4DDN User’s Guide and Man Pages

Document Number 007-0820-030

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 for 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. 4DDN nodes on remote networks are reachable by means of a DECnet router.

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 on your IRIS simultaneously. If TCP/IP is also running on your workstation, you can use IRIX™ network services (such as *rlogin* and *rcp*), in addition to 4DDN services, to connect to other IRIS nodes on the network.

1.1 4DDN User Services

4DDN user services are provided by a set of commands that you enter at your workstation to connect to other DECnet nodes (computers) or transfer data over the network. 4DDN commands provide these services:

sethost	Logs you in to a remote node for a virtual terminal session. During the session, the commands you enter are executed by the remote node, rather than by your workstation.
dnls	Lists the contents of a directory on a remote node on the network.
dnmv	Moves or renames a file on a remote node in the network.
dnrm	Removes a file in a remote directory on the network.
dncp	Transfers files to, from, or between remote nodes on the network.
dnex	Executes a command file on remote nodes on the network.
dnlp	Prints a file to a network printer.
dnMail	Provides mail service between IRIS workstations running 4DDN and VMS™ nodes on the network.

1.2 Using This Guide

The *4DDN User's Guide* documents 4DDN user services, available by means of command entries, to establish connections and transfer data over a DECnet network. It describes how to use the virtual terminal service; how to access, execute, and print files; and how to send mail over the network from an IRIS Series workstation.

The *4DDN User's Guide* is one of a three-volume set that documents the 4DDN product. Other members of the set are the *4DDN Programming Guide*, which explains how to write network applications for 4DDN; and the *4DDN Network Management Guide*, which explains how to configure 4DDN software and how to monitor and control the 4DDN network.

This guide assumes you are acquainted with the user-level services available over the DECnet and have experience using the VMS operating system. If you are not familiar with network concepts and terminology, and if you are new to IRIX, see Chapter 2, "Notes to New Users," for introductory information.

1.3 Guide Contents

The *4DDN User's Guide* is written to help you use 4DDN commands to do tasks over a DECnet network. It describes each command used to invoke a network service, explains its syntax and options, and gives sample command entries. It also explains the probable cause of error messages that might be displayed as you use 4DDN and what you can do to correct errors.

The *4DDN User's Guide* is organized into these chapters:

Chapter 1	This introduction explains the purpose of the *4DDN User's Guide* and introduces the 4DDN product.
Chapter 2	"Notes to New Users" orients new users to networking concepts, explains terms used throughout this guide, and gives tips on using the IRIS interface for entering 4DDN commands. This chapter also provides a table of command equivalents for VMS, IRIX, and 4DDN.

Chapter 3	"Preparing to Use 4DDN" contains procedures to customize your 4DDN work environment, explains the structure of 4DDN commands, and describes how to specify files in 4DDN command entries.
Chapter 4	"Using the Virtual Terminal Service" describes how to connect to a remote network node and use it as if you were a local user.
Chapter 5	"Managing Remote Files" describes how to list, rename, and remove files on remote nodes.
Chapter 6	"Copying Files over the Network" describes how to copy files between local and remote nodes and between two remote nodes in the network.
Chapter 7	"Executing Remote Files" describes how to execute files that reside on remote nodes.
Chapter 8	"Printing over the Network" describes how to print local and remote files on printers connected to either your local node or a remote node in the network.
Chapter 9	"Using 4DDN Mail Service" describes how to send mail messages to other network nodes.

1.4 Additional Command Information

All commands for 4DDN user services are described in detail in this guide. In addition to user commands, you might want to use a group of monitoring commands to check network activity. Network monitoring commands are described in the *4DDN Network Management Guide.*

1.5 Style Conventions

This guide uses standard UNIX® conventions when referring to entries in 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 *IRIS-4D User's Reference Manual, Volume 1*.

In body text, commands and file and directory specifications entered on IRIX systems appear in *italics*, while commands entered on VMS systems appear in *UPPER-CASE ITALICS*. In addition, we use these typographical conventions throughout this guide:

`typewriter font`	Command syntax descriptions, examples, and screen displays appear in typewriter font.
`bold typewriter`	User entries are shown in bold typewriter font.
`UPPER CASE`	In examples and instructions, VMS entries are shown in upper-case typewritier font. Press **`<return>`** after typing these entries.
bold	Command options are shown in bold when they appear in body text. Options do not appear in bold in syntax descriptions and examples.

1.6 Related Documentation

For additional information on using the 4DDN, see Silicon Graphics publications:

4DDN Programming Guide

4DDN Network Management Guide

Also see these Digital Equipment Corporation guides:

Guide to Networking on VAX/VMS

Guide to VAX/VMS File Applications

1.7 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 new users with some of the concepts and terms that apply to networking in general and others that are specific to the DECnet network. It also gives tips for using the IRIX user interface to enter 4DDN commands and a table of equivalent commands for VMS, IRIX, and 4DDN.

2.1 If You Are New to Networking

A network is a way of linking computers so users can share their resources and transfer data among the computers. Specialized hardware and software connect the computers and support network services. The resources offered in a network can include files, programs, processors, printers, and other devices. Individual networks vary in size and overall shape, or *topology*, and networks can be grouped together in an arrangement called an *internet*.

The computers in a network are referred to as *nodes*, *hosts*, or *stations*, depending on the network application. Computers in the DECnet network are referred to as *nodes*. The node at which you enter commands is known as your *local node*. All other nodes in the network are known as *remote nodes*. Each node in the network is assigned a name and address to identify it to users, as well as to network hardware and software.

When you enter a network command, the software that handles your request is referred to as the *client*. The software on the remote node that responds to the request is referred to as the *server*. In some network applications, the hosts involved in the transaction are also referred to as *client* and *server*.

2.1.1 Network Interoperability

To meet user needs, networks must provide *interoperability*; that is, they must support communications among computers manufactured by different vendors, running a variety of software. To ensure interoperability, network hardware and software are designed according to widely accepted industry standards called *protocols*. Digital Network Architecture, DNA, developed by Digital Equipment Corporation, and Transmission Control Protocol/Internet Protocol (TCP/IP), developed by the Department of Defense, are architectural models that specify how protocols are implemented
in a network.

Many networks in widespread use, such as DECnet and TCP/IP, are based on an architecture of *layered protocols*. In this architecture, each layer of the network performs a specific function and provides a particular service to the layer above it. Because of standardization, a set of layers supplied by one vendor can provide services to the layers above it, even if the upper layers are supplied by a different vendor. This modularity of a layered protocols makes interoperability possible.

Figure 2-1 illustrates the layers in a DNA network.

Figure 2-1 DNA Network Layers

2.1.2 Ethernet Networks

The DECnet network has seven layers, as specified in the DNA model. Frequently, the first two layers, network hardware and its link-level software, are an implementation of the Ethernet specification. Ethernet networks are local area networks (LANs) optimized for high-speed data transfer. Nodes in an Ethernet network are connected by coaxial cable, and data is routed between nodes in the form of *packets*.

2.2 If You Are New to the DECnet

DECnet networks contain two types of nodes: end nodes and router nodes. An *end node* can send packets to other DECnet nodes and receive packets that bear its address. A *router node* can receive and forward packets addressed to other nodes. An IRIS workstation running 4DDN operates as an end node on the DECnet.

Nodes attached to the same Ethernet line are considered *adjacent nodes*. All nodes on the network are identified by a *node name* and a *node address*.

When you connect to a remote node in the network, the logical path between your local node and the remote one is called a *circuit*. Circuits operate over physical lines in the network and can support simultaneous connections among multiple network nodes.

When a message is sent from one network node to another, the starting point in the transfer is known as its *source*, and the arrival point is known as its *destination,* or *target* (these terms are used interchangeably). The terms *source*, *destination*, and *target* also describe data and locations on the network. For example, a *source file* transferred to another system in the network becomes the *target file* in the *destination directory.*

2.3 If You Are New to IRIX

IRIX is the operating system supplied with your IRIS workstation. 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 *IRIS 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 *IRIS Owner's Guide* for more introductory information on IRIX.)

2.3.1 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.

2.3.2 Using Command Equivalents

Table 2-1 below 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 lowercase or uppercase, as shown in this table.

