NFT User's Guide

HP OfficeShare Network

HEWLETT
PACKARD

Manual Part Number: 50923-90001

Printed in U.S.A., February 1988

Notice

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

Hewlett-Packard Co.
3404 E. Harmony Rd.
Fort Collins, CO 80525 U.S.A.

Printing History

February 1988 . . . Edition 1

Table of Contents

Chapter 1: Using This Guide

Chapter 2: Using NFT

Chapter 3: NFT with the HP 1000 Computer

Chapter 4: NFT with the HP 3000 Computer

Chapter 5: NFT with the HP 9000 Computer

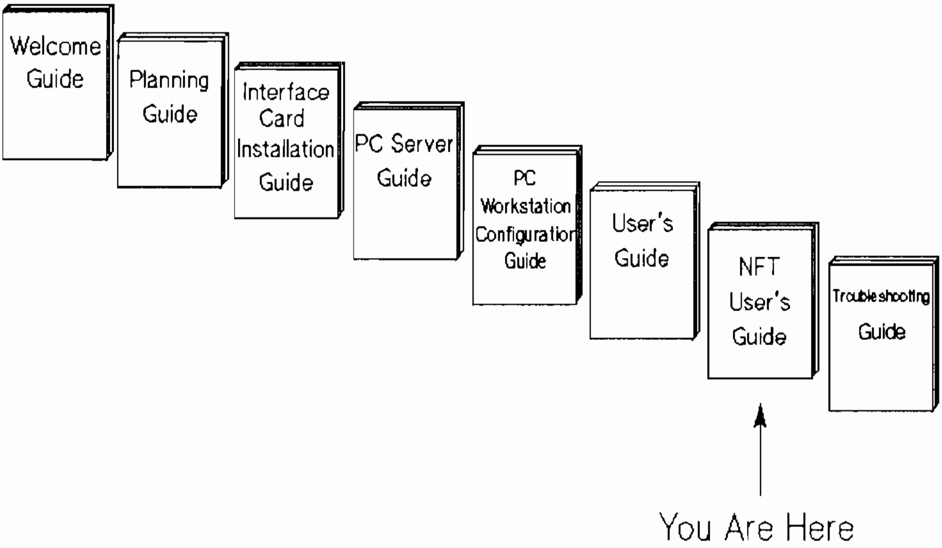
Chapter 6: NFT with the DEC VAX Computer

Chapter 7: Using NFT Options

Notes

1

Using This Guide



Guide Overview

This guide is part of the HP OfficeShare set of guides. It tells you how to use the Network File Transfer service. The Network File Transfer service, called NFT, allows you to copy files between your PC workstation and other computers on the network.

With NFT, you can use your network to exchange files with other computers that support Hewlett-Packard's AdvanceNet NFT. NFT cannot be used to transfer files from one PC workstation to another.

How to Use This Guide

For basic information about NFT:	See chapter 2, "Using NFT". This chapter explains what NFT is and how it works. It covers the basic syntax of the **DSCOPY** command.
For NFT syntax information particular to the computer with which you want to exchange files:	See chapters 3 through 6. These chapters cover the exact syntax requirements for working with each system that supports NFT.
For technical details about NFT between specific systems:	See the *NS Cross-System NFT Reference Manual*. This technical manual covers NFT between each pair of supported systems. It contains full technical details about the effect of each NFT option in the context of each pair of supported systems.
For references to other related documentation:	See the chapter titled "Additional References" in the *Welcome Guide for LANs, HP OfficeShare Network*

Before You Begin

The information in this guide assumes the following:

- The network hardware and software is properly installed and working correctly on your PC workstation.
- You have a User Reference Worksheet that:
  - describes the network services available to your PC workstation.
  - provides the names of any remote computers with which you can exchange files.

The person who configured your PC workstation on the network should complete this form.

Conventions Used in This Guide

UPPERCASE — Bold uppercase text represents a command in DOS or the network software.

[ ] — Items in square brackets are optional.

italics — Italicized text represents the proper name of manuals.

`Computer font text` — Text shown in computer font represents screen text; that is, prompts and messages that you see on the screen.

Enter — This type face represents keys to press on the keyboard. The key has this label.

CTRL-Alt-DEL — Keystroke characters connected by hyphens mean that you press all indicated keys at the same time.

<*lower case*> — A lower-case italicized word between arrows (< >) means that you need to choose a name to use in place of the word or words.

