is running.**

   To verify that 4DDN is running, enter the commands shown below. The first command posts a list of DECnet nodes and their status; the nodes you can connect to are listed as ‘‘Reachable.’’ The second command shows you whether *dnserver* is running; if *dnserver* is running, its name is posted in the output from this command:

   ```
   # ./ncp show known nodes
   ```

   ```
   # ps -e | grep dnserver
   ```

   If you do not get success messages from these commands, be sure you completed the instructions in Section 3.3.1 and 3.3.2 correctly; then try the instructions in this step again. If you completed each step correctly and 4DDN fails to start on your workstation, contact your service organization for assistance.

5. **Set 4DDN to start automatically (optional).**

   To set 4DDN to start automatically when your workstation is rebooted, issue the command shown below. This command sets a flag in the file */etc/config/4DDN* to signal the restart sequence to run *dnstart.sh*. *4DDN* must be entered in uppercase letters.

   ```
   # /etc/chkconfig 4DDN on
   ```

   Note: If you choose not to have automatic startup, you can start 4DDN software manually by running *dnstart.sh* after your system is rebooted.

After you complete the procedures in this section, the 4DDN database and server process are set up in your workstation. To make 4DDN commands easily accessible to users, complete the instructions in Section 3.3.4, ‘‘Setting the Path to 4DDN Software.’’

3.3.4 Setting the Path to 4DDN

During its startup sequence, the IRIS workstation sets a path to the directories it will search in order to find user commands. 4DDN directories must be listed in this path so a user can execute 4DDN commands without specifying their entire pathnames.

In this part of the configuration procedure, you will enter 4DDN directories in the search path for your user account (and accounts for other users who might be using 4DDN on this system). You should also add 4DDN directories to the path of the *root* account if you expect to use 4DDN while logged in as *root*.

The file that contains the *set path* command varies with the UNIX shell you use. Since the C Shell is standard on the IRIS, the instructions below assume you are using *csh* and your *set path* command is in the *.cshrc* file. If you are using the Bourne Shell, make your changes to the *.profile* file.

1. **Change to your home directory**.

   The file containing the *set path* command for your user account is *.cshrc* in your home directory. To change to your home directory, enter this command:

   ```
   # cd ~login-name
   ```

2. **Edit *.cshrc* in your home directory**.

   The directories of 4DDN software are */usr/etc/dn* and */usr/bin/dn*. To add these directories to your path, open the *.cshrc* file in your home directory and add this line to the end of the file:

   ```
   set path = ($path /usr/etc/dn /usr/bin/dn)
   ```

3. **Edit *.cshrc* on other user accounts**.

   If you are configuring 4DDN for other users on this system, repeat steps 1 and 2 (above) for each additional 4DDN user.

4. **Edit *.cshrc* in *root* (optional)**.

   If you want 4DDN commands to be executable by the *root* user without a full path specification, add the *set path* command shown above in the */.cshrc* file located in the *root* directory.

5. **Activate the new path**.

   To activate the new search path, you can either log out and log into your workstation, or enter this command:

   ```
   # source .cshrc
   ```

Once you complete the instructions in this section, users whose login files you edited will be able to execute 4DDN commands without entering full pathnames. Go on to the instructions in Section 3.3.5, ‘‘Creating a Default Account on the IRIS.’’

3.3.5 Creating a Default Account

Some incoming connections to your 4DDN node will arrive without the access information needed to access a particular user account. These connections use a default account that is set up like other user accounts on the IRIS. The default account is similar to the IRIS guest account. It grants non-privileged access to the connections it services.

To set up the default account, you create an entry for it in the */etc/passwd* file. The as-shipped version of 4DDN software assumes your entry for the default account will specify the user name *decnet* and the password *DECNET*. If you wish, you can change these values later, using the instructions in Section 7.1, ‘‘Maintaining Default Accounts.’’

In this part of the configuration procedure, you will create the default account by creating an entry for it in the */etc/passwd* file and assigning it a password.

1. **Change to the */etc* directory**.

   The *passwd* file is stored in the */etc* directory. To change to this directory, enter this command:

   ```
   # cd /etc
   ```

2. **Edit the *passwd* file.**

   The *password* file must identify *decnet* as the account to be used for incoming requests without access information, with a user-ID and group-ID of 998 (guest). It also identifies the login directory for the connection and the login program to be used at the start of the session.

   Edit */etc/passwd*, adding the following line to the end of this file:

   ```
   decnet::998:998:Decnet_Account:/usr/tmp:/bin/login
   ```

The previous example sets */usr/tmp* as the home directory for incoming connections. However, if incoming connections will use applications that create files in the home directory, you might want to specify a directory with more disk space than */usr/tmp*.

3. **Set the password on the *decnet* account**.

   To set the password for the *decnet* account on your node, issue the *passwd* command and the password `DECNET` when prompted, as shown in the sample below. Your password entry must be in uppercase letters.

   ```
   # /bin/passwd decnet
   New password:  DECNET
   Re-enter new password:  DECNET
   ```

4.

   Since the *decnet* account is not intended as a login option for IRIS users, it should be omitted from the account icons posted on the visual login screen. To disable the *decnet* login icon, add this line to the */etc/passwd.sgi* file:

   ```
   decnet: noshow
   ```

When this procedure is finished, the IRIS node is ready to accpet incoming connections that contain no access control information. These connections will access a non-privileged account called *decnet*.

3.3.6 Creating a Privileged Account (optional)

The as-shipped version of 4DDN software does not permit superuser access to incoming connections, but you can enable superuser access if you set up a privileged account. The privileged account is a user account on the IRIS with a user identification number (UID) of zero (0). To access the privileged account, a remote user must specify both the account name and password in the connection request.

The */etc/passwd* file on the IRIS must contain an entry that specifies the name and password for the privileged account. The name of the privileged account must also be entered in the */usr/etc/dn/servers.priv* file. The *priv* entry in *servers.priv* specifies the name of the privileged account.

Any incoming connection with a UID of zero is validated against the *priv* entry in *servers.priv*. In addition, it's name and password are validated in */etc/password* before it is granted superuser access to the IRIS.

To set up a privileged account on the IRIS, follow the instructions below.

Note: Before you set up a privileged account, consult with your network administrator.

1. **Edit the** ***/etc/passwd*** **file.**

   Add an entry to */etc/passwd* file like the one shown in step 2 of Section 3.3.5, ''Creating a Default Account.'' Use a different user name, such as *priv_usr*, and make a note of the name you used. Assign it a UID of zero (0), and specify it's login directory and IRIX shell type.

2. **Edit the** ***/usr/etc/dn/servers.priv*** **file.**

   Enter the name of the privileged account next to the *priv* entry in *servers.priv*. This name must correspond to the account name in the */etc/passwd* file whose UID is zero. Figures 3-7 and 3-8 illustrate the *servers.priv* file before and after editing.

   ```
   null                          decnet
   priv                          decnet
   ```

   Figure 3-7 The *servers.priv* File Before Editing

   In Figure 3-7, the *null* entry in *servers.priv* specifies *decnet* as the name of the nonprivileged default account. It also specifies *decnet* (UID 998) as the privileged account, preventing incoming connections from privileged access to this workstation.

   ```
   null                          decnet
   priv                          priv_usr
   ```

   Figure 3-8 The *servers.priv* File After Editing

   In Figure 3-8, the nonprivileged account is still *decnet*, but the privileged account is specified *special-user*. Incoming connections to the *special-user* account will be allowed superuser access to this node, provided *special-user* is an account listed in */etc/passwd* with a UID of zero and the connection also contains the correct password.

When you complete the instructions in this section, your 4DDN node is ready to accept connections to the privileged account. Go on to the information in Section 3.4, ''Finalizing Software Setups.''

3.4 Finalizing Software Setups

Once you complete the instructions in Sections 3.3.1 through 3.3.5, a 4DDN node is ready to support most user services. However, before you use this node for sending mail or printing, you must complete the instructions in Chapter 4 to enable the mail and print utilities.

You can also do these optional procedures to finalize 4DDN software setup:

- Implement proxy accounts on the 4DDN node.

  4DDN nodes can be set up to provide proxy accounts that function like those on VAX/VMS systems. Chapter 7, ''Managing Nodes,'' contains instructions for implementing proxy accounts on an IRIS workstation.

- Include the 4DDN application in your Network License System.

  This version of 4DDN software works in conjunction with Silicon Graphics' Network License System product to automatically authorize user access to the 4DDN application. To use 4DDN with the Network License System, you need to add 4DDN to the License Server database. See your copy of the *Network License System Administration Guide* for instructions on this procedure.

Chapter 4

Enabling the Mail and Print Utilities

After you complete the configuration procedure in Chapter 3, you will need to enable 4DDN mail service before you can exchange mail with VAX/VMS systems. In addition, if you installed 4DDN software on an IRIS-4D workstation where the 4DDN default printer configuration files are not suitable, you will also need to modify the printer configuration file before using the 4DDN print utility.

This chapter explains how to enable 4DDN mail service. It also explains how to determine whether you need to modify your printer configuration file, and it gives procedures for doing the modifications if you determine they are necessary.

4.1 Enabling 4DDN Mail

4DDN mail service requires specific entries in your local mail configuration file */usr/lib/sendmail.cf*. To determine whether your current copy of *sendmail.cf* contains 4DDN entries, issue this command:

```
grep -i 4ddn /usr/lib/sendmail.cf
```

If the *grep* command returns *4ddn* character strings, then your copy of */usr/lib/sendmail.cf* contains the entries required for 4DDN mail service, as well as instructions for enabling the entries. To set up 4DDN mail service to your machine, follow the instructions in */usr/lib/sendmail.cf* for enabling 4DDN entries.