VMS	IRIX	4DDN
COPY	cp or rcp	dncp
DELETE	rm	dnrm
DIRECTORY	ls	dnls
MAIL	mail, Mail	dnMail
RENAME	mv	dnmv
PRINT	lp	dnlp
SET HOST	rlogin	sethost
SUBMIT	sh	dnex
TYPE	more	(none)

Table 2-1 4DDN Command Equivalents

Chapter 3

Preparing to Use 4DDN

This chapter describes some things you can do to prepare for using 4DDN software on your workstation. It explains how to customize your own 4DDN environment and gives general instructions for entering 4DDN commands. The command information provided in this chapter pertains to these 4DDN commands:

dnls	lists individual remote files or the contents of remote directories
dnmv	renames files on a remote node
dnrm	removes files from a remote node
dncp	transfers files to, from, or between remote nodes
dnex	executes files on a remote node
dnlp	prints files to a network printer

If 4DDN software is not already installed and configured on your IRIS, you (or your system administrator) must complete these procedures before you proceed. Installation and configuration information is given in the *4DDN Network Management Guide*.

3.1 Adding 4DDN Software to Your Path

The commands that support 4DDN user services are located in the directories */usr/bin/dn* and */usr/etc/dn* on your IRIS. To execute 4DDN commands from any directory in your user account, you must set *environment variables*. IRIX environment variables direct the command interpreter to use default values for certain information as it interprets your commands.

If you are using the C Shell as your command interpreter, execute these commands to set the PATH for 4DDN directories:

```
# setenv PATH /usr/etc/dn
# setenv PATH /usr/bin/dn
```

If you are using the Bourne Shell as your command interpreter, execute these commands:

```
$ PATH=/usr/etc/dn
$ PATH=/usr/bin/dn
```

If you would like these variables to be set automatically when you log into your workstation, include the 4DDN directories in your login file. If you use the C Shell, add your entries to the PATH variable in your *.login* file.

```
setenv PATH /usr/etc/dn /usr/bin/dn
```

If you use the Bourne Shell, add your entries to the PATH variable in your *.profile* file.

```
PATH=/usr/etc/dn:/usr/bin/dn
```

3.2 Setting Access Control Information

When you enter a 4DDN command, *access control information* is sent to the destination node so it can determine your privileges for reading, writing, or executing the files you access. Access control information comprises three elements:

a *user name*	that identifies the user on the remote system in whose name the access is performed
a *password*	that protects the user account from unauthorized access
an *account ID*	that indicates the party to be billed for network access time (not supported on 4DDN nodes).

You can provide access control information in several ways, one of which is to include it as part of your 4DDN command entry (described in Section 3.3, "4DDN Command Elements). Although entering access control information in your commands is satisfactory in many cases, it has two disadvantages: it requires extra time and effort; and, it presents security risks, since other users can view your passwords by issuing the *ps*(1) command. For these reasons, 4DDN provides ways to set up default access control information so it is supplied automatically whenever you omit it from 4DDN commands.

You can supply default access control information in outgoing 4DDN requests in two ways: by creating the *.4ddnrc* file, a file that contains default access control information; or by setting environment variables for your login session. Procedures for using each method are described in the sections that follow. In considering which method to use, keep these facts in mind:

- The *.4ddnrc* file supplies both a default user name and a default password in outgoing requests to the remote accounts you specify. When you want to access these accounts, no access information is required in your entries.

- Environment variables can supply a user name and user identification number in outgoing requests, but they cannot supply passwords. To complete your connection to a remote account, you will be prompted to enter a password manually (this entry is not viewable by other users).

- Access control information you enter on the command line always takes precedence over default access control information; and access control information set by environment variables takes precedence over the information in the *.4ddnrc* file.

3.2.1 Using the *.4ddnrc* File

If you create a *.4ddnrc* file, the default access control information it contains is automatically supplied to your 4DDN connection request if you omit access information from the command line. *.4ddnrc* will supply both an account name and password, or the password only, to your outgoing requests. (A remote node name and remote file specification are always required in 4DDN commands).

The *.4ddnrc* file must reside in your home directory, and it must have read-write permission by the user only. It can contain any number of entries for accounts you are authorized to use, and the accounts can be located on any remote nodes accessible to 4DDN commands.

The Format of *.4ddnrc*

Each entry in *.4ddnrc* specifies a remote node where you have an account, the remote account name, and the password to the account. 4DDN software includes a preliminary version of the *.4ddnrc* file. This version is located in a file called */usr/etc/dn/4ddnrc* and looks like the sample in Figure 3-1.

```
default:slug:pam:techky
account:slug:jim:orange
account:vms_node:system:secure
default:vms_node:john:banana
account:barrow:rich:flex
default:barrow:bent:scuba
```

Figure 3-1 Preliminary *4ddnrc* File

Notice the two types of entries in *.4ddnrc*, *default* and *account*. A *default* entry specifies the remote account you want to reach when your 4DDN command contains no access control information. An *account* entry specifies the remote ne 3v account you want to reach when you enter a user name (**-u** option) in your enter but not a password (see Section 3.4.3, "Specifying Access Information").

3.2.2 Editing *.4ddnrc* (optional)

The procedure below explains how to copy *4ddnrc* to *.4ddnrc* in your home directory and replace the dummy values with valid access control information.

Note: Access control information that you enter on the command line overrides your entries in *.4ddnrc*.

1. Change to your home directory.

```
# cd ~
```

2. Copy the preliminary file to your home directory and rename it.

```
# cp /usr/etc/dn/4ddnrc .4ddnrc
```

3. Edit *default* entries (see Figure 3-1 for an illustration):

   - Do not change the first field, *default*.

     default is a keyword identifying the entry type.

   - Enter a valid remote node name.

     Replace the dummy node name (*slug*) with the name of the node where the remote default account is located.

   - Enter a valid user name.

     Replace the dummy user name *pam* with the name of the remote account you want to reach on this host when you omit access control information in 4DDN commands.

   - Enter a valid password.

     Replace the dummy password (*techky*) with the password to this default account.

4. Edit the *account* entry.

   - Do not change the first field, *account*.

     account is a keyword identifying the entry type.

   - Enter a valid remote node name.

     Replace the dummy node name (*slug*) with the name of the node where the remote account is located.

   - Enter a valid user name.

     Replace the dummy user name *jim* with the name of the remote account you want to reach when you enter the user name (**-u** option) in 4DDN commands to this host (see Section 3.4.3, "Specifying Access Information").

   - Enter a valid password.

     Replace the dummy password (*4bye*) with the password to this account.

5. Repeat Steps 3 and 4, creating entries for remote accounts on other nodes you wish to access.

3.2.3 Setting Environment Variables

You can set two environment variables to supply default information in your outgoing 4DDN requests:

NET_USER	supplies the user name on your home account
NET_ACCOUNT	supplies your account identification number (UID)

The commands shown above set the *NET_USER* and *NET_ACCOUNT* environment variables so they remain in effect during your login session. If you would like these variables to be set automatically when you log in to your workstation, enter the C Shell commands in your *.login* file and the Bourne Shell commands in your *.profile* file.

```
setenv NET_USER LEE           (C Shell format)
NET_USER LEE ; export NET_USER      (Bourne Shell format)
```

Note: Access control information you enter on the command line overrides your environment variable settings.

3.3 4DDN Command Elements

A 4DDN entry consists of a 4DDN command, command options, and file specifications for the local and/or remote files. 4DDN command lines follow IRIX syntax conventions:

```
command [options...] local_file_spec <==> remote_file_spec
```

Notice that the order of the local and remote file specifications is reversible. The order you use depends upon the action you wish to take on the file.

On 4DDN nodes, command entries and options are lower case. Both upper- and lower-case entries are valid on VMS systems. You can usually entry command options in any order (see individual command descriptions for instructions on ordering options).

4DDN commands and most 4DDN command options are entered in lower case. However, where options appear in upper case in this guide, use upper-case letters in your command entries. Arguments to commands, such as user names and passwords, are handled by the destination node. When arguments are passed to VMS systems, case is irrelevant; however, arguments passed to 4DDN or ULTRIX™ nodes must observe case conventions.

Local file specifications follow the conventions of the node where you enter the command line. Remote file specifications provide information that 4DDN software needs to access the file across the network.

3.4 Entering Remote File Specifications

A fully qualified remote file specification consists of a node identifier (name or address), access control information (optional), and a file specification that can include a device name. The node name and access control information (if it is included) are always followed by a double colon (::). Access control information is enclosed in double quotes (" ").

The syntax of remote file specifications is shown below:

```
node"access-information"::device:file-spec
node::device:file-spec
node::file-spec
```

The sample entry below illustrates a remote file specification in a 4DDN command entry. Labels below the entry identify each field in the specification.

```
dncp  'ginsu"rob boston"::/usr/rob/myfile1'  myfile1

        node     access         remote        local
        ID      information    file-spec     destination
```

Notice that the remote file specification is enclosed in quotation marks. The following sections explain the purpose of the quotation marks and describe each item in a remote file specification.

3.4.1 Using Quotes in Your Specifications

The IRIX and VMS operating systems each reserve a set of special characters, such as punctuation marks, that are processed in a particular way when they appear in command line entries. When special characters occur in remote file specifications, they can be misinterpreted by the local VMS or IRIX operating system and cause your 4DDN entries to fail.