A First Look at This Guide

Chapter 1: Using This Guide
No matter how you work with NFT, be sure to at least glance through this chapter. It provides a guide to using this manual, and references for further reading.

Chapter 2: Using NFT
This is the heart of what you need to know to use NFT. This chapter covers all the basics of how you put NFT to work. There are lots of examples in this chapter to help you learn. This chapter also shows you what to do if you encounter difficulties.

Chapters 3, 4, 5, and 6: The NFT Partners
These chapters show you the details for using NFT with each of the NFT partners. In each chapter, you get a concise review of the basics, followed by information targeted for a particular NFT partner. Each chapter covers a different NFT partner:

- Chapter 3: The HP 1000 computer
- Chapter 4: The HP 3000 computer
- Chapter 5: The HP 9000 computer
- Chapter 6: The DEC VAX computer

Chapter 7: Using NFT Options
This chapter is for users who want or need to tailor NFT to meet special demands. Casual users of NFT will want to know about a group of options called the **conventional** options, which can make using NFT easier. The other options make NFT very flexible. However, you will probably need some technical background to use them effectively.

Appendix A: NFT Error Messages
Whenever NFT detects an error, it lets you know about it. This appendix catalogs the error messages, and tells you how to interpret them.

Appendix B: Notes for Programmers
Applications programmers should read this appendix. The information here applies whenever you load and execute NFT from another application.

2

Using NFT

By using NFT on your PC workstation, you can copy files between your PC and another computer, or between two remote computers on your network.

Some background in these areas could be helpful as you learn NFT:

- DOS commands, such as **COPY** or **RENAME**
- Active drives and directories on the PC workstation
- Directories and subdirectories on the PC workstation
- Shared discs (for PC LAN users)

Chapter Overview

This chapter explains the syntax and basic use of NFT. It contains:

- A quick reference page for experienced users
- Definitions
- Before You Begin
- How to use the **DSCOPY** command to transfer files
- Two common kinds of file transfers: default and binary. When and how to use each kind
- Security and protection measures
- What to do if a transfer fails

Quick Reference

To use NFT:

1. Load the LAN software and have the NFT software ready to execute.

2. To issue a **DSCOPY** command, use this general syntax:

   `dscopy [options]` *<source_file> <target_file>* **Enter**

   Options are covered in the chapter 7, "Using NFT Options". Either the source or target (or both) can be a remote file.

3. Specify local files exactly the same way as for any other purpose. See your DOS documentation for details on the DOS file naming scheme.

4. Remote files use the following syntax:

   ```
   node_name#login#file_name
   ```

   a. the **Node_name** has the following syntax, which can often be abbreviated to only the first field, without any periods:

   ```
   nodename.domain.organization
   ```

   b. the **login** identifies an authorized user by name and by one or more passwords. The syntax depends on your NFT partner. See the appropriate chapter of this manual.

   c. the **file_name** syntax depends on your NFT partner. See the appropriate chapter of this manual.

   Use the binary option, **-B**, for remote storage and binary data transfers. Use default transfers for copying text files over the network.

   For password security on remote computers, use password prompting. This prevents passwords from being visible on the screen of the PC workstation.

Definitions

Computers on the Network

Node

A **node** is just another name for any computer on the network. It can be either a PC workstation or another computer.

Local computer

The **local** computer is the one you are working on. A file on the PC is called a **local** file. If you did not have a network, you would still have your local computer. For our purposes, your PC workstation is *always* the local computer.

Remote computer

All the other computers on the network are **remote**. A file on a remote computer is called a **remote** file. The resources of remote computers are only available over the network. A PC workstation is *never* the remote computer for NFT.

The Three Roles in a File Transfer

There are three roles in every network file transfer:

The initiator

This computer reads and interprets your command. It initiates the process of copying the file from one computer to another. Its only function is to read your command and later notify you of the results. The PC workstation is *always* the initiator. It is frequently the producer or consumer as well.

The producer

This computer has the original file that you want to copy. It produces a copy of the file in the same sense that a printer produces a copy of a book. The original file is called the **source** file; NFT does not affect the source file in any way.

The consumer

This computer is the destination for the copy. It consumes the copy that the producer sends, and stores it for later use. The file that NFT creates on the consumer is called the **target** file.

It is especially important that you understand the roles of the **producer** and the **consumer**, since the terms are used frequently. The ideas of a **source file** and **target file** are also common.

These drawings illustrate:

- The object labeled **PC** represents the PC workstation running NFT. It is always the initiator.