4.2 Using Alternative Mail Configuration Files

If you executed the *grep* command on your copy of */usr/lib/sendmail.cf* and it did not return *4ddn* character strings, you can enable 4DDN mail service in one of two ways:

- Add 4DDN entries to your current version of */usr/lib/sendmail.cf*.
- Replace your current version of */usr/lib/sendmail.cf* with a 4DDN default version.

The method you choose depends on whether you plan to run both 4DDN and TCP/IP mail, or only the 4DDN mail service.

4.2.1 Using 4DDN and TCP/IP Mail

If you plan to use both 4DDN mail and TCP/IP mail, and you have not modified your working version of *sendmail.cf*, you will need to add 4DDN entries to your current version of */usr/lib/sendmail.cf*. The entries you need to add and instructions for adding them are contained in the file */usr/etc/dn/sendmail.cf.4ddn*.

To add 4DDN entries to */usr/lib/sendmail.cf*, follow the procedure below.

Note:
When modifying *sendmail.cf*, keep in mind that the *sendmail* program is sensitive to blank spaces; extra spaces will invalidate your entries.

1. If you have not done so, log into the *root* account on your IRIS system. After your entry, the superuser prompt is displayed:

   ```
   su
   Password:      (password not echoed)
   #
   ```

2. Edit your copy of */usr/lib/sendmail.cf* to include the entries from */usr/etc/dn/sendmail.cf.4ddn*. Follow the instructions in */usr/etc/dn/sendmail.cf.4ddn* to add 4DDN entries.

3. Stop and restart the mail process:

```
/etc/init.d/mail stop

/etc/init.d/mail start
```

4DDN mail service is ready to use after you complete this procedure.

4.2.2 Using Only 4DDN Mail

4DDN is shipped with a default version of the *sendmail.cf* file, called *sendmail.cf.default+4ddn*, that can be used on systems where only 4DDN mail service is needed. If you plan to use only 4DDN mail and not TCP/IP mail, and you have not modified your working version of *sendmail.cf*, follow this procedure:

1. If you have not done so, log into the *root* account on your IRIS system. After your entry, the superuser prompt is displayed:

```
su
Password:      (password not echoed)
#
```

2. Replace your copy of *usr/lib/sendmail.cf* with the 4DDN default version:

```
cp /usr/etc/dn/sendmail.cf.default+4ddn /usr/lib/sendmail.cf
```

3. Stop and restart the mail process:

```
/etc/init.d/mail stop

/etc/init.d/mail start
```

4DDN mail service is ready to use after you complete this procedure.

4.3 Using Alternative Spooling Programs

The 4DDN printer utility *dnlp* assumes that your local print spooling program is the standard IRIS spooler */usr/bin/lp*. If you are not using */usr/bin/lp* as your local spooler, you need to modify the 4DDN printer configuration file */usr/etc/dn/dnlp.cf* to suit the spooler program you are using. Follow the instructions given in the following section to make the necessary changes.

4.4 Modifying the *dnlp* Configuration File

Since the *dnlp* command uses the services of your local spooling program, it must have information about that program to process print requests. *dnlp* gets this information from the configuration file *dnlp.cf*, located in the directory */usr/etc/dn*. A default version of */usr/etc/dn/dnlp.cf* is shipped with your 4DDN software.

The default version of *dnlp.cf* assumes that you are using the standard IRIS print spooler program *lp*, and that your printer has been installed and configured. */usr/etc/dn/dnlp.cf* contains the entries shown in Figure 4-1.

```
/usr/bin/lp
d<space>
```

Figure 4-1 Entries in *dnlp.cf*

The first line identifies the local spooling program that *dnlp* should use. In this case, that program is */usr/bin/lp*. The first column of line 2 contains the option flag that the spooling program uses to identify a destination printer in a command entry. For the *lp* program, the option flag is **d**. With this version of the *dnlp.cf* file in effect, *dnlp* issues the following command whenever you request a nondefault printer (the **-q** option) attached to your local node:

```
/usr/bin/lp -dprintername filename
```

If you are using a program other than */usr/bin/lp*, you will need to modify *dnlp.cf* for your spooler program. Follow the format of the as-shipped version of *dnlp.cf* when you make your changes:

1. Enter the name of your spooling program.

   Enter your spooler program name on line 1, using its full pathname.

2. Enter the nondefault printer option flag.

   Column 1 of line 2 must contain the option flag used by your spooling program to specify a nondefault printer. If, unlike */usr/bin/lp*, your spooling program does not support this option, enter a space in column 1.

3. Enter the file delete option flag.

   Column 2 of line 2 must contain the option flag used by your spooling program to specify that a file should be deleted after printing. If your spooling program does not support this option, enter a space in column 2.

The 4DDN print service is ready to use when you complete this procedure.

Chapter 5

Using the Network Control Program

The Network Control Program provides a set of interactive commands that allow you to configure, control, monitor, and test network components. NCP is a client process that requests services from a responding server called the Network Management Listener (NML). The node where NML is processing NCP commands is called the *executor node*. NML must reside on any node where you want to execute an NCP command.

This chapter explains how to enter and exit the Network Control Program. It also explains the elements of NCP commands, illustrates how to enter them, and describes the responses and error messages that result from NCP command entries.

5.1 Entering and Exiting NCP

To start NCP, type the *ncp* command shown below. After your entry, the NCP program prompt is displayed. (If you entered the */usr/etc/dn* directory in your IRIX path, you can omit the directory specification shown in this entry.)

```
% /usr/etc/dn/ncp <Return>
NCP>
```

To exit NCP, use the *exit* command at the NCP> prompt, as shown below.

```
NCP> EXIT
```

If it is necessary to leave NCP while a command is executing, you can do so by typing the **<Ctrl D>** key sequence.

5.2 Entering NCP Commands

The NCP commands available from 4DDN nodes are a subset of those available on DECnet-VMS nodes. NCP commands comprise four elements: the command verb, such as *SHOW*; a network entity that is the target of the command, such as a *NODE*; one or more parameters from a specified parameters list, such as the *STATE* of an entity; and a qualifier from a qualifier list (not required for most commands).

When you enter an NCP command, you order its elements so as to take a specified action on a designated entity. Some entities, such as *NODE* and *EXECUTOR*, have modifiers such as *NAME* and *ADDRESS*. They can also take DECnet node names and node addresses as arguments.

NCP commands are not case sensitive, except when you specify a user ID or password. Lowercase letters are automatically converted to uppercase internally.

Table 5-1 shows commonly used verbs, entities, and parameters for NCP commands.

Verbs	Entities	Parameters
CLEAR	CIRCUIT	CHARACTERISTICS
DEFINE	EXECUTOR	COUNTERS
DISCONNECT	KNOWN entity	STATE
EXIT	LINE	STATUS
LOOP	LINK(S)	SUMMARY
SET	NODE(S)	
SHOW		
TELL		
ZERO		

Table 5-1 NCP Command Elements

For more information on NCP commands and screen output, refer to the Digital Equipment Corporation publication, *VAX/VMS Network Control Program Reference Manual*.

5.2.1 Using Command Verbs

Use NCP command verbs to do the tasks listed below. Some commands in this list require superuser privileges.

CLEAR	Removes data from the volatile database on the local node or resets it to its default value. (Requires superuser privileges.)
DEFINE	Changes data in the permanent database. (Requires superuser privileges.)
EXIT	Terminates the NCP program.
LOOP	Transmits blocks of data to test network components.
SET	Changes data in the volatile database. (Requires superuser privileges.)
SHOW	Displays data about a network component.
TELL	Designates a remote node where an NCP command is to be executed.
ZERO	Sets counters to zero. (Requires superuser privileges.)

The examples below illustrate the effects of NCP commands:

```
NCP> SET EXECUTOR STATE ON
NCP> SHOW KNOWN NODES SUMMARY
NCP> ZERO KNOWN CIRCUIT COUNTERS
```

In the first example, the command verb *SET* operates on the *EXECUTOR* entity to set its *STATE* parameter to *ON*. This specifies the local NCP process as the executor. The second example displays a list of all nodes entered in the volatile database of the local node, and the third example assigns a value of zero to all circuit counters in the volatile database at the local node.

5.2.2 Entering Multiple-line Commands

Some NCP commands may require more than one line on the screen. To continue a command entry on the next line, use a hyphen, as shown below:

```
NCP> SHOW KNOWN LINKS -
SUMMARY
```

A space is required before the hyphen if it separates two words. To shorten your entries, you can use an abbreviation such as *CHAR* for *CHARACTERISTICS* or *COUNT* for *COUNTERS*.

5.2.3 Entering Individual NCP Commands

You can execute a single NCP command without starting the NCP program first. To do this, type **`NCP`** (the program command) followed by the interactive command, all on one line. This example illustrates such an entry:

```
% /usr/etc/dn/ncp SHOW EXECUTOR COUNTERS
```

5.3 NCP Responses

Depending on the command entered, NCP responds by displaying one of the following:

- The requested data after a *SHOW* command
- The prompt ''NCP>'' after a *SET* or *DEFINE* command
- An error message (described in Section 5.3.1) after a *SET* or *DEFINE* command.