You can avoid problems with special characters by using quotation marks to enclose remote file specifications:

- Use single quotes (’ ’) to enter a VMS file specifications from a 4DDN node.
- Use double quotes (" ") to enter a 4DDN file specification from a VMS node.

For example, although the entry below includes double quotes and square brackets, this file specification is acceptable when issued from a 4DDN node because it is enclosed in single quotes:

```
  ’vms-node"username password"::dev1:[dir.subdir]file-spec’
```

This file specification with forward slashes is acceptable when issued from a VMS node because it is enclosed in double quotes.

```
irix-node"username password"::"/usr/eng/tom/data1"
```

3.4.2 Specifying a Node

The node identifier in a remote file specification must be a DECnet node name or DECnet node address. A node name can be up to six characters long. A node address must conform to the area-number.node-number format.

3.4.3 Specifying Access Information

Access control information sets user privileges for reading, writing, or executing files on a remote system. It is not mandatory for all 4DDN entries, but might be required on some remote accounts you use. You can also set up your 4DDN software to automatically include access information when you omit it from your entries (see Section 3.2). The access information you specify on a command line takes precedence over automatically entered default information.

The syntax for access control information is shown below. Notice that not all fields are required.

```
"user password account-id"     (account ID is optional)
"user password"
"user"                         (user is prompted for the password)
```

Enclose access control information in double quotes and separate each access control word by a space. If you specify a user name but omit the password in the access control information, you will be prompted to enter the password before your connection to the remote host is completed.

Note: Passwords entered on the command line are visible to the IRIX *ps(1)* command. Do not specify a password if you are concerned about security.

You can supply access control information on a 4DDN command line in two ways:

1. Enclose it in double quotes after the node name.

   ```
   'node"access-info"::file-specification'
   ```

   When you include access control information in a remote file specification, enter a double colon (::) after the quoted information.

2. Specify it as arguments to command options.

   `-u` *user-name* `-p` *password* `-a` *account-id* `'node::file-specification'`

   where:

   –**u** User name: The argument following –**u** is the user name used to access all remote files on the command line.

–p Password: The argument following –**p** is used as the password for the specified user name. If you omit –**p** from the access control information, you are prompted to enter a password before your connection is completed, unless you are using a *.4ddnrc* file (see Section 3.2).

–a Account ID: The argument following –a is the account string used to access all the files specified on the command line. This argument is not valid when the remote host is a 4DDN node.

3.4.4 Specifying Remote File Names

Specify a remote file name as if you were using the remote system locally. The command below illustrates how to enter remote file specifications from a VAX/VMS node when the destination node is an IRIX node, *mutant*.

```
# COPY [USERS.JOHN]FIX1.LOG MUTANT::"/usr/mail/jim/fix1.log"
```

This command illustrates how to enter remote file specifications from an IRIX node when the destination node is a VAX/VMS node, *gorgo*:

```
# dncp /usr/ben/fix2.log ’gorgo::sys$device:[users.joe]fix2.log’
```

Periods in File Specifications

On VMS nodes, periods (.) in a directory specification are equivalent to the forward slash (/) on 4DDN nodes; that is, they separate levels of the VMS directory structure. For this reason, only one period can be used in a file name that is processed by a VMS node. So, although the entries */u0/baa/rfas.doc/doc.n* and */usr/bin/a.b.c* are legitimate on UNIX systems, they are illegal in the VMS environment.

Case Conventions in File Specifications

When working on a VAX™/VMS node, keep in mind that VMS makes no case distinctions. This means that although the files *Myfile* and *myfile* are two distinct files in the UNIX environment, they cannot exist as separate files on a VMS node.

3.5 Unsupported Commands

When a 4DDN command arrives at a remote network node, it must connect with a reciprocating service at the destination node. Not all 4DDN commands are supported at a given network node, however. For example, most remote nodes provide service to incoming *dncp* and *dnls* commands, but some nodes do not service incoming *dnmv* or *dnrm* commands.

When you request a 4DDN service that is not supported on a remote node, you are likely to receive an error message about an *unsupported operation*. To verify that a particular service is not available, check with the system administrator of the remote node where the problem occurred.

Chapter 4

Using the Virtual Terminal Service

The *sethost* command provides virtual terminal service to DECnet nodes. It allows you to connect to a remote node, log in, and work on the remote node as if you were a local user. The work you do over this kind of connection is referred to as a *virtual terminal session*.

Using *sethost* from your IRIS workstation, you can log in to a remote VMS or ULTRIX node running DECnet. You can also use *sethost* from a VMS or ULTRIX node to log in to a remote IRIS running 4DDN.

4.1 Establishing a Connection

Before you log in to a remote system, you must first establish a connection between your node and a process at the remote node called the *virtual terminal server*. The connection allows communication between the virtual terminal program in your local node and a virtual terminal server in the remote node.

When the connection is established, the remote system prompts you for the necessary login information. Once you are logged in to a remote node, the virtual terminal session has begun, and you can issue any commands available to local users. Your commands are executed by the operating system at the remote node, as if you were using a terminal.

For example, you can log in to a remote node and edit a file residing there, direct a file to be printed on the remote system's printer, or compile a program using the processing resources of the remote system.

4.2 Using the *sethost* Command

The *sethost* command connects you to a remote node so that you can log in and use the remote node as if it were your local node. *sethost* resides in the directory */usr/bin/dn*. To establish a connection with the *sethost* command, the remote host you specify must be running a virtual terminal server.

The syntax of a *sethost* command is shown below.

```
sethost -r node-name
sethost -r node-address
```

4.2.1 Specifying Remote Nodes

You must use a valid node name or node address when you use the *sethost* command. If you are not sure of the names and addresses of nodes in your DECnet network, check with your network manager, or use the Network Control Program commands *SHOW KNOWN NODES SUMMARY* or *SHOW ACTIVE NODES SUMMARY* to display the names of nodes you can reach.

4.2.2 Using *sethost* Options

The options and arguments to the *sethost* command are defined as follows:

–r	displays the version level and release date of *sethost* and the 4DDN product.
node-name	the name of a remote node. A node name is a group of not more than six alphanumeric characters with at least one letter. The node name should be unique across the network.
node-address	the address of an active remote node. It must conform to the *area-number.node-number* format. If the remote node is in the same network area as the local node, you can simply use the node number.

The area number identifies a group of nodes in the network. The area number must be an integer in the 1–63 range.

The node number must be an integer in the 1–1023 range. Node numbers must be unique across your network area.

If neither *node-name* nor *node-address* is specified, you are prompted to enter the information.

Next, you are prompted to supply the login information required by the remote node.

4.3 Ending a Virtual Terminal Session

You can normally exit a virtual terminal session by returning to the remote operating system prompt and logging out of the remote system in with the standard logout procedure. If for some reason a routine logout is not possible, you can abort the virtual terminal session in the middle of a program.

To log out of a remote node, issue the logout command for that remote node. For example, type **`LOGOUT`** at a VAX/VMS node.

To abort the virtual terminal session from your terminal, type the `<CTRL>-Y>` key combination twice. The following prompt is displayed:

```
Do you wish to abort the network virtual terminal session ?
```

Type **`Y`** (yes) to terminate the currently executing program and disconnects you from the remote host. You return to the command level in your local system.

Type **`N`** (no) to return you to the local command level in the remote system.

Chapter 5

Managing Remote Files

4DDN has three commands that help you manage files over the network: *dnls* lets you list individual files and the contents of directories on remote nodes; *dnmv* lets you move files on remote nodes and change their names; and *dnrm* lets you delete (remove) files stored on remote nodes. When you use these commands, your file or directory specification can be a symbolic link (a special file that references another file or directory); 4DDN software follows the link to the designated location.

5.1 Listing Files with *dnls*

The *dnls* command lists the files in each remote directory name you specify in your command entry. If you enter a file specification rather than a directory specification, *dnls* lists all files with names that match your entry.

The output of a *dnls* command is a line identifying the directory where the files were found, followed by a list of files contained in the directory. The file list is alphabetical, unless you specify otherwise. If you specify more than one directory, directory listings are separated by a blank line.

The syntax of a *dnls* command is shown below.

```
dnls -options directory ...
dnls -options filename ...
```

You can get on-line help with *dnls* syntax by entering the command with no arguments:

```
dnls <return>
```

5.1.1 Specifying Files in *dnls* Entries

Follow the instructions in Section 3.4, "Entering Remote File Specifications," to enter file specifications in *dnls* commands. You can use wildcard characters in your specification, as long as you observe the wildcard rules in effect on the remote node containing the files.