- The other objects represent other computers on the network.

- The arrows represent the transfer of a file from the producer to the consumer.

As you can see, there are four basic kinds of transfers:

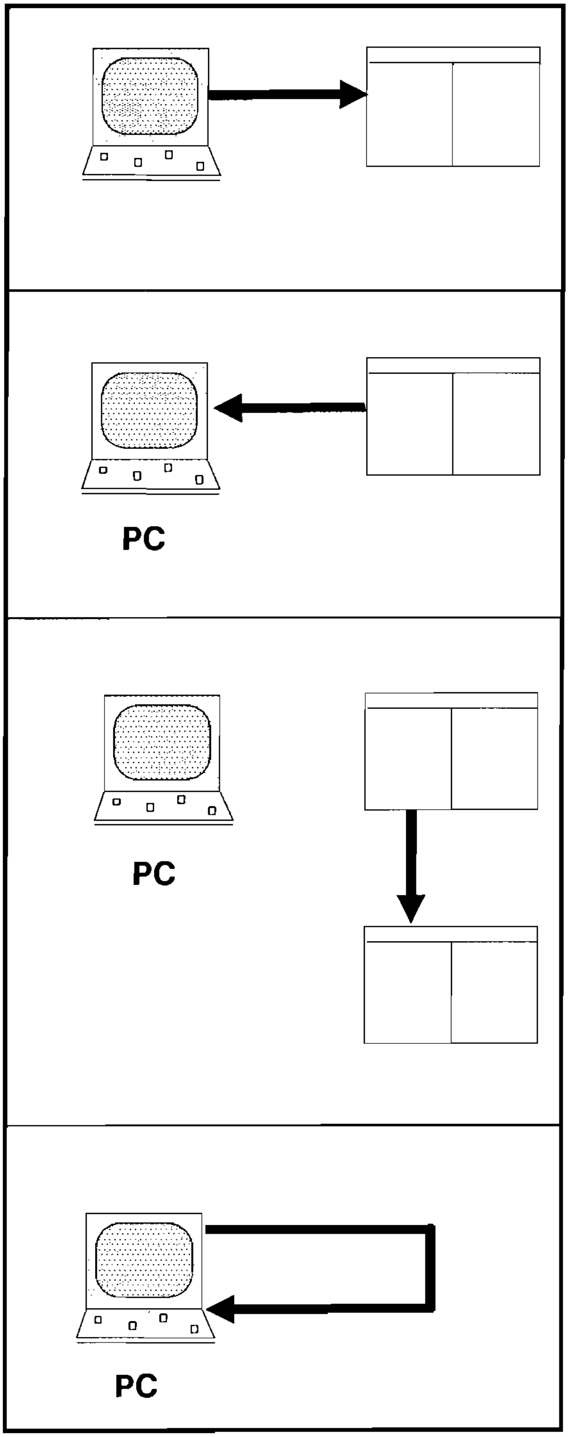
1. Local to remote

2. Remote to local

3. Remote to remote

4. Local to local

The term **NFT partner** refers in a general way to any computer with which you can exchange files using NFT.



Before You Begin

Before you use NFT, you have to start the network on your PC workstation, and have the NFT software ready to execute. If you try to use NFT without loading the LAN software, you will receive an error message.

Follow these steps:

1. Start the network

Your PC workstation may automatically start the LAN software when you boot it, or you may need to start the LAN software manually.

If you have any questions or if you need to start the LAN software manually, refer to the *User's Guide – HP OfficeShare Network.*

2. Have the NFT software available

Once the LAN software is loaded, the NFT software must be available and ready to execute. Follow the instructions for your PC workstation:

HARD DISC: If your PC workstation uses a hard disc:	**FLEXIBLE DISC: If your PC workstation uses flexible discs:**
You should ensure that the directory where the network is installed is on your directory search path (For details, see the description of the PATH command in your DOS reference manual). Then you can proceed without further preparation.	Insert the *User Configuration and NFT Work Disc* in drive A:. **NOTE:** Flexible disc users must specify the disc drive when using the DSCOPY command. For example, instead of typing: `dscopy fileA fileB` **Enter** you must type: `a:dscopy fileA fileB` **Enter**

3. Have your PC workstation ready to accept a DOS command.

If you use PAM: Go to the main screen. Once the main screen is visible, a flashing underscore appears in the upper left corner. You can then use the DSCOPY command.

If you use DOS: Get to the DOS prompt. Your PC workstation is then ready to accept the **DSCOPY** command.