See Appendix C for a listing of NCP error messages and their meanings.

Chapter 6

Managing the Database

The database needed to support DECnet operation on a 4DDN node contains information about network nodes and real-time information about other network entities and events. The DECnet database is first established from information in the */usr/etc/dn/nodes* file when you run the 4DDN configuration scripts. However, you usually need to add additional nodes to the database after 4DDN is running.

This chapter explains the organization of the database on a 4DDN node and how to use NCP commands to modify it. It also explains how to copy between real-time and permanent versions of the database to simplify network configuration and maintenance.

6.1 The Configuration Database

The body of information that defines the network to a particular node and stores data about network conditions and events is known as the *configuration database*. The configuration database comprise two separate databases, the *permanent database* and the *volatile database*.

6.1.1 The Permanent Database

The *permanent database* contains the names and addresses of reachable nodes in the network and initialization values for the volatile database on the local node. At 4DDN startup, the permanent database is copied to the volatile database, which is used during network operation. Files that contain the permanent database are stored on disk in the directory */usr/etc/dn*.

6.1.2 The Volatile Database

The volatile database resides in memory and controls the operation of the network. This database is automatically generated when 4DDN is started, and it is changed dynamically as network conditions change. Changes to values in the volatile database do not automatically cause changes in the permanent database. To copy volatile database values to the permanent database, you must issue an NCP command.

The parameters in the volatile database can be changed with NCP commands while the network is running. These changes remain in effect until further modifications are made or until the network is shut down.

6.2 Building the Database

The preliminary permanent database built when 4DDN is configured might require expanding to make the node fully operational. The sections that follow explain how to build an expanded database, using NCP commands. These commands can also be used whenever you need to modify database information.

Before you use the information in these sections, you might want to check the */usr/etc/dn/nodes* on your node for node name and address information.

6.2.1 Identifying the Executor Node

The local *executor node*, the network node responsible for invoking NCP commands in your local node, must be identified in the local node's database. During the installation procedure, the executor node name and address are entered in the permanent database. You can use NCP commands to establish these parameters in the volatile database.

To establish the executor node address, use the command shown below. In this entry, *aa* is the area number whose value is in the range of 1 to 63; and *nnnn* is the node number whose value is in the range of 1 to 1023 (see Section 2.3.2, ''DECnet Node Identification'' for more information).

```
NCP> SET EXECUTOR ADDRESS aa.nnnn
```

This command assigns the node address to the local executor node and must be the first command issued when setting executor node parameters.

Note: When an executor's address has been defined in the volatile database, it can be changed only by setting the node's state to *OFF* and restarting the node.

To establish the executor node name, use the command shown below. A node name is optional if the node address is already defined; however, executor node names are generally assigned in addition to addresses.

```
NCP> SET EXECUTOR NAME saturn
```

This command assigns the local node name *saturn* to the executor.

6.2.2 Identifying Remote Nodes

In addition to establishing the executor node, you must also identify remote nodes in the configuration database. Some remote nodes are entered in the permanent database during the configuration procedure. However, you can use NCP commands to identify remote node names in the volatile database.

To identify a remote node in the volatile database, use the command shown below.

```
NCP> SET NODE 63.1 NAME sam
```

This command sets the address (*63.1*) and name (*sam*) of the remote node in the volatile database. Remember that this entry is not permanent; it will be lost when the 4DDN application is shut down. (For additional information on the *SET NODE* command, see Chapter 9, ''Testing the Network.'')

6.2.3 Making Your Entries Permanent

The NCP commands in the previous sections enter data in the volatile database. The data will not be retained when a station is shut down. If you want to store these values in the permanent database, use the information in Section 6.3 to copy the contents of the volatile database to the permanent database.

6.3 Copying between Databases

You can use NCP commands to copy the contents of the permanent and volatile databases between one another. Instructions for using these commands are given in the sections below.

6.3.1 From Volatile to Permanent

The *DEFINE* command copies entries from the volatile database to the permanent database. The form of the command you use depends on whether you are copying from the volatile database to the permanent database on a single node or to all nodes known to your local executor node.

To copy information for a single node, use this command:

```
NCP> DEFINE NODE saturn ALL
```

This command copies the executor's volatile data about node *saturn* to the permanent database.

To copy information for all known nodes, use this command:

```
NCP> DEFINE KNOWN NODES ALL
```

This command copies, for each of the known nodes, the content of the volatile database to the permanent database.

6.3.2 From Permanent to Volatile

The *SET* command copies entries from the permanent database to the volatile database. The form of the command you use depends on whether you are specifying a single node or all nodes known to your local executor node.

To copy information for a single node, use this command:

```
NCP> SET NODE saturn ALL
```

This command copies the executor's permanent database about node *saturn* to the volatile database.

To copy information for all known nodes, use this command:

```
NCP> SET KNOWN NODES ALL
```

This command copies, for each of the known nodes, the contents of the permanent database to the volatile database.

Note: When the *dnstart.sh* script is executed, parameter values are automatically copied from the permanent database to the volatile database.

Chapter 7

Managing Nodes

This chapter contains information needed to manage local and remote nodes on the DECnet network using 4DDN tools. It describes how to handle network accounts and proxy logins, how to assign node names and addresses, and how to use NCP commands to control the executor and remote nodes in the network.

In addition to the commands described in this chapter, 4DDN provides other commands for node management. These are described in Chapter 10, ''NCP Reference Guide.'' The Network Management Listener must be present on the target node before any NCP command can take effect.

7.1 Maintaining Network Accounts

Recall from Chapter 3 that 4DDN nodes have two special accounts for handling special types of network connections: the default account, which is used for all incoming connections with no access control information; and a privileged account, which is used for an incoming connection from a remote user who requires superuser privileges. The name of the default account and privileged account are specified in the */usr/etc/dn/servers.priv* file. *servers.priv* contains two entries: *null*, which specifies the default account name; and *priv*, which specifies the privileged account name.

Both VMS and 4DDN nodes pass null access control information when no access information is specified in the connection request. When such a request arrives at a 4DDN node, the *dnserver* process checks the *null* entry in the */usr/etc/dn/servers.priv* file for the name of a default account. If the *null* entry specifies a valid account on the IRIS (an account listed in */etc/passwd*), *dnserver* accepts the connection and starts the appropriate server. The default account is accessed without any password verification.

Note: As an option, 4DDN node users can set their own default access control values using the *.4ddnrc* file (see Chapter 3 of the *4DDN User's Guide* for details). In additon, VMS systems can be set to provide default access control information in outgoing connection requests (see Section 7.2, ''Controlling Access from VMS Nodes'').

To access a 4DDN node with superuser privileges, an incoming connection request must contain both a user name and password. When such a request arrives, *dnserver* checks */etc/password* for the UID associated with the user name. If the UID is zero, *dnserver* checks the *priv* entry in */usr/etc/dn/servers.priv* for a match to the user name entered in the connection. If the names match, *dnserver* checks */etc/password* for a password match. If the passwords match, the incoming connection is granted superuser access to the 4DDN node.

The procedures in the sections that follow review the procedure for setting up a default non-privileged account and privileged account on an IRIS system. They assume you have already set up these accounts, using the instructions in Chater 3. If you are setting up these accounts for the first time, or if you need more details, refer to 3.3.5 ''Creating a Default Account'' and 3.3.6, ''Creating a Privileged Account.''

7.1.1 Changing the Default Non-privileged Account

In the configuration procedure (Section 3.3.5), you edited the IRIX file */etc/passwd* file to establish a non-privileged default account called *decnet* and set the password for the account to *DECNET*. Since the as-shipped version of the */usr/etc/dn/servers.priv* file already specified *decnet* as the name of the default account, you were not required to edit the *servers.priv* file in that procedure. The procedure below explains how to change *servers.priv* and */etc/passwd* to change the name and password the on default account.

1. **Open */usr/etc/dn/servers.priv*.**

2. **Change the *null* entry in *servers.priv*.**

   Replace the *decnet* argument to the *null* entry of *servers.priv* with the new name for the default nonprivileged account.

3. **Update the */etc/passwd* file.**

   Replace the *decnet* account name in */etc/password* with the new name for the default account.

4. **Set the new password with the *passwd* command.**

   Since VMS systems use uppercase, type the password in uppercase letters when you make your entries. See the *passwd(1)* man page if you need help with this command.

7.1.2 Changing Privileged Accounts

As an optional part of the configuration procedure (Section 3.3.6), you established a privileged account to allow superuser access to the IRIS node. The procedure below explains how to change *servers.priv* and */etc/passwd* files to change the name and password on the privileged account.

1. **As superuser, open */usr/etc/dn/servers.priv*.**

2. **Change the argument to the *priv* entry in *servers.priv*.**

   The account name you enter must be listed in */etc/passwd* with a UID of zero (0).

3. **Update the */etc/passwd* file.**

4. **Set the new password with the *passwd* command.**

7.2 Controlling Access from VMS Nodes

On VMS nodes, you can specify values in the NCP database so that access control information for default accounts is supplied in outgoing connection requests. If user entries contain no access control information, the VMS client fills in your specified default information before sending the request to the destination node.

VMS systems make two types of default accounts available, privileged and nonprivileged accounts, and you can specify default access control information for either type of account. Although privileged accounts are optional, it is important to recognize that certain NCP functions cannot be executed from a 4DDN node that contains no privileged account.