5.1.2 Using *dnls* Options

The *dnls* command offers a number of options that modify the way it works. A *dnls* entry can include one or more of the options described in the list below, as well as the **-u** (user name), **-p** (password), and **-a** (account ID) options described in Section 3.4.3, "Specifying Access Information." You can enter options in any order; however, the **-u**, **-p**, and **-a** options affect only file specifications appearing to their right. By default, *dnls* displays files in column format.

Use the options listed below in *dnls* command entries. For multiple options, you can use a single minus sign, followed by a group of options.

-h Suppresses headers that identify file directories and prints only the file list.

-l Lists files in long format. The protection mode, user ID code (owner), size in bytes, and time of last modification are listed along with the name of the file. File size is the number of bytes required to store the file in its native format. File size can differ if the record format of the file is converted (from VMS to UNIX, for example).

When the long output format is requested, the twelve-character protection mode is printed showing four protection levels:

1. System, for the system user (this value is meaningless on 4DDN nodes, since system protection is a VMS feature that has no equivalent on an IRIX system)
2. Owner, for the owner of the file
3. Group, for users in the owner's group (or the file's group on IRIX)
4. World, for all other users

Each three-character set specifies user privileges for reading, writing, and executing a file (or listing, for a directory). The privileges are as follows:

`r` if the file can be read

`w` if the file can be written to

`x` if the file can be executed (or listed, for a directory)

`-` for a permission not granted

-s Prints size of the file in 512-byte blocks before other file information.

-c Lists time of creation instead of the last modification time. When used with the –**t** option, files are sorted by time of creation (for VMS nodes only). Not valid when the remote host is a 4DDN node.

-U Lists time of last access instead of last modification time. When used with the **-t** (sort by time) option, files are sorted by time of last access.

-1 Prints only one file entry on each line (the default mode for long format).

-r Displays the version level and release date of the *dnls* and the 4DDN product.

-t Sorts by time instead of alphabetically.

5.1.3 Sample *dnls* Entries

The examples below illustrate how to use *dnls* and its options to list files on remote network nodes.

Example

Remote directory listing of a VAX/VMS node:

Input:

```
dnls 'gorgo"lee wiz"::[lee]'
```

Output:

```
DIRECTORY GORGO::DUA1:[LEE]

TEST1.C;2    JFEP.TXT;11    DQU.RNO;1
TEST1.C;1    TEST2.C;1
```

Example

Remote directory listing of a VAX/VMS node using the **–l** (long format) option:

Input:

```
dnls -l -u lee 'gorgo::[lee]'
```

Prompt:

```
PASSWORD: [gorgo::lee]:
```

Output:

```
DIRECTORY GORGO::DUA1:[LEE]

rwxrwxr-xr-- [200,134]       3190  03-MAR-86  14:22  CNTRCHART.RNO;1
rwxrwxr-xr-- [200,134]       7728  03-MAR-86  14:17  DUMPCH3.XXX;3
rwxrwxr-xr-- [200,134]       4090  03-MAR-86  14:17  DUMPCH4.XXX;4

Protections     UIC          Bytes   Modification       Filename
            (User ID Code)           Date     Time
```

Example

Remote directory listing of the user’s login directory, using wildcard, release, filesize, and sort by time options.

Input:

```
dnls -srt ’gorgo"lee wiz"::n*’
```

Output:

```
dnls: version 4.0, date 17 March 1989
    NFARS version 4.0 date 17 March 1989
    NFTL  version 4.0 date 17 March 1989
    Build SGI 4DDN Release 2.0
DIRECTORY GORGO::DUA1:[LEE]
    2 NETSERVER.LOG;48
    3 NETSERVER.LOG;47
    2 NETSERVER.LOG;46
    4 NFARS_INTRO.DOC;2
```

Example

Remote ULTRIX directory listing:

Input:

```
dnls 'mutant"lee wiz"::/u0/usr/lee/'
```

Output:

List of remote files or directories when command is successful. If command is unsuccessful, output is an error message.

5.2 Moving Files with *dnmv*

The *dnmv* command moves files from one directory to another. In the move, you can rename the file and store it in a different directory. You can use *dnmv* to move either a single file or a group of files to a new location. When you move a group of files, you must enter a directory specification as the destination. You cannot use *dnmv* to move files from one network node to another.

The syntax of a *dnmv* command is shown below.

```
dnmv -options source-filespec destination-filespec
dnmv -options file1 file2 ... destination-dirspec
```

You can get on-line help with *dnmv* syntax by entering the command with no arguments:

```
dnmv <return>
```

Note: The *dnmv* command corresponds to the *RENAME* command on VMS systems. It removes a file from its original location and stores it in a new one; no copy of the file is maintained at the original location.

5.2.1 Specifying Files in *dnmv* Entries

Follow the instructions given in Section 3.4, "Entering Remote File Specifications," to enter file specifications in *dnmv* commands. When you enter *dnmv* commands, you can omit the node name and access control information in the destination specification. Since *dnmv* does not move files to a different node, it assumes the source node name and access control information are the same for the destination directory.

You can use wildcard characters in your source specification, if they conform to the rules in effect on nodes containing the source files. If you use wildcards, you must enter a directory specification as the destination.

5.2.2 Using *dnmv* Options

The *dnmv* command offers a number of options that modify the way it works. A *dnmv* entry can include one or more of the options described in the list below, as well as the **-u** (user name), **-p** (password), and **-a** (accountID) options described in the Section 3.4.3, "Specifying Access Information." You can enter options in any order; however, the **-u**, **-p**, and **-a** options affect only file specifications appearing to their right.

Use the options listed below in *dnmv* command entries. For multiple options, you can use a single minus sign, followed by a group of options.

-i Interactive: Before moving each input file, prompts you to confirm the operation by entering one of these options:

`Y` or `y`	Move the file and continue with interactive prompting.
`N` or `n`	Do not move the file and continue with interactive prompting.
`R` or `r`	Move the file and all remaining files without interactive prompting.
`Q` or `q`	Do not move the files and terminate the *dnmv* command.

Any other response skips the current file and proceeds to the next file. The interactive option is particularly useful when you use a wildcard in your file specification.

-l Logging: Displays a success message for each file that is successfully moved.

-n Noisy: Prints a message on the standard error stream indicating when an attempt is made to connect to FAL, the remote file transfer server. This often takes several seconds, and this message provides a way to monitor the operation.

-r Release Number: Displays the version level and release date of the *dnmv* command and the 4DDN product.

5.2.3 Handling *dnmv* Errors

dnmv displays error messages, regardless of the setting of the logging options (**-l**). When an error occurs, a message is displayed in the form shown below:

```
node::file-spec error-description
```

Support for *dnmv* might not be provided by the FALon the destination node. When this is the case, you will see a message containing the phrase "unsupported operation" when you use *dnmv* on files stored at these nodes. If this message appears, check with the system manager of the node where the problem occurred to determine whether *dnmv* is supported.

5.2.4 Sample *dnmv* Entries

The examples below illustrate how to use *dnmv* and its options to move files on a remote network node. As these examples show, there is no output from *dnmv* when it executes successfully, unless you specify an option that generates output. If *dnmv* is unsuccessful, the output is an error message.

Example

Using the VAX node name in the first file specification:

Input:

```
dnmv 'gorgo::dual:[dir1]file1.tst'  'dua1:[dir1]file2.tst'
```

Example

Using the VAX node name in both file specifications:

Input:

```
dnmv 'gorgo::dual:[dir1]file1.tst'
      'gorgo::dual:[dir1]file2.tst'
```

Example

Renaming files from two directories, *dir1* and *dir2*, to another directory, *dir3*:

Input:

```
dnmv 'gorgo::dual:[dir1]file1.tst'
      'dua1:[dir2]file2.tst' '[dir3]'
```

Example

Renaming files on a UNIX node using the logging option:

Input:

```
dnmv mutant::/u0/user1/file1 /u0/user2/file2 -l
```

Output:

```
mutant::/u0/user1/file1 moved to mutant::/u0/user2/file2
```

Example

Wildcard renaming on a VMS node:

Input:

```
dnmv  'gorgo::dual:[dir1]file*.tst' 'dual:[dir2]'
```

Example

Wildcard renaming using an IRIX directory specification:

Input:

```
dnmv 'gorgo::/dua1/dir1/file*.tst' '/dua0/user1/'
```

5.3 Deleting Files with *dnrm*

The *dnrm* command deletes files on remote nodes. Files you delete with this command are permanently removed from the remote directory. You must have write permission to the file in order to delete it.

The syntax of a *dnrm* command is shown below.