The following instructions explain how to set default access control information on a VMS system so outgoing requests connect to a nonprivileged and privileged account on a remote node. These procedures assume that you are issuing the commands from the local VMS host where you want the default values to take effect.

7.2.1 Setting Nonprivileged Account Defaults

Issue NCP commands, like the ones below, on the VMS host to set up its default access control information for nonprivileged user accounts. The values you assign to the user name and password must correspond to those you set on the receiving 4DDN node (see Section 7.1 for details).

```
NCP> SET NODE iris_nodename NONPRIV USER default_account_name
NCP> SET NODE iris_nodename NONPRIV PASSWORD password
```

Assume that, in the command sequences above, you specified *PLUTO* as the 4DDN host name, *james* as the NONPRIV USER account, and *BOND* as the NONPRIV PASSWORD on the account. In this case, these user commands issued from a VMS node are equivalent:

```
$ DIR pluto::usr:[bin]
$ DIR  pluto"james BOND"::usr:[bin]
```

In the first example, the user entry contains no access control information, so the sending VMS node fills in the default values for the user name and password (*james* and *BOND*). In the second example, access control information entered by the user in the file specification is processed on the receiving node.

7.2.2 Setting Privileged Account Defaults

On the VMS host, issue NCP commands like the ones below to set up its privileged user account. The values you assign to the user name and password must correspond to those you set up on the receiving 4DDN node (see Section 7.1 for details).

```
NCP> SET NODE iris_nodename PRIV USER default_account_name
NCP> SET NODE iris_nodename PRIV PASSWORD password
```

7.3 Using Proxy Accounts

Incoming connections that do not contain access control information need not be logged into the default DECnet account on your 4DDN node. As an alternative, you can set up proxy accounts.

Proxy accounts allow designated remote users to access specified accounts, without entering access control information. In addition to providing convenience to users, proxy accounts offer two other advantages: they reduce the overhead of maintaining user-account-password lists on individual computer nodes; and they increase security, since passwords to sensitive accounts are not disclosed in connection requests where they might be read by unauthorized users.

The 4DDN implementation of the proxy feature is similar to the VMS implementation, with one exception. VMS systems provide user names in the proxy field of outgoing requests; 4DDN provides a user identification number (*UID*) in outgoing requests. The UID is a code that identifies the user by number rather than by name.

7.3.1 Access to Proxy Accounts

When proxy accounts exist on a 4DDN node, incoming connection requests that do not contain passwords are validated in the proxy configuration file (see Section 7.3.2). Authorized users who do not enter access control information are logged into their *default proxy account*. Authorized users who specify the user name (**-u**) for a particular proxy account (**-u** *proxy_username*) are logged into the proxy account they named. Proxy accounts designated by name are called *secondary proxy accounts*. 4DDN supports up to 15 secondary proxy accounts.

Table 7-1 shows how incoming connections to a 4DDN node are assigned to accounts, based on access control information entered by a user and the existence of proxy accounts on the node.

User name in entry?	Password in entry?	Proxy accounts?	Connection Events
No	No	No	Use account specified by the *null* entry of *servers.priv* file.
Yes	Yes	No	Use standard validation process.
Yes	Yes	Yes	Ignore *proxy.cf*. Use standard validation process.
No	No	Yes	Check *proxy.cf* for match with proxy UID (4DDN sender) or proxy field name (VMS sender). If match, use sender's default proxy account. If no match, use the account specified by the *null* entry in *servers.priv* file.
Yes	No	Yes	Check *proxy.cf* for match with user name in access control field. If match, use proxy account (default or secondary) user specified. If no match, use standard validation.

Table 7-1 Connections to Default Accounts

7.3.2 Proxy Configuration File Format

On 4DDN nodes, proxy information is maintained in the file */usr/etc/dn/proxy.cf*. This file contains a list of authorized users on remote nodes and specifies the accounts to which each user has proxy access. Each line in *proxy.cf* has this format:

```
remote_user:remote_node:local_default_account:secondary_account,...
```

Fields in *proxy.cf* have these meanings:

remote_user	The name or user identification (UID) of the user on the remote system requesting the connection. VMS systems pass a user name in connection requests while IRIX systems pass a UID.
remote_node	The name of the network node where the request originated.
local_default_account	The name of the proxy account to be accessed when this user enters no access control information.
secondary_account	The names of other proxy accounts this user can access by entering the **-u** option in the access control string. A maximum of 15 secondary proxy accounts can be listed. Separate account names with commas.

7.3.3 Editing the Proxy File

The procedure below explains how to edit the *proxy.cf* file to configure proxy accounts on a 4DDN node. To do this procedure, you need to know the VMS user name or IRIS UID of each remote user to be granted proxy account access, and the network name of the remote host where these users' connections will originate.

1. **Change to the *****/usr/etc/dn***** directory.**

   Use this command to change to the directory where *proxy.cf* is stored:

   ```
   # cd /usr/etc/dn
   ```

2. **Edit the *proxy.cf* file.**

   Replace the dummy values in the as-shipped version of *proxy.cf* with legitimate remote user names, remote host names, and local proxy account names. You should have one entry in *proxy.cf* for each user authorized to use proxy accounts on this 4DDN node.

7.4 Setting Node Names and Addresses

In NCP commands, you specify the name of a remote DECnet node by the *NODE NAME* parameter and specify its address by the *NODE ADDRESS* parameter. You specify the name of the local node by the *EXECUTOR NAME* parameter and specify its address by the *EXECUTOR ADDRESS* parameter.

Use the *SET* command to change node names and addresses. Whenever you change node names and addresses, follow these name and address rules:

- Set the executor state to *OFF* before changing the address of the local executor node.

- Do not mix local and remote node parameters in one command.

These commands show how to set name and address parameters:

```
NCP> SET EXECUTOR NAME mynode
NCP> SET NODE 44.12 NAME node5

NCP> SET EXECUTOR ADDRESS 2.31
NCP> SET NODE node5 ADDRESS 15.3
```

You can also combine parameters to assign names and address, as shown in this command:

```
NCP> SET EXECUTOR NAME mynode ADDRESS 2.31
```

7.5 Managing the Executor Node

Two major processes support network management functions: the NCP program, which requests services; and the Network Management Listener, which executes NCP commands in incoming requests. It is not necessary that both NCP and NML reside on every node in the network. However, NML must reside on any node that will function as an executor node.

7.5.1 Changing the Executor Node

When NCP starts, the local node is the executor, and it remains the executor until you change it. You can specify a remote node as the executor by issuing the *SET EXECUTOR* command, as shown in this example:

```
NCP> SET EXECUTOR NODE pluto
```

NCP commands entered at the local node are executed at the remote executor node, *pluto*. Each command is interpreted as if issued at the executor node. However, any display of information that results from executing a command is displayed at your local node.

To reset the executor to the local node, use the following NCP command:

```
NCP> CLEAR EXECUTOR NODE
```

7.5.2 Setting the Executor State

The executor state specifies the operating status of the executor node. Normally, the state is ON, allowing new logical links to be created. However, it is sometimes necessary to change the executor state to OFF. The OFF state prevents creation of new links, terminates existing links, and shuts down the node to network traffic.

Use the *SET* command to change the executor state, as shown below:

```
NCP> SET EXECUTOR STATE OFF
NCP> SET EXECUTOR STATE ON
```

The first command disables incoming and outgoing connections at the node. The second command restores the node to network participation, using the values in the volatile database to initialize the node.

Note: If you set the executor state to *off* on a remote network node, it breaks the link from your local node, preventing any further communication with the remote node. This condition is usually undesirable.

7.5.3 Setting the Executor Segment Buffer Size

In DNA networks, the Network Services Protocol (NSP) running in the End Communications Layer partitions and reassembles message packets transmitted over the network. The size of packets passed between two communicating end nodes is limited by the size of the transmit (or segment) buffers of an intermediate router node servicing the connection. If either end node transmits a packet that exceeds the router's segment size, the message is lost.

To prevent packet loss, you can set the size of the segment buffer on the executor node to a size that is acceptable to the router node. You must set the executor state to *off* before using this command:

```
NCP> SET EXECUTOR SEGMENT BUFFER SIZE xxxx
```

This command specifies (in bytes) the maximum size of transmit buffers, to control the size of an NSP message segment transmission. Use a value in the range of 255 to 1458 bytes. The default value is 1458.

Note: Most routers cannot route packets longer than 512 bytes. If you are experiencing packet loss and you suspect a router, set the segment buffer size to 512 bytes.

7.6 Managing Remote Nodes

You can use remote network management to connect to any node in the network, examine its parameters and counters, zero its counters, and perform loop testing. You perform remote node management in two ways: by designating the remote node as the executor, or by using *TELL* as a prefix to NCP commands.

7.6.1 Designating a Remote Executor

To designate a remote node as the executor, use this NCP command:

```
NCP> SET EXECUTOR NODE node-id
```

The *node-id* is either the node name or its address. Once this command takes effect, NCP forwards any commands it receives to the node specified in *node-id* until you type a new *SET EXECUTOR NODE* or *CLEAR EXECUTOR NODE* command.

7.6.2 Using the *TELL* Prefix

Instead of designating the remote node as the executor, you can use the *TELL* prefix to identify the executor node for a particular NCP command. *TELL* sets the executor for only one command, and it must prefix the command for which it is intended. This example illustrates how to use the *TELL* command prefix.

```
NCP> TELL pluto SHOW EXECUTOR CHAR
```

This command sets the executor to node *pluto* where *SHOW EXECUTOR CHAR* executes.

7.7 Resetting Node Counters

4DDN collects statistical information about network traffic between the local node and other nodes. This information includes the number of bytes or messages sent or received. You can display this information using the *SHOW* command (see Chapter 10, ''NCP Command Reference'').

Use the *ZERO* command to reset counters for all known nodes or for a given node:

```
NCP> ZERO KNOWN NODES COUNTERS
NCP> ZERO NODE pluto COUNTERS
NCP> ZERO EXECUTOR COUNTERS
```

7.8 Clearing a Specified Node

The *CLEAR NODE* command clears all parameters for a specified node from the volatile database. Use this command with caution, particularly if you are clearing active links to the node. Before you use this command, issue a *SET EXECUTOR STATE OFF* or *DISCONNECT KNOWN LINKS* command.

```
NCP> CLEAR NODE node-id ALL
```

Chapter 8

Managing Circuits, Lines, and Links

This chapter describes some commonly used NCP commands for managing the circuits, lines, and logical links in the DECnet network. The commands described in this chapter are described in more detail in Chapter 10, ‘‘NCP Command Reference.’’ Chapter 10 also describes additional circuit, line, and link commands.

8.1 Managing Circuits

4DDN currently supports only a single Ethernet circuit. The name of a circuit is based on the controller type. The output of the *SHOW* commands identifies circuits using the following format:

```
dev-c
```

where:

dev	is a device name, such as UNA on a VAX, or ENP on some IRIS-4Ds.
c	is an empty string or a decimal number (0 or a positive integer) identifying the controller for the device.

8.2 Resetting Circuit Counters

4DDN maintains statistical information about circuit performance. This information includes the number of data packets sent, the number of data packets received or lost, timeouts, and the time elapsed since counters were last zeroed.

When the network is running, you can reset counters to zero for known circuits:

```
NCP> ZERO KNOWN CIRCUIT COUNTERS
```

For a detailed description of circuit counters, see Section 10.20, ‘‘Show Known Circuit,’’ in Chapter 10.

8.3 Managing Lines

4DDN currently supports a single Ethernet line. The line name is based on the controller type. The output of the *SHOW* command identifies lines using the circuit format described above.

8.4 Resetting Line Counters

4DDN maintains statistical information on line performance. This information includes the number of bytes, data blocks sent or received, local and remote process errors, and time elapsed since the counters were last zeroed.

When the network is running, counters can be reset to zero for the known line:

```
NCP> ZERO KNOWN LINE COUNTERS
```

For a detailed description of line counters, see Section 10.21, ‘‘Show Known Line,’’ in Chapter 10.

8.5 Monitoring Links

4DDN software provides a set of commands that lets you monitor logical links at a network node and disconnect them. Use the *SHOW* command whenever you want information about active links at a given node. The *DISCONNECT* commands are used primarily for recovering from network failures; use these commands judiciously, since any application will terminate abnormally if it is using a link you disconnect.

Use this command to display information on the links currently active on a local node:

```
NCP> SHOW KNOWN LINKS SUMMARY
```

For a detailed description of link summaries, see Section 10.22, ‘‘Show Known Links,’’ in Chapter 10.

8.6 Disconnecting Links

Use the command below to disconnect a particular logical link at a local node. When you use this command, you must specify the number of the link that you want to disconnect. Use the link number given in the *SHOW KNOWN LINK SUMMARY* command to determine the link number.

```
NCP> DISCONNECT LINK link-id
```

Use this command to disconnect all logical links at a local node:

```
NCP> DISCONNECT KNOWN LINKS
```

Caution: Any application using a link you disconnect will terminate abnormally.

Chapter 9

Testing the Network

NCP provides the *LOOP* command to help you determine whether the network is operating properly. Node-level loopback tests enable you to evaluate the operation of logical links and network software by exchanging test data between processes on two different nodes or between processes in different layers of the same node.

If a test completes successfully, data messages loop back to the source without any change. If a test fails, messages either do not return to the source or they return in a corrupted state. When a failure occurs, NCP displays an error message indicating a failure, specifying the reason for the failure and providing a count of the data messages that were not returned.

The loopback mirror resides above the Session layer. For local tests, it allows checking of the Session Layer, End Communication Layer, and Routing Layer. For local-to-remote tests, it also checks the Physical Layer and Data Link Layer.

9.1 Loop Parameters

You can execute the *LOOP* command with the parameters listed below. For a more complete description of using the *LOOP* command, see descriptions of the *LOOP* command in Chapter 10, ‘‘NCP Command Reference.’’

COUNT	Specifies the number of blocks sent during loopback testing over the line or node. This parameter must be an integer in the 1–65535 range. The default value is 1.
LENGTH	Specifies in bytes the length of blocks to be sent during loopback testing. This parameter must be an integer in the 1–65535 range, but it must not exceed the maximum segment buffer size. The default value is 40.
WITH	Specifies the type of binary information to be sent during testing. The options are: MIXED, ONES, and ZEROS. MIXED is the default value.
USER	Specifies the user ID for access control information.
PASSWORD	Specifies the password that corresponds to the user’s account.

9.2 Local Loopback Test

The local loopback test evaluates local 4DDN software, using an internal logical link path. To do local loopback testing, issue this command:

```
NCP> LOOP EXECUTOR COUNT 5 LENGTH 40 WITH zeros
```

Figure 9-1 illustrates a local loopback test.

Figure 9-1 Testing a Local Logical Link

9.3 Remote Loopback Test

The *LOOP NODE* command tests the logical link connection between two nodes. To use *LOOP NODE*, you must have previously identified the remote node using the *SET NODE* command.

For example, this command checks the connection to a remote VMS system named *pluto*:

```
NCP> LOOP NODE pluto COUNT 5 LEN 40 WITH zeros -
     USER james PASSWORD bond
```

Figure 9-2 illustrates a local-to-remote loopback test.

Figure 9-2 Testing a Local-to-Remote Logical Link

Figure 9-3 shows the protocol layers checked in the test.

Figure 9-3 Protocols Tested by Loopback Commands

Chapter 10

NCP Command Reference

This appendix describes all NCP commands 4DDN currently supports, listed in alphabetical order for quick reference. Each description explains the function, format, and, if necessary, the output of the command. Command syntax is described in Chapter 4, *Using the Network Control Program*.

Note: If you use the SET EXECUTOR NODE command to run NCP remotely on another DECnet system, the output of the NCP commands might differ from the output described here. The remote system might also support NCP commands that are not described here. Refer to the appropriate DECnet documentation for more information.

10.1 CLEAR EXECUTOR NODE

This command sets the executor back to the local node. It clears the default executor node designation previously specified through the *SET EXECUTOR NODE* command.

10.2 CLEAR NODE ALL

This command clears all parameters for a specified node from the volatile database.

Format:

```
NCP> CLEAR NODE node-id ALL
```

where:

node-id	is the address or name of the node to be cleared.

Caution: Can result in loss of data over active links, since active links are cleared from the volatile database. Before invoking this command, issue a *DISCONNECT LINK* or *SET EXECUTOR STATE OFF* command.

10.3 DEFINE EXECUTOR ALL

This command copies the contents of the executor's volatile database to the permanent database.

10.4 DEFINE KNOWN NODES ALL

This command copies, for each of the known nodes, the content of each volatile database record to the permanent database.

10.5 DEFINE NODE ALL

This command copies, for the specified node, the content of the volatile database to the permanent database.

Format:

```
DEFINE NODE node-id ALL
```

10.6 DISCONNECT KNOWN LINKS

This command disconnects all links to the local node. It is used primarily to recover from network failure.

Caution: Any applications using the links you disconnect will terminate abnormally.

10.7 DISCONNECT LINK

This command disconnects a specified logical link at a local node. It is used primarily to recover from network failure.

Format:

```
DISCONNECT LINK link-id
```

where:

link-id	is the number of the link to be disconnected. Use the *SHOW LINK* command to obtain the link ID.

Caution: Any application using the link will terminate abnormally when the link is disconnected.

10.8 EXIT

This command exits NCP. Typing ˆD (control-D) also exits NCP.

10.9 LOOP EXECUTOR

This command tests the logical links within a single node. This test is performed by looping messages to the loopback mirror on the local node. Note that the count, length, and block-type are optional, but when used, must be used in the specified order, i.e., count, followed by length, followed by block-type.

Format:

```
LOOP EXECUTOR [COUNT count] [LENGTH length]
 [WITH block-type] [USER user-id PASSWORD password]
```

where:

count	specifies the number of times the command is be repeated. Each iteration transmits one block of data. The default is 1 and the maximum is 65,535.
length	specifies the number of bytes in the loop message. The default value is 40 and the maximum length is the maximum segment buffer size, not to execeed 65,535. Display maximum segment buffer size with *SHOW EXEC CHAR* command.
block-type	specifies the test data. Three options are available: ZEROS ONES MIXED (default value)
user-id	specifies the user name in the access control information used to connect to the remote mirror. Not validated if the remote node is a 4DDN node.
password	specifies the password that corresponds to the specified user-id. Not validated if the remote node is a 4DDN node.

10.10 LOOP NODE

This command tests the logical links to a remote node. Note that the count, length, and block-type are optional, but when used, must be used in the specified order, i.e., count, followed by length, followed by block-type.