`dnrm` **-options** *filespec*

You can get on-line help with *dnrm* syntax by entering the *dnrm* command with no arguments:

```
dnrm <return>
```

5.3.1 Specifying Files in *dnrm* Entries

Follow the instructions given in Section 3.4, "Entering Remote File Specifications," to enter file specifications in *dnrm* commands. You can use a wildcard character in your specification, as long as you observe the wildcard rules in effect on the remote node containing the files.

5.3.2 Using *dnrm* Options

The *dnrm* command offers a number of options that modify the way it works. A *dnrm* entry can include one or more of the options described in the list below, as well as the **-u** (user name), **-p** (password), and **-a** (account-ID) options described in Section 3.4.3, "Specifying Access Information." You can enter options in any order; however, the **-u**, **-p**, and **-a** options affect only file specifications appearing to their right.

Use the options listed below in *dnrm* command entries. For multiple options, you can use a single minus sign, followed by a group of options.

-i Interactive - Prompts you to confirm each file deletion by entering one of the following responses:

`Y` or **`y`**	Delete the file and continue with interactive prompting.
`N` or **`n`**	Do not delete the file and continue with interactive prompting.
`R` or **`r`**	Delete the file and all the remaining files without interactive prompting.
`Q` or **`q`**	Do not delete the file and terminate the *dnrm* command.

-l Logging: Prints a success message after each remote file is deleted, in this form:

```
node::file-spec removed
```

-n Noisy: Prints a message on the standard error stream indicating when an attempt is made to connect to FAL, the remote file transfer server. This often takes several seconds, and the message provides a way to monitor the operation.

-r Release Number: Displays the version level and revision date of the *dnrm* command and the 4DDN product.

5.3.3 Handling *dnrm* Errors

dnrm prints an error message, regardless of the setting of the logging option (**-l**). When an error occurs, a message is displayed in this format:

```
node::file-spec not removed: error-description
```

Support for *dnrm* might not be provided on some remote nodes. When this is the case, you will see a message containing the phrase "unsupported operation" when you use *dnrm* on files stored on these nodes. If this message appears, check with the system manager of the node where the problem occurred to determine whether *dnrm* is supported.

5.3.4 Sample *dnrm* Entries

The examples below illustrate how to use *dnrm* and its options to delete remote files. As these examples show, there is no output from *dnrm* when it executes successfully, unless you specify an option that generates output. If *dnrm* is unsuccessful, the output is an error message.

Example

Deleting a file on a remote VAX/VMS system:

Input:

```
dnrm 'gorgo"sqa"::[sqa]test.obj'
```

Prompt:

```
Password [gorgo::sqa]:
```

(password is not echoed)

Example

Deleting files on a VAX/VMS remote system using wildcard notation and specifying access control information:

Input:

```
dnrm -u sqa -p test 'gorgo::[sqa]*.obj'
```

Example

Deleting files on a VAX/VMS remote system using wildcard notation, interactive prompting, and logging options:

Input:

```
dnrm -l -i 'gorgo"sqa test"::[sqa]*.obj'
```

Interactive output and responses:

```
remove gorgo::[sqa]test1.obj (y/n/r/q) ? y
gorgo::[sqa]test1.obj removed
remove gorgo::[sqa]test2.obj (y/n/r/q) ? n
gorgo::[sqa]test2.obj not removed
remove gorgo::[sqa]test3.obj (y/n/r/q) ? r
gorgo::[sqa]test3.obj removed
gorgo::[sqa]test4.obj removed
gorgo::[sqa]test5.obj removed
```

Example

Deleting multiple files on two different VAX/VMS nodes:

Input:

```
dnrm -n ’gorgo"sqa1 test1"::[sqa1]*.obj’
               -u sqa2 -p test2  ’galaxy::[sqa2]test.*’
```

Output:

```
Waiting for response from FAL on node gorgo ...ok
Waiting for response from FAL on node galaxy ...ok
```

Example

Deleting a file on a remote IRIX or ULTRIX node:

Input:

```
dnrm ’mutant"sqa test"::/u0/usr/sqa/hello.o’ -l
```

Output:

```
mutant"sqa"::/u0/usr/sqa/hello.o removed
```

Chapter 6

Copying Files over the Network

The *dncp* command copies files between a local and remote node or between two remote nodes in the network. You can also specify an input device such as a keyboard as the source of the copy, or an output device such as a display screen as the destination.

In addition to copying data files, *dncp* also copies executable files to remote nodes for processing, if you use the **-x** option. This option differs from the *dnex* command, in that the target file in a *dncp* command need not reside on the node where it is executed. When you use *dncp* to execute a file, a success message indicates that the file was successfully submitted for execution, not that it executed successfully.

The syntax for a *dncp* command is shown below.

`dncp` **-options** *file1 file2*
`dncp` **-options** *file... directory*

You can get on-line help with *dncp* syntax by entering
the command with no arguments:

```
dncp <return>
```

6.1 Specifying Files in *dncp* Entries

The source and destination files in a copy operation are specified by either pathnames to local files or by remote file specifications. Follow the instructions given in Section 3.4, "Entering Remote File Specifications," to enter remote file specifications.

You can use wildcard characters in a source file, but not in destination file specifications. When you use wildcards, observe the wildcard rules in effect at the node where the files reside.

Note: When you use *dncp* to copy an executable file (**-x** option) to your local node for processing, enter a fully qualified file specification (see Section 3.4, "Entering Remote File Specifications," for details) to your node as the destination for the copy operation. If you do not use a full file specification, the file will be copied to your node but it will not be scheduled for execution.

The source and destination files specified in a *dncp* command follow these conventions:

file1, file	The local or remote file specification for the source of the file to be copied. The specification can contain wildcard characters. *file1* cannot be a directory specification. To specify an input device (such as a keyboard) as a source, use a hyphen (-) for this specification.
file2	The local or remote file specification for the destination of the file to be copied. This file specification must not include wildcard characters. To specify an output device (such as a display screen) as a destination, use a hyphen (-) for this specification.
directory	The local or remote directory specification for the destination of the files to be copied. A remote directory specification must include all trailing punctuation (everything up to the first letter of the filename that normally follows the directory).

6.2 Using *dncp* Options

The *dncp* command offers a number of options that modify the way it works. A *dncp* entry can include one or more of the options described in the list below, as well as the **-u** (user name), **-p** (password), and **-a** (account ID) options described in Section 3.4.3, "Specifying Access Information." You can enter options in any order; however, the **-u**, **-p**, and **-a** options affect only file specifications appearing to their right.

Use the options listed below in *dncp* command entries. For multiple options, you can use a single minus sign, followed by a group of options.

-v Verbatim: Transfers all the input files byte for byte, without record format conversion and with no bytes lost, altered, or inserted. Output files are created with a record format appropriate to their byte-stream nature. Default maximum record length is 1024 bytes. On VAX/VMS systems, the output files always have FIXED RECORD format and NO RECORD attributes.

> **Note:** You must use the **v** option to transfer binary files. Also use **-v** for faster copying between UNIX systems.

-b Bytes: Allows you to specify the maximum fixed record length of your files. Overrides the VERBATIM default maximum of 1024 bytes.

-i Interactive: Prompts you to confirm each copy operation by entering `Y` or `y` for yes, or `T`, `t`, or `1` for true. Any other response skips the current file and proceeds to the next file. The interactive option is particularly useful when you enter a wildcard specification.

-l Logging: Prints logging information to indicate the start of data transfer in this form:

source-file `=>` *destination-file*

n Noisy: Prints a message on the standard error stream indicating when an attempt is made to connect to FAL, the remote file transfer server. This often takes several seconds, and the message provides a way to monitor the operation.

-r Release Number: Displays the version level and release date of the *dncp* command and the 4DDN product. If the release number is the sole argument, *dncp* prints release information and terminates.

-t	Total: Prints the total number of bytes and files transferred.
-x	Copies files to a remote node for execution and deletes them after they are executed.
-z	Appends all data to the destination file rather than overwriting it. Use a separate command entry for multiple appends to a single target file.
-	Specifies an input device (such as a keyboard) for the source, or and output device (screen display) for the destination.

6.3 Creating the Output File

Unless you use the **-z** option, a *dncp* command always creates a new file. If you copy to a VAX/VMS system and a file by the name you specified already exists, a new version of the file is created. If you copy to a UNIX system and a file by the name you specified already exists, the existing file is replaced by the new file, unless it is write-protected. You must have write permission on the directory where the new file is to be created.

The protection mode of the output files is the default protection mode in effect for the accessing user. The creation and modification times for the output file are the time of the creation of the new file.

Do not try to copy multiple input files to a single output file. Multiple input files can be copied only to a directory or to the standard output device.

When you copy files between two remote nodes, the newly created files have the same record attributes as if they had originated on the local 4DDN system.