Format:

```
LOOP NODE node-id [COUNT count] [LENGTH length]
 [WITH block-type] [USER user-id PASSWORD password]
```

where:

node-id	identifies the node (name or address).
count	specifies the number of times the command is repeated. The default is 1 and the maximum is 65,535.
length	specifies the number of bytes in the loop message. The default value is 40 and the maximum length is the maximum segment buffer size, not to exceed 65,535. Display maximum segment buffer size with the *SHOW EXEC CHAR* command.
block-type	specifies the test data. Three options are available: ZEROS ONES MIXED (default value)
user-id	specifies the user name to be used for access control information in connecting to the remote mirror. Not validated if the remote node is a 4DDN node.
password	specifies the password that corresponds to the specified user-id. Not validated if the remote node is a 4DDN node.

10.11 SET EXECUTOR ADDRESS

This command specifies a new node address for the executor node. This command can be issued only if the state of the executor is *OFF*. It is not possible to issue NCP commands to a remote executor node when the state of that node is *OFF*. Therefore, the executor node must be the local node when this command is issued.

Format:

```
SET EXECUTOR ADDRESS node-address
```

where:

node-address	specifies the node address for the local executor node.

10.12 SET EXECUTOR ALL

This command copies the content of the permanent database to the volatile database for the executor node.

10.13 SET EXECUTOR NAME

This command specifies a new node name for the local executor node. (The restrictions for SET EXEC ADDRESS apply to this command, too).

Format:

```
SET EXECUTOR NAME node-name
```

where:

node-name	specifies a node name for the executor node. The node name is a string of up to six alphanumeric characters that contains at least one alphabetic character.

10.14 SET EXECUTOR NODE

This command specifies the local node or a remote node as the executor for all subsequent NCP commands. As an alternative, you can use the *TELL* command prefix to identify the executor node for a particular NCP command.

Format:

```
SET EXECUTOR NODE node-id [USER user-id PASSWORD password]
```

where:

node-id	identifies the local or remote node (name or address).
user-id	identifies user ID on the remote system to use for the connection. Validated on remote 4DDN and VMS nodes.
password	the password that is associated with this user-id. Validated on remote 4DDN and VMS nodes.

Note: The user-id and password are optional. If they are not specified, then the default user-id and password are used.

You can use the *CLEAR EXECUTOR NODE* command to reset the executor to the local node.

10.15 SET EXECUTOR SEGMENT BUFFER SIZE

When a logical link is established, the communicating nodes negotiate the segment buffer size. They agree on the smaller of the message segment sizes used by the Network Services Protocol in each node. However, if an intermediate router in the circuit has a segment buffer size that is smaller than the negotiated size (such as one handling Digital Data Communications Protocol and Ethernet circuits), the logical link is broken and data is lost.

This command specifies in bytes the maximum size of transmit buffers to control the maximum size of a Network Services Protocl (NSP) message segment. The executor state must be set to *OFF* when you issue this command.

Format:

```
SET EXECUTOR SEGMENT BUFFER SIZE xxxx
```

where:

xxxx	is the maximum number of bytes in the buffer. This value must be in the range of 255 to 1458. The default value is 1458.

10.16 SET EXECUTOR STATE

This command turns the state of the node off and on. When the node state is on, the node is reachable from other network nodes, i.e., new logical links to that node can be created. When the node state is off, the node is unreachable.

Format:

```
SET EXECUTOR STATE node-state
```

where:

node-state is one of two options:

OFF	Prevents creation of new logical links, terminates existing link, and shuts down the node.

ON	Allows creation of new logical links. On is the normal operational state of a node.

Caution: Setting the state off terminates any active links to the node and could result in loss of data. Applications using the links terminate abnormally.

10.17 SET KNOWN NODES ALL

This command copies the content of the permanent database to the volatile database for all known nodes in the network.

10.18 SET NODE

This command specifies node names and addresses when building or adding to the executor's volatile configuration database.

Formats:

```
SET NODE node-id ADDRESS node-address
SET NODE node-id NAME node-name
SET NODE node-id ADDRESS node-address NAME node-name
```

where:

node-id	identifies the remote node to be added to the volatile database. Node IDs may be either a name or an address.
node-address	specifies the address of the node you want to include in the configuration database.
node-name	specifies the name of the node you want to include in the configuration database. Only one name can be assigned to a node address. Duplicate node names are not permitted.

For example,

```
SET NODE 1.7 NAME SATURN
```

creates an entry for the node SATURN at address 1.7. If *node 1.7* already exists with a different name, this command renames it.

10.19 SET NODE ALL

This command copies the content of the permanent database to the volatile database for a single node.

Format:

```
SET NODE node-id ALL
```

where:

node-id	specifies the local or remote node.

10.20 SHOW KNOWN CIRCUIT

This command displays static line information.

Format 1: Displaying Circuit Characteristics

```
SHOW KNOWN CIRCUIT CHARACTERISTICS
```

Description of Characteristics

- State – Operational state of a circuit. There are two circuit states: OFF and ON.
- Designated router – Node that is used for routing to non-adjacent nodes.

- Hello timer – Frequency in seconds of routing layer hello messages sent to the adjacent node(s) on the circuit.
- Type – Circuit type: ethernet.
- Adjacent node – Adjacent node on the circuit. On an ethernet circuit, there can be several adjacent nodes.
- Listen timer – Maximum time allowed before a message is received from an adjacent node on the circuit. If this time is exceeded, the node is declared unreachable.

Output Format

```
NCP> SHOW KNOWN CIRC CHAR

Known Circuit Volatile Characteristics as of Mon Jul 2 12:00:00 1990

Circuit = ENP

State              = on
Designated router  = 1.1 (VENUS)
Hello timer        = 15              (Display for 1st Node)
Type               = Ethernet
Adjacent node      = 1.1 (VENUS)
Listen timer       = 15

Circuit = ENP

Adjacent node      = 1.2 (TOPCAT)    (Display for all other nodes)
Listen timer       = 15
```

Format 2: Displaying Circuit Counters

```
SHOW KNOWN CIRCUIT COUNTERS
```

Description of Counters

- Seconds since last zeroed – Time elapsed since counters were zeroed for the last time.
- Bytes sent – Number of bytes of data sent by the local node over the circuit.
- Bytes received – Number of bytes of data received by the local node over the circuit.
- Data blocks sent – Number of data blocks sent by the local node.
- Data blocks received – Number of data blocks received by the local node.

Output Format

```
NCP> SHOW KNOW CIRC COUNT

Known Circuit Volatile Counters as of Mon Jul 2 12:00:00 1990

Circuit = ENP

 14971   Seconds since last zeroed
  3906   Data blocks sent
  3049   Data blocks received
340969   Bytes sent
452334   Bytes received
```

Format 3: Displaying Circuit Status

```
SHOW KNOWN CIRCUIT STATUS
```

Description of the Display

- Circuit ID – Identifies a particular circuit for which the information is displayed.
- Circuit current state – ON or OFF.
- Address and name of adjacent nodes on that circuit.
- Block size.

Output

```
NCP> SHOW KNOWN CIRCUIT STATUS
```

```
Known Circuit Volatile Status as of Mon Jul 2 12:00:00 1990

Circuit    State    Loopback    Adjacent        Block
                    Name        Node            Size
ENP        on                   1.1 (VENUS)
ENP                             1.2 (TOPCAT)
```

Format 4: Displaying Circuit Summary

```
SHOW KNOWN CIRCUIT SUMMARY
```

Description of the Display

- Circuit ID – Identifies a particular circuit for which the information is displayed.
- Circuit current state – ON or OFF.
- Address and name of adjacent nodes on that circuit.

Output

```
NCP> SHOW KNOWN CIRCUIT SUMMARY

Known Circuit Volatile Summary as of Mon Jul 2 12:00:00 1990

Circuit    State    Loopback    Adjacent
                    Name        Node
ENP        on                   1.1 (VENUS)
ENP                             1.2 (TOPCAT)
```

10.21 SHOW KNOWN LINE

This command displays the line information.

Format 1: Displaying Line Characteristics

```
SHOW KNOWN LINE CHARACTERISTICS
```

Description of the display

- Protocol
- Hardware address

Output

```
NCP> SHOW KNOWN LINE CHAR
```

```
Known Line volatile characteristics as of Mon Jul 2 12:00:00 1990

Line = ENP

Protocol             = Ethernet
Hardware address     = aa-00-04-00-04-04
```

Format 2: Displaying Line Counters

```
SHOW KNOWN LINE COUNTERS
```

Description of Line Counters

- Seconds since last zeroed – Number of seconds elapsed since the line counters were zeroed.
- Data blocks received – Number of data blocks received over the line.
- Data blocks sent – Number of data blocks sent over the line.
- Bytes sent – Number of bytes sent over the line.
- Bytes received – Number of bytes received over the line.
- Blocks sent, multiple collisions – Number of times that a frame was successfully transmitted on the third or later attempt after normal collisions on previous attempts.
- Collision detect check failure – Number of collision detect failures not sensed after a transmission.
- System buffer unavailable – Number of times no data link buffer was available for an incoming frame.

Output

```
NCP> SHOW KNOW LINE COUNT
```

```
Known Line volatile Counters as of Mon Jul 2 12:00:00 1990

Line = ENP

>65534    Seconds since last zeroed
  76622    Data blocks sent
  36046    Data blocks received
```

```
397468   Bytes sent
271671   Bytes received
     0   Blocks sent, multiple collisions
     0   Collision detect check failure
     0   Unrecognized frame destination
     0   System buffer unavailable
```

Format 3: Displaying Line Status

```
SHOW KNOWN LINES STATUS
```

Description of the Display

- Line name (e.g., ENP).
- State – On or Off.