6.3.1 Copying to 4DDN Nodes

When remote files are copied to a target directory on a 4DDN node, their names are converted, if necessary, to names that are suitable for use on the IRIX system. Files are stripped of version numbers, and, if the files are from a non-case-sensitive system like VAX/VMS, they are converted to lowercase. For example, "DUA1:[LEE.PROJ1]MYFIL.RNO" on a VMS system becomes "myfil.rno" on an IRIX system. The names of files from other case-sensitive systems, such as ULTRIX, are unchanged.

IRIX files contain record-structured character data, that is, lines of text or source code. On VAX/VMS systems, files may be stored in a variety of formats; file and record format information is stored as part of the file. Three types of VAX/VMS file formats can be copied to a 4DDN node: Variable Fixed Control (VFC) format; Fixed Record Format with NO RECORD attributes; and Variable Record Format with Carriage Return, Carriage Control record attributes.

By default, when these VMS files are copied to a 4DDN node, they are written in line-per-record mode. During the copy, *dncp* coverts the file, stripping record attributes and replacing VAX/VMS record-delimiters with IRIX-style line delimiters (linefeed or ASCII 10). Copying with conversion (the *dncp* default) is effective for transferring ASCII files from VMS to 4DDN nodes, but it can be harmful to binary files. However, you can inhibit conversion by using the *dncp* verbatim option (**-v**). In verbatim mode (also referred to as *image mode* in VAX/VMS nomenclature), *dncp* copies files byte-for-byte, including record delineation bytes, from the VAX/VMS to the 4DDN node.

Follow these recommendations for VAX/VMS to 4DDN node copies:

- Use the *dncp* default (conversion mode) to copy ASCII files.
- Use *dncp* **-v** (verbatim mode) to copy binary files.
- Use *dncp* **-v** to copy between UNIX-based systems.

  Verbatim transfers between UNIX-based systems (such as IRIX to ULTRIX) are faster, since no conversion is performed.

6.3.2 Copying to VAX/VMS Nodes

When you copy a file in conversion mode (the default) from a 4DDN node to a VAX/VMS node, the file created on the VAX/VMS node is in Variable Record Format, with the Carriage Return, Carriage Control record attribute. When you copy a file in verbatim mode from an IRIX system to a VAX/VMS system, the new VAX/VMS file is in Fixed Record Format with NO RECORD attributes.

6.4 Setting File Permissions

In general, files copied between VAX/VMS and IRIX nodes maintain the permissions that were set on the source node. The exceptions are that IRIX does not recognize the *system* user, and it ignores the *delete* permission.

If a VMS host is running a file access list, frequently its files have no permissions set. When these files arrive at an IRIX node, their permissions are set to 000, preventing user access.

To determine the permissions on a file after it is copied to an IRIX node, use the *ls -l* command. If you need to reset its permissions, use the *chmod*(1) command. The permissions you assign with *chmod* will be maintained when the file is returned to a VMS host.

6.5 File Transfer Errors

When an error occurs during a file transfer operation, the message that is printed is either a *dncp*, NFARS, or task-to-task error message. Information on handling *dncp* errors is given in Section 6.4.2. Information on handling NFARS and task-to-task error messages is given in the *4DDN Programming Guide*, in the sections entitled "NFARS Error Messages" and "4DDN Error Codes."

6.5.1 Printing Local IRIX Errors

In addition to *dncp*, NFARS, and task-to-task error messages, operations on local files can generate local IRIX error messages. These errors are printed by calls to *perror(3C)*. Information on using *perror* is given in the *intro(2)* manual page in the *IRIS-4D Programmer's Reference Manual*.

6.5.2 Handling *dncp* Errors

Whenever *dncp* fails to complete the requested operation, diagnostic messages are printed on the standard output device (this is usually your display screen or console). In addition, a status value is returned to the shell. A status value equal to 0 means that the *dncp* command was successfully completed. Any other value indicates a failure.

The *dncp* error messages are listed below. Each message includes an explanation and recommended action. For some error conditions, it may be necessary to consult your system manager. You should advise your system manager whenever a message indicates a software error.

Error Message	`operation LOCAL to LOCAL not supported`
	`operation STDIO to LOCAL not supported`
	`operation LOCAL to STDIO not supported`
	`operation STDIO to STDIO not supported`

Meaning:	Copying within the local node is not supported.
Recommended Action:	Use *cp*(1), *cat*(1), or an editor.

Error Message	`source filespec cannot be a directory`
Meaning:	*dncp* determined that the source filespec is a directory and this is not permitted. The source of any transfer must be a file specification that results in a list of files.
Recommended Action:	Respecify the source as a list of files.

Error Message	`remote wildcards require a destination directory`
Meaning:	A destination directory is required when either multiple source files or a source file specification including wildcards is specified. This restriction applies even if a wildcard file specification results in a single file.
Recommended Action:	Specify a directory as the destination. Use a trailing forward slash (/) if the destination directory is on a UNIX or ULTRIX node.

Error Message	`Failed: <source> => <destination>`
Meaning:	The copying operation for the indicated file failed. The reason for the failure is explained by an earlier message.
Recommended Action	Check earlier error messages to determine the cause of the failure and take the appropriate action to correct it.

Error Message	`can’t delete old destination file`
	`can’t rename destination file`

Meaning:	The destination file already exists. When you copy a file, a temporary working file is created to protect the data as it is transferred. When the transfer is complete, the temporary file replaces the target file. These messages indicate that the target file cannot be deleted, or that the temporary file cannot be renamed to the target destination file.
Recommended Action:	Check permissions on the destination file and directory.

Error Message	`dncp: stdin and interactive are inconsistent`
Meaning:	*dncp* determined that the given command requires both interactive mode and input from standard input to be used at the same time. The *dncp* command does not simultaneously support these operations.
Recommended Action:	Omit the interactive option (**-i**) with *stdin* (**-**).

Error Message	`dncp: stdin to stdout is not permitted`
Meaning:	*dncp* determined that both *stdin* and *stdout* would be used simultaneously as the source and destination for a transfer. The *dncp* command does not support this operation.
Recommended Action:	Use one of the operating systems text editors or similar facilities to create the source file.

6.6 Sample *dncp* Entries

The samples below illustrate how to use *dncp* and its options to copy files to and from various network nodes.

Example

Copying a local IRIX file to a remote VAX/VMS file using embedded access control information:

Input:

```
dncp -ln /usr/max/bow.c  ’gorgo"lee wiz"::[lee]test1.c’

         Local IRIX       Remote node      Remote VAX
         filespec         name with user   filespec
                          and password
```

Output:

```
Waiting for response from FAL on node gorgo ... ok
/usr/max/bow.c => gorgo"lee"::[lee]test1.c;1
```

Example

Copying a local IRIX file to a remote VAX/VMS file with access control information specified as options:

Input:

```
dncp -ln /usr/max/bow.c -u lee -p wiz 'gorgo::[lee]test1.c'

         Local IRIX       User and           Remote VAX
         filespec         password           filespec
```

Output:

```
Waiting for response from FAL on node gorgo ... ok
/usr/max/bow.c => gorgo"lee"::[lee]test1.c;1
```

Example

Copying a local IRIX file to a remote VAX/VMS file with the user name specified by the environment variable:

Input:

```
csh: setenv NET_USER LEE
sh:  NET_USER=LEE ; export NET_USER
dncp -ln /usr/max/bow.c  'gorgo::[lee]test1.c'

          Local IRIX         Remote VAX
          filespec           filespec
```

Prompt:

```
Password [gorgo::lee]:     (password not echoed)
```

Output:

```
Waiting for response from FAL on node gorgo ... ok
/usr/max/bow.c => gorgo"lee"::[lee]test1.c;1
```

Example

Copying a remote VMS file to a local IRIX file using logging and noisy options:

Input:

```
dncp -ln -u lee -p wiz   'gorgo::[lee]mad.exe'  /usr/lee/test

          Access info    Remote VAX filespec   UNIX target file
```

Output:

```
Waiting for response from FAL on node gorgo ... ok
gorgo"lee"::[lee]mad.exe => /usr/lee/test
```

Example

Copying local IRIX files to a remote directory using wildcards and the logging and noisy options:

Input:

```
dncp -ln /usr/max/*.c   'gorgo"lee wiz"::user:[lee]'

         Local IRIX       Remote VAX/VMS directory
         filespec         with access information
```

Output:

```
Waiting for response from FAL on node gorgo ... ok
/usr/max/bow.c => gorgo::user:[lee]bow.c
/usr/max/yow.c => gorgo::user:[lee]yow.c
/usr/max/zow.c => gorgo::user:[lee]zow.c
```

Example

Copying local IRIX files to a remote directory using wildcards and the logging option.

Input:

```
dncp -l   *.h  *.c      'gorgo"lee wiz"::user:[lee]'

         Local IRIX     Remote VAX/VMS directory
         filespec       with access information
```

Output:

```
foo.h =>  gorgo"lee"::[lee]foo.h
bar.h =>  gorgo"lee"::[lee]bar.h
bow.c =>  gorgo"lee"::[lee]bow.c
yow.c =>  gorgo"lee"::[lee]yow.c
```

Example

Copying a file to the current working directory on the local node:

Input:

```
dncp -lnr 'gorgo::[max]hello.c'  .

           Remote filespec      Current local
                                directory
```

Output:

```
dncp: version 4.0 date 17 March 1989
      NFARS: Version 4.0 date 17 March 1989
      NFTL:  Version 4.0 date 17 March 1989
      Build SGI 4DDN Release 2.0

Waiting for response from FAL on node gorgo ... ok
gorgo::[max]hello.c => ./hello.c
```

Example

Copying a remote file to the standard output device (*stdout*):

Input:

```
dncp -u lee 'gorgo::[max]hello.c'  - |  more

            Remote filespec   Standard output
                              as destination
```

Prompt:

```
Password [gorgo::lee]:     (password not echoed)
```

Output:

```
main ( )
{
               printf  ("Hello world0);
}
```

Example

Printing local IRIX files on a remote VAX/VMS printer. The output from a *pr* command is piped (|) to redirect it to a printer on GORGO:

Input:

```
pr -l60 *.h *.c | dncp -lt - gorgo::lca0:source.list

                      Standard    Remote printer
                      input       with named listing
```

Output:

```
STDIO => gorgo::lca0:source.list
1 file (32596 bytes) copied.
```

Example

Copying a remote file to a local IRIX directory with a password prompt:

Input:

```
dncp -ln -u lee 'gorgo::[lee]mad.exe'    /bin/

      Access     Remote VAX              Target
      control    filespec                directory
                                         spec
```

Prompt:

```
Password [gorgo::lee]:     (password not echoed)
```

Output:

```
Waiting for response from FAL on node gorgo ... ok
gorgo"lee"::[lee]mad.exe => bin/mad.exe
```

Example

Copying a group of local files to a directory on a remote node using wildcard notation and the logging option:

Input:

```
dncp -l data*.tst 'mutant::/tmp/'
```

Output:

```
/data1.tst => mutant::/tmp/data1.tst
/data2.tst => mutant::/tmp/data2.tst
/data3.tst => mutant::/tmp/data3.tst
```

Example

Copying a local IRIX file to a remote IRIX or ULTRIX file:

Input:

```
dncp -ln bow.c ’mutant"sqa test"::/uO/usr/sqa/test1.c’

       Local IRIX         Remote UNIX filespec
      filespec            with access information
```

Output:

```
Waiting for response from FAL on node mutant ... ok
bow.c => mutant"sqa"::/u0/usr/sqa/test1.c
```

Example

Copying a remote IRIX or ULTRIX file to a local IRIX file in the current working directory:

Input:

```
dncp -ln ’mutant::/u0/usr/sqa/test1.c’ .
```

Output:

```
Waiting for response from FAL on node mutant ... ok
mutant::/u0/usr/sqa/test1.c => ./test1.c
```

Chapter 7

Executing Remote Files

The remote file execution command, *dnex*, locates command files on remote nodes and submits them for execution on the remote node. The responding node executes the files as if they were submitted locally. The responding remote node can be any VMS, IRIX, or ULTRIX node running the File Access Listener.

A success message from *dnex* means that your files were submitted successfully to the responding node. It does not mean that they were executed successfully.

The syntax of a *dnex* command is shown below.

`dnex` **-options** *file-specification ...*

You can get on-line help with *dnex* syntax by entering the command with no arguments:

```
dnex <return>
```

7.1 Specifying Files with *dnex*

Follow the instructions given in Section 3.4, "Entering Remote File Specifications," to enter file specifications in *dnex* commands. You can specify one or more remote files for which you have executable permissions, and your entries can include wildcards.

7.2 Using *dnex* Options

You can use a number of options with *dnex* commands. Your entry can include one or more of the options listed below. In addition to the options in this list, you can also use the **-u** (user), **-p** (password), and **-a** (account ID) options described in Section 3.4.3, "Specifying Access Information."

-i Interactive: Before submitting each file to the batch queue on the remote node, it prompts you to confirm the operation by entering one of these options:

Y or **y** Submit the file and continue with interactive prompting.

N or **n** Do not submit the file and continue with interactive prompting.

R or **r** Submit the file and all remaining files without interactive prompting.

Q or **q** Do not submit the files and terminate the *dnex* command.

Any other response skips the current file and proceeds to the next file.

-l Logging: Displays a success message for each file that is successfully submitted.

-n Noisy: Prints a message on the standard error stream indicating when an attempt is made to connect to FAL, the remote file transfer server. This often takes several seconds, and this message provides a way to monitor the operation.

-r Release Number: Displays the version level and release date of the *dnex* command and the 4DDN product.

7.3 Handling *dnex* Errors

If an error occurs while *dnex* is executing, the system displays an error message, regardless of the setting of the logging option (**-l**). Error messages have the format shown below. Error descriptions are the same as those displayed for *dncp* commands, explained in Chapter 6, "Copying Files over the Network."

```
node::file-spec not queued error-description
```

Support for *dnex* may not be provided on some nodes. When this is the case, you will see a message containing the phrase "unsupported operation" when you use *dnex* on files stored at these nodes. If this message appears, check with the system manager of the node where the problem occurred to determine whether *dnex* is supported.

7.4 Sample *dnex* Entries

The examples below illustrate how to use *dnex* and its options to submit command files for execution on a remote node.

Example

Submitting a file for execution on a remote VAX/VMS node:

Input:

```
dnex 'vmsnod"sqa"::[sqa]test.com'
```

Prompt:

```
PASSWORD [vmsnod::sqa]:
```

(password is not echoed)

Example

Submitting multiple files for execution on different VAX/VMS nodes. This example assumes the same user name and password are valid for both systems:

Input:

```
dnex 'vmsnod"sqa1 test1"::[sqa1]*.com' 'boris::[sqa2]test.*'
```

Output:

No output when command is successful. If command is not successful, output is an error message.

Example

Submitting a file for execution on a remote ULTRIX node, using the logging option:

Input:

```
dnex -l 'ultrix"sqa test"::/u0/usr/sqa/a.out'
```

Output:

```
ultrix"sqa"::/u0/usr/sqa/a.out queued
```

Chapter 8

Printing over the Network

The 4DDN print command, *dnlp*, allows you to print local or remote files on any printer directly attached to a DECnet node, including your local node. Any VAX/VMS files you want to print at a 4DDN node printer must be in either variable record format or variable fixed control format.

4DDN uses the services of a local spooling program to provide the network printing service. 4DDN software assumes that you are using *lp*, the standard IRIX spooling program. However, if you are using a spooling program other than *lp*, you might need to modify */usr/etc/dn/dnlp.cf*, the configuration file for *dnlp*. The *4DDN Network Management Guide* explains how to determine whether your printer configuration file should be modified and gives instructions for changing it.

The syntax of a *dnlp* command is shown below.

```
dnlp [-options] file-spec
```

You can get help with *dnlp* syntax by entering the command with no arguments:

```
dnlp <return>
```

8.1 Specifying Files with *dnlp*

Follow the instructions given in Section 3.4, "Entering Remote File Specifications," to enter file specifications in *dnlp* commands. You can use wildcard characters in your specification, as long as you observe the wildcard rules in effect on the remote node containing the files.

8.2 Using *dnlp* Options

The *dnlp* command offers a number of options that modify the way it works. A *dnlp* entry can include one or more of the options described below, as well as the **-u** (user name), **-p** (password), and **-a** (account ID) options described in Section 3.4.3, "Specifying Access Information." You can enter options in any order; however, the **-u**, **-p**, and **-a** options affect only file specifications appearing to their right.

Use the options listed below in *dnlp* command entries. For multiple options, you can use a single minus sign, followed by a group of options.

-n Specifies the node where the file to be printed resides.

-q Specifies the print device, in the form of a file specification, where the files are to be queued for printing. If you enter a **-q** and specify a node without specifying the device, files are queued at the default printer on the node you specified.

> **Note:** The name of a print queue is not a valid argument to **-q**. You must specify, in the form of a file name, the port to which the printer is connected.

-r Displays the version level and release date of the *dnlp* command and the 4DDN product.

8.3 Printing Local Files

You can use the *dnlp* command to print local files on printers attached to your node, or on printers attached to remote nodes. The sections that follow explain how to use *dnlp* to send local files to local and remote printers.