Output

```
NCP> SHOW KNOWN LINE STAT
```

```
Known Line volatile Status as of Mon Jul 2 12:00:00 1990

Line                                                      State
ENP                                                       On
```

Format 4: Displaying Line Summaries

```
SHOW KNOWN LINES SUMMARY
```

Description of the Display

- Line name (e.g., ENP).
- State – On or Off.

Output

```
NCP> SHOW KNOWN LINE SUMM
```

Known Line Volatile Summary as of Mon Jul 2 12:00:00 1990

```
l l.
Line                                                State
ENP                                                 On
```

10.22 SHOW KNOWN LINKS SUMMARY

Displays information on all links currently active on a local node.

Output Format

```
Known Link volatile Summary as of Fri Oct 12 09:35:10 1990

   Link   PID   Node      Remote    Remote User    Process
                          Link

   8313         1.232()   804
```

10.23 SHOW NODE

SHOW NODE is a collection of subcommands that fall into four categories, or formats. This command displays static node information for the executor, a specified node, all active nodes, or all known nodes.

Format 1: Displaying Node Characteristics

```
SHOW EXECUTOR CHARACTERISTICS
SHOW NODE node-id CHARACTERISTICS
SHOW ACTIVE NODES CHARACTERISTICS
SHOW KNOWN NODES CHARACTERISTICS
```

where:

node-id	identifies the local or remote node (name or address).

Description of the Screen Display for the Executor

- Identification – 4DDN version string.
- Management Version – Version number of the Network Management layer.
- NSP Version – Version number of the End Communication layer.
- Maximum Links – Maximum number of logical links for the node.
- Delay Factor – The value of the DELAY FACTOR parameter is multiplied by one sixteenth of the estimated round trip delay time to determine the value of retransmit time for certain NSP messages.

  When specifying values for the DELAY FACTOR parameter, use a value in the range 0 to 255. For example:

  ```
  NCP> SET EXEC DELAY FACTOR 10
  ```

- Delay Weight – The NSP layer estimates the current delay in the round trip transmission to a node with which it is communicating. The value of the DELAY WEIGHT parameter is used to calculate a new value of the estimated round trip delay. The value is in the 0-255 range. For example:

```
NCP> SET EXEC DELAY WEIGHT 15
```

- Inactivity timer – A logical link is inactive when no data is transmitted in either direction for a given interval of time. The *INACTIVITY TIMER* regulates the frequency with which 4DDN tests the viability of an inactive link. The *INACTIVITY TIMER* parameter specifies the maximum duration of inactivity before the local 4DDN node tests the viability of the link. For example, this command sets the inactivity interval to 30 seconds:

  ```
  NCP> SET EXECUTOR INACTIVITY TIMER 30
  ```

  When this timer expires, 4DDN generates artificial traffic to test the link.

- Retransmit factor – This specifies the number of times a packet may be retransmitted before a link is declared broken. The value of the RETRANSMIT FACTOR parameter regulates the number of times the Network Servers Protocol (NSP) Layer reattempts a transmission when its retransmission timer expires for a logical link. Use a number in the range of 0 to 65,535 for this value. For example, the following example specifies that NSP reattempts a transmission no more than 10 times:

  ```
  NCP> SET EXECUTOR RETRANSMIT FACTOR 10
  ```

  If NSP tries to retransmit the eleventh time, the logical link is disconnected.

- Routing Version – Version number of the Routing layer.

- Type – Type of the specified node. The values are:

  - ROUTING III
  - ROUTING IV
  - AREA
  - NONROUTING III
  - NONROUTING IV

- Maximum Address – Highest address that the local node will recognize.

- Max Broadcast Nonrouters – Maximum number of end nodes the executor node can have on its ethernet circuits. Valid values are in the 0–1023 range.

- Segment buffer size – Maximum size of message segment in bytes. This value is system dependent, but must be in the range of 255-1458 bytes.

Output Format

```
NCP> SHOW EXEC CHAR

Node Volatile Characteristics as of Mon Jul 2 12:00:00 1990

Executor Node = 1.4 (PLUTO)

Identification              = NML 1.0 SGI 4DDN Release 2.0
Management Version          = V4.0.0
NSP Version                 = V4.0.0
Maximum Links               = 32
Delay Factor                = 80
Delay Weight                = 5
Inactivity Timer            = 10
Retransmit Factor           = 10
Routing Version             = V2.0.0
Type                        = Nonrouting IV
Maximum Address             = 1023
Max Broadcast Nonrouters    = 1023
Segment Buffer Size         = 1458
```

```
NCP> SHOW NODE topcat CHAR

Node Volatile Characteristics as of Mon Jul 2 12:00:00 1990/

Remote Node = 1.2 (TOPCAT)

NO INFORMATION AVAILABLE
```

Format 2: Displaying Node Counters

```
SHOW EXECUTOR COUNTERS
SHOW NODE node-id COUNTERS
SHOW ACTIVE NODES COUNTERS
SHOW KNOWN NODES COUNTERS
```

where:

node-id	identifies the local or remote node (name or address).

Description of the Display

The node counters provide the following information about user traffic between the executor and the specified node.

- Seconds since last zeroed – Number of seconds that elapsed since the node counters were zeroed.
- User data bytes received – Number of user data bytes received by your node. This does not include protocol headers and checksums.
- User data bytes sent – Number of bytes of data sent by your node.
- User data messages received – Number of user data messages received by your node.
- User data messages sent – Number of user data messages sent by your node.
- Connects received – Number of logical link connection requests received by your node.
- Connects sent – Number of logical link connection requests sent by your node.
- Response timeouts – Number of times there was no response to an NSP segment within the allotted timeout period. Usually indicates overloading, messages being discarded, and necessary retransmission.
- Received connect resource errors – Number of incoming connection requests for which the local node did not have enough resources.

- Packet format error – Number of packet format error that occur because of invalid packet control information. This counter is kept for the executor node only.

Note: If no links have been established on the executor, *SHOW EXEC COUNT* displays only the "Packet format error" count.

Output Format

```
NCP> SHOW EXEC COUNT

Node Counters as of Mon Jul 2 12:00:00 1990

Executor Node = 1.4 (PLUTO)

12948   Seconds since last zeroed
19215   User data bytes received
63023   User data bytes sent
 1100   User data messages received
 1156   User data messages sent
    3   Connects received
  161   Connects sent
    0   Response timeouts
    0   Received connect resource errors
    0   Packet format error

NCP> SHOW NODE saturn COUNT

Node Volatile Counters as of Mon Jul 2 12:00:00 1990

Remote Node = 1.7 (SATURN)

8821   Seconds since last zeroed
1153   User data bytes received
1153   User data bytes sent
  60   User data messages received
  60   User data messages sent
  14   Connects received
  14   Connects sent
   0   Response timeouts
   0   Received connect resource errors
```

```
NCP> SHOW NODE 2 COUNT

Node Volatile Counters as of Mon Jul 2 12:00:00 1990

Remote Node = 1.2 (TOPCAT)

No information available

NCP> SHOW ACT NODE COUNT

Active Node volatile Counters as of Mon Jul 2 12:00:00 1990

Executor Node = 1.4 (PLUTO)

>65534    Seconds since last zeroed
 30225    User data bytes received
 30277    User data bytes sent
  1189    User data messages received
  1192    User data messages sent
     3    Connects received
     3    Connects sent
     0    Response timeouts
     0    Received connect resource errors
     0    Packet format error

Remote Node = 1.5 (ROMULA)

>65534    Seconds since last zeroed
131773    User data bytes received
 37128    User data bytes sent
  1291    User data messages received
  1275    User data messages sent
     0    Connects received
    20    Connects sent
     0    Response timeouts
     0    Received connect resource errors

Remote Node = 1.7 (SATURN)

>65534    Seconds since last zeroed
 47270    User data bytes received
 18196    User data bytes sent
```

```
  1849   User data messages received
  1851   User data messages sent
     2   Connects received
    12   Connects sent
    10   Response timeouts
     0   Received connect resource errors


NCP> SHOW KNOWN NODE COUNT

Known Node Volatile Counters as of Mon Jul 2 12:00:00 1990

Executor Node = 1.4 (PLUTO)

>62545   Seconds since last zeroed
  2000   User data bytes received
  8000   User data bytes sent
    30   User data messages received
    20   User data messages sent
     2   Connects received
     0   Connects sent
    20   Response timeouts
     0   Received connect resource errors

Remote Node = 1.2 (TOPCAT)

No information available
```

Format 3: Displaying Node Status

```
SHOW EXECUTOR STATUS
SHOW NODE node-id STATUS
SHOW ACTIVE NODES STATUS
SHOW KNOWN NODES STATUS
```

where:

node-id	identifies the local or remote node (name or address).

Description of the Display for the Executor Node

- Node address and name.
- State – On or Off.
- Ethernet Physical Address.

Description of the Display for a Remote Node

- Node address and name.
- Routing state – reachable or unreachable.
- Number of active logical links associated with the node.
- Delay time to set the retransmission.
- Node type – ROUTING IV, NONROUTING IV, or AREA.