8.3.1 Sending Local Files to Local Printers

When you use *dnlp* for local printing, your file is sent to the default printer on your local node, unless you specify a nondefault printer. To specify a nondefault printer, use the **-q** option with the printer name.

The examples below illustrate how to print local files to the default and nondefault printers on your local node.

```
dnlp file1 file2               (uses default printer)
dnlp -q laser3 file3 file4     (uses specified printer, laser3)
```

In the examples above, local files *file1* and *file2* are sent to the default printer. Local files *file3* and *file4* are sent to a nondefault printer, *laser3*.

8.3.2 Sending Local Files to Remote Printers

You can send local files to be printed on either the default or nondefault printers attached to remote network nodes. To use the default printer on the remote node, specify the **-q** (queue) option, the name of the remote node where the printer is attached, and the name of the file to be printed. No printer name is required in your entry.

The example below illustrates how to send a local file to a default printer at a remote node.

```
dnlp -q vax1:: file5
```

In the example above, local file *file5* is sent to the default printer on the remote node *vax1*.

To use a nondefault printer on the remote node, you must specify the name of the port where the printer is connected. The example below illustrates how to send a local file to a nondefault printer on a remote VMS node.

```
dnlp -q vax1::lca0: file6      (lca0 is a device)
```

In the example above, local file *file6* is sent to a nondefault printer connected at port *lca0* on the remote *vax1* node. A nondefault printer is specified by a device rather than a print queue name.

8.4 Printing Remote Files

You can use the *dnlp* command to print remote files on your local printer and on printers that are directly attached to remote network nodes. The sections that follow explain how to use *dnlp* to send remote files to local and remote printers.

8.4.1 Sending Remote Files to Local Printers

To send a remote file to the default printer attached to your local node, enter the *dnlp* command with the file specification for the remote file you want to print and the access control information for that file.

The examples below illustrate equivalent entries to print a remote file on your local printer.

```
dnlp -n node1 -u joe -p montana file1
dnlp 'node1"joe montana"::file1'
```

Both examples above print the file *file1*, stored on the remote node *node1* to the default printer on the local node (node where the command was entered). In the first example, the user name *joe* and password *montana* are entered as *dnlp* command options; whereas, in the second example, the access control information is entered in the file specification.

8.4.2 Sending Remote Files to Remote Printers

You can send remote files to be printed on a default or nondefault printer attached to a remote network node. To send a file to a default printer, enter the *dnlp* command with the **-q** option, the name of the remote node where the printer is attached, and the remote file specification for the file you want to print.

The example below illustrates how to send a remote file to a default printer on a remote node.

```
dnlp -q vmsnod:: ’magic"tom lunar"::file1 -n photon file2 file3’
```

In the example above, the default printer on remote node *vmsnod* is used to print three files. *file1* resides on magic, and *file2* and *file3* reside on *photon*. The access control information is *tom* (user name) and *lunar* (password) on both machines.

To send a remote file to a nondefault printer on a remote node, you must specify the printer name in your entry. If the remote node is a VAX/VMS node, use the device name to specify the printer.

The example below shows how to print a remote file to a nondefault remote printer.

```
-1dnlp -q vmsnod::txb6: ’magic"tom lunar"::file1 -n photon file2 file3’
```

8.5 Handling *dnlp* Errors

If an error occurs while a *dnlp* command is executing, it posts an error message. The format of *dnlp* error messages is shown below.

```
Error:  node::file-spec not queued  error-description
```

Support for *dnlp* might not be provided on some remote nodes. When this is the case, you will see a message containing the phrase "unsupported operation" when you try to send files to printers at these nodes. If this message appears, check with the system manager of the node where the problem occurred to determine whether *dnlp* is supported.

The success of *dnlp* means that your file was queued successfully. It does not mean that it has successfully printed. Check the status of the print queue if you want further verification.

Error messages for *dnlp* are given in Table 8-1.

Error	Meaning	Recommended Action
Account too long	You used an account name or number that is longer than 16 characters.	Try the command again with a correct account.
Could not open input file *file-spec*	You used a filename that was either incorrectly spelled or not in the designated directory.	Correct the filename or directory specification and try the command again.
Could not open output device *device*	You used an output device specification that was incorrect.	Try the command again with a correct output device specification.
Could not open /usr/etc/dn/dnlp.cf	The *dnlp.cf* file was not found.	Check */usr/etc/dn* for the *dnlp.cf* file. If *dnlp* is not there, create a copy of *dnlp* and define your printer (see the *4DDN Network Management Guide* for directions).
Invalid configuration file	The configuration file does not properly define the printer.	Correct the configuration file */usr/etc/dn/dnlp.cf* (see the *4DDN Network Management Guide* for directions).
Invalid option *opt* ignored	You used an option that is not appropriate to the situation and *dnlp* is ignoring it.	Try the command again with a valid option.
Node name too long	You used a node name that was longer than 6 characters.	Try the command again with a correct node name.
Password too long	You used a password that was longer than 32 characters.	Try the command again with a correct password.
Printer spec. invalid	You used a printer specification that is not acceptable.	Try the command again with a valid printer specification.
User name too long	You used a username that was longer than 32 characters.	Try the command again with a correct username.

Table 8-1 *dnlp* Error Messages

Chapter 9

Using 4DDN Mail Service

4DDN mail service enables mail delivery between IRIS and VMS nodes on a DECnet network. The software that provides 4DDN mail service is configured to work with *sendmail*, a standard mail utility shipped with your IRIS system.

When integrated with *sendmail*, the 4DDN mail program, *dnMail*, acts as an extension to the mail program you currently use. Your mail program will work exactly as it always has, except that you will be able to send and receive mail from the VMS and other 4DDN nodes in your DECnet network. When you are sending mail to a DECnet node, you will need to use a 4DDN address specification.

In order to use 4DDN mail, */usr/lib/sendmail.cf*, the configuration file for *sendmail*, must contain 4DDN entries. If you or your system administrator have not already done so, check your copy of *sendmail.cf* to be sure it contains the entries you need to send and receive DECnet mail. Instructions for checking and modifying *sendmail.cf* are given in the *4DDN Network Management Guide*.

9.1 How *dnMail* Works

UNIX mail systems generally rely on four kinds of programs to provide mail services: a user interface that allows people to compose outgoing messages and read and manage incoming ones; a preprocessor that translates addresses and routes them to local or remote delivery services; a local delivery service that deposits mail as a data file in a user account; and a transport service that delivers mail to remote users. Different programs from each functional group can be combined to provide mail service for a given UNIX system.

With *dnMail*, the user interface and local delivery services are performed by mail programs currently implemented on your system, such as */usr/sbin/Mail* and */bin/mail*. Address translations and routings are performed by *sendmail*. If the destination address on your message is a local user (someone else on your IRIS system), *sendmail* gives the message to the local UNIX delivery service. If the destination address is a remote 4DDN or VMS node, *sendmail* gives the message to *dnMail* for delivery.

9.2 Invoking 4DDN Mail Service

You invoke 4DDN mail by giving the usual command to your regular mailer and addressing the mail to its recipient on the DECnet node. For example, if your usual mail command looks like this:

`Mail` *recipient*

you would also type **`Mail`** with the recipient's name to send mail to a 4DDN or VMS node. Your regular mail program will call *sendmail*, which will invoke *dnMail* automatically when it is needed.

If you prefer, you can invoke *dnMail* directly, rather than invoking your regular mailer. However, if you use the user interface to *dnMail*, you must use standard VMS address formats on your messages. Use this command to send mail with the *dnMail* user interface:

`dnMail` *node*`::`*recipient*

9.3 Addressing *dnMail* Messages

You can address *dnMail* messages to any destination that can normally be reached through VMS mail services. The address on your messages can be in Internet format or standard VMS mail format.

VMS address formats are given below. You can use either node names or node addresses in your address.

node-name`::`*user-name*
node-number`::`*user-name*
area-number.node-number`::`*user-name*

Internet address format is given below. When you use Internet format, it must end with the ASCII string *.4ddn*, the pseudo-domain name for the *dnMail* mailer.

user-name@node-name`.4ddn`
user-name@node-address`.4ddn`

These samples illustrate what your mail addresses will look like:

```
mary@tonto.4ddn
tonto::mary
1.232::mary

mary@1.207.4ddn
1.207::mary
```

9.4 Sending *dnMail* Messages

To send a *dnMail* message, you start your mail program with the usual command, then use its message composer as you normally do to create your message. Since *dnMail* sends and receives messages using variable record format, you must limit the line lengths in your message to 512 characters or less.

Appendix A

IRIX Manual Pages

This appendix contains the IRIX manual pages that pertain to using 4DDN software.

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 007-0820-030.

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