Output Format

```
NCP> SHOW EXEC STAT
```

```
Node Volatile Status as of Mon Jul 2 12:00:00 1990

Executor Node = 1.4 (PLUTO)

State            = On
Physical Address = aa-00-04-00-04-04
```

```
NCP> SHOW NODE 1 STAT
```

If node 1 is the executor, see the *SHOW EXEC STAT* command.

```
NCP> SHOW NODE 2 STAT

Node Volatile Status as of Mon Jul 2 12:00:00 1990

Node           State      Active  Delay  Type        Cost  Hops  Circuit
                          Links
1.2 (TOPCAT)   reachable                 routing IV              ENP

NCP> SHOW ACTIVE NODE STAT

Node Volatile Status as of Mon Jul 2 12:00:00 1990

Executor Node = 1.4 (PLUTO)

State              = On
Physical Address   = aa-00-04-00-04-04

Node           State      Active  Delay  Type        Cost  Hops  Circuit
                          Links
1.2 (TOPCAT)   reachable                 routing IV              ENP

NCP> SHOW KNOWN NODE STAT

Known Node Volatile Status as of Mon Jul 2 12:00:00 1990

Executor Node = 1.4 (PLUTO)

State              = On
Physical Address   = aa-00-04-00-04-04

Node           State      Active  Delay  Type        Cost  Hops  Circuit
                          Links
1.2 (TOPCAT)   reachable                 routing IV              ENP
```

Format 4: Displaying a Node Summary

```
SHOW EXECUTOR SUMMARY
SHOW NODE node-id SUMMARY
SHOW ACTIVE NODES SUMMARY
SHOW KNOWN NODES SUMMARY
```

where:

node-id	identifies the local or remote node (name or address).

Description of the Display

- Node address and name.
- Routing state – reachable or unreachable.
- Number of active logical links associated with the node.
- Retransmission delay time in seconds.

Output Format

```
NCP> SHOW EXEC SUMM
```

```
Node Volatile Summary as of Mon Jul 2 12:00:00 1990

Executor Node = 1.4 (PLUTO)

State           = On
Identification  = NML 1.0 SGI 4DDN Release 2.0
```

If node is the executor, see the *SHOW EXEC SUMM* command.

```
NCP> SHOW NODE 2 SUMM
```

```
Node Volatile Summary as of Mon Jul 2 12:00:00 1990

Node          State      Active   Delay   Circuit   Next Node
                         Links
1.2 (TOPCAT)  reachable                   ENP
```

```
NCP> SHOW ACT NODE SUMM

Node Volatile Summary as of Mon Jul 2 12:00:00 1990

Executor Node = 1.4 (PLUTO)

State          = On
Active Links   = 4

Node           State        Active   Delay   Circuit   Next Node
                            Links
1.2 (TOPCAT)   reachable                     ENP


NCP> SHOW KNOWN NODE SUMM

Known Node Volatile Summary as of Mon Jul 2 12:00:00 1990

Executor Node =   1.4 (PLUTO)

State           = on
Identification  = NML 1.0 SGI 4DDN Release 2.0

Node            State         Active   Delay   Circuit   Next node
                              Links
1.2 (TOPCAT)    reachable                      ENP
1.5 (ROMULA)    reachable     1        4       ENP
1.6 (VULCAN)    unreachable
1.7 (SATURN)    unreachable
```

10.24 TELL

Use the TELL prefix to identify the executor node for a particular NCP command. TELL sets the executor for only one command and must prefix the command for which it is intended. For example:

```
NCP> TELL pluto SHOW EXECUTOR CHAR
```

This command sets the executor to node *pluto* where the SHOW EXECUTOR CHAR executes.

10.25 ZERO EXECUTOR COUNTERS

This command resets all counters to zero on the executor node.

10.26 ZERO KNOWN CIRCUIT COUNTERS

This command resets circuit counters to zero for all known circuits.

10.27 ZERO KNOWN LINE COUNTERS

This command resets line counters to zero for all known lines.

10.28 ZERO KNOWN NODES COUNTERS

This command resets node counters to zero for all known nodes.

10.29 ZERO NODE COUNTERS

This command resets node counters to zero for a specified node, remote or local.

```
ZERO NODE node-id COUNTERS
```

where:

node-id	identifies the local or remote node (name or address)

Appendix A

Troubleshooting Hints

The most common errors that administrators encounter with 4DDN usually occur during installation and generate messages from the Network Control Program. Occasionally, errors also occur during 4DDN operation because of improper password specifications on an IRIS workstation.

This appendix explains how to troubleshoot and correct installation and password access problems. In addition to errors described in this appendix, check the IRIX file */usr/adm/SYSLOG* for errors that might occur when *dnserver* invokes other 4DDN servers.

Symptom:	As *dninstall* executes, NCP posts this error:

```
NCP> %NCP-I-NMLRSP - Invalid parameter value
```

Probable Cause:	A line in your */usr/etc/dn/nodes* file contains a syntax error.
Corrective Action:	Locate and correct the syntax error in *nodes*:

1. Identify the number of the incorrect line.

   As NCP processes each line of the *nodes* file, it generates a prompt on your screen, `NCP>`. The prompt is followed by a blank line if the corresponding line in *nodes* processed successfully. If a line in *nodes* contains an error, the `NCP>` prompt is followed by an error message. Note the number of the line containing the `NCP>` prompt with the error message.

2. Find the corresponding line in your *nodes* file.

   Find the line in your *nodes* file with the same number as the screen line containing the error message.

3. Check your entry for syntax errors.

   Identify any errors in the *nodes* line that caused the processing error. For information on the syntax of *nodes* entries, see Step 3 of the configuration procedure in Chapter 3.

Symptom:	As *dninstall* executes, NCP posts this error:

```
NCP> %NCP-I-NMLRSP - Component in wrong state
```

Probable Cause:	The executor node (the node where *dninstall* is executing) is set to *on*. The executor node state must be set to *off* to run the installation script.
Corrective Action:	Set the executor state to off, using one of these commands:

```
# /usr/etc/dn/ncp set executor state off
NCP> set executor state off
```

Symptom:	As *dninstall* executes, NCP posts this error:

```
NCP> Fatal error: /dev/net_man ioctl (c0424e32) failed:
                                        Invalid ioctl function
```

Probable Cause:	You did not rebuild or reboot the new 4DDN kernel after software installation.
Corrective Action:	Run *autoconfig(1)* and reboot your workstation. Then, run *dninstall* again.

Symptom:	Outgoing requests from an IRIS to VMS nodes succeed, but requests coming into the IRIS from VMS nodes fail. VMS posts this error: `%SYSTEM-F-INVLOGIN`
Probable Cause:	The */etc/passwd* file entry for the default account for DECnet connections is not in uppercase characters.
Corrective Action:	Reset the password on the default account using commands like those shown below. This example assumes the name of the default account is *decnet* and the password to it is *DECNET*.

```
# /bin/passwd decnet
New password: DECNET
Re-enter new password: DECNET
```

Appendix B

Default Parameter Values

B.1 Node Parameters

Delay Factor	80
Delay Weight	5
Inactivity Timer	10
Retransmit Factor	10
Maximum Address	1023
Maximum Broadcast Nonrouters	1023
Segment Buffer Size	1460
Address	1
Type	Nonrouting IV

B.2 Loop Parameters

Count	1
Length	40
With	Mixed

Appendix C

NCP Error Messages

NCP generates error messages for a number of system-level and network-level error conditions. Messages include a brief description of the error and a flag specifying the parameter that produced the error. The invalid parameter is enclosed by slashes (/). The example below illustrates the format of NCP error messages.

```
NCP> SHOW FOO
NCP: Syntax Error
SHOW /FOO/
```

The following is a list of error messages NCP generates and an explanation of their meanings.

Unimplemented Function or Option

The NCP command function or option is not supported by the implementation.

System-specific management function not supported

The NCP command is not supported by the implementation.

Unimplemented Parameter Type

The NCP command parameter is not supported by the implementation.

Incompatible Management Version

The NCP process tried to connect to an earlier version of Network Management Listener (lower than 4.0.0).

Unrecognized Component

There is no database entry for the indicated entity.

Component in Wrong State

This message may be generated when a user tries to change the node address while the operational state of the node is set to ON.

Rejected Access Control, Privilege Violation

This message may be generated when a user attempts to perform an operation to which he or she has insufficient privilege.

Object Not Found

The Network Management Listener (NML) was not found on the remote node.

Invalid Parameter Value

The parameter value sent to NML was not acceptable.

Unrecognized Node Name

The node name is missing from the NCP database.

Privilege Violation

The user did not have superuser privilege to modify the NCP database.

Link Access Failure

The link to the remote node cannot be established.

Management Program Error

4DDN software error.

Appendix D

Man Pages

The pages that follow are the IRIX manual pages that pertain to 4DDN network management tasks.

Index

We'd Like to Hear From You

As a user of Silicon Graphics documentation, your comments are important to us. They help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics to comment on:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Please include the title and part number of the document you are commenting on. The part number for this document is 007-0821-030.

Thank you!

Three Ways to Reach Us

The **postcard** opposite this page has space for your comments. Write your comments on the postage-paid card for your country, then detach and mail it. If your country is not listed, either use the international card and apply the necessary postage or use electronic mail or FAX for your reply.

If **electronic mail** is available to you, write your comments in an e-mail message and mail it to either of these addresses:

- If you are on the Internet, use this address: techpubs@sgi.com
- For UUCP mail, use this address through any backbone site: *[your_site]*!sgi!techpubs

You can forward your comments (or annotated copies of manual pages) to Technical Publications at this **FAX** number:

415 965-0964