The DSCOPY Command

You use a single command to use NFT – the **DSCOPY** command. The **DSCOPY** command copies a file from one computer to another. The **DSCOPY** command is similar to the **COPY** command in DOS.

To use NFT to transfer files, you type the command word – **DSCOPY** – and then identify the source file and the target file. Then you press the **Enter** key.

Syntax of the DSCOPY Command

The **DSCOPY** command takes this form:

```
dscopy [options] <source_file> <target_file> Enter
```

This command copies the information in the **source_file** to the **target_file.** Usually either the source file or the target file (or both) belong to a remote computer.

There are a number of options you can apply to a file transfers as necessary. The brackets around `options` indicates that you may or may not include NFT options in this position. The purpose and use of options is covered in the chapter **Using NFT Options**. When you use *no* options, NFT assumes the source and target files are ordinary text files, like those generated by EDLIN or another editor.

EXAMPLE

```
dscopy my_list.txt DataCentral#karen:open4me#partlist Enter
```

This example copies the local file called **my_list.txt** to a file called **partlist** that belongs to a user named **karen** on a computer called **DataCentral**. This example is explored in the rest of this section.

Note

If you run NFT from a flexible disc, use the command `a:dscopy`, **not** `dscopy`, to execute NFT.

Local Files

Specifying a PC (local) file as either the source or target file in a **DSCOPY** command is no different than specifying a PC file for any other purpose. If you have questions about how to specify a file, refer to your DOS reference manual. In the above example, **my_lis.txt**, is the local file. If your PC workstation is on a LAN with a PC or HP 3000 server, you can use shared discs just as you would use your local discs.

Note

Wildcard characters are not permitted in a **DSCOPY** file specification. You also cannot transfer entire directories.

Remote Files

It takes more information to identify a remote file. To specify a remote file (as either the source or target file in a **DSCOPY** command) you need:

- the name of the remote computer, or **node_name**
- authorization to use the remote computer, or **login**
- the name of the file, or **file_name**

These three parts are separated by pound-signs (#). So a complete identification for a remote file looks like this:

```
node_name#login#file_name
```

Review the example again:

```
dscopy  my_list.txt  DataCentral#karen:open4me#partlist  Enter
```

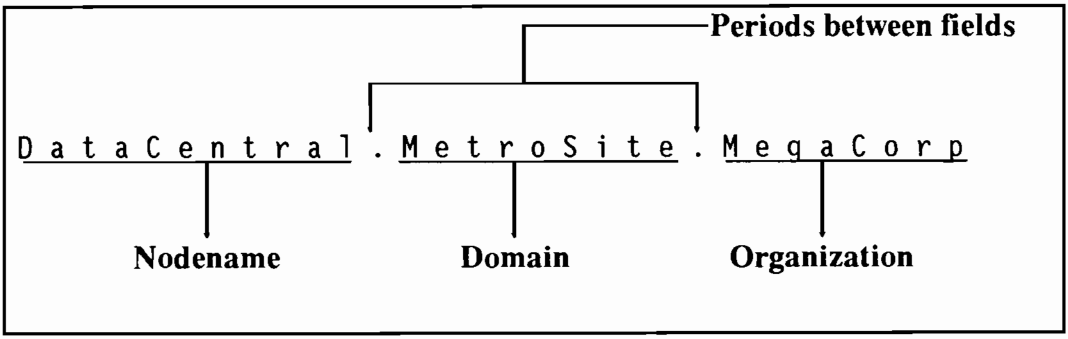
In this example, **DataCentral** is the name of the remote computer, **karen:open4me** authorizes the use of the remote computer, and **part_list** is the name of the file. The pound-signs (#) separate fields. The next paragraphs describe each of these fields.

The Node Name Field

Each computer on your network has a unique name, its **Node name**. This name is the first part of a remote file specification, and is always followed by a pound-sign (#).

Full node names have three fields, separated by periods:

- the **nodename**
- the **domain**
- the **organization**



Abbreviating the Node Name

Depending on how your network is configured, you may be able to abbreviate the Node Name. On your User Reference Worksheet compare the Node Name of your intended NFT partner with the Node Name of your PC workstation.

- If the organization fields match, you can use just the nodename and domain, with a period between.
- If the domain and organization fields both match, you can use just the nodename, with no periods.

Note

For clarity, most examples use an abbreviated node name. The abbreviated node name for the previous example is **DataCentral**.

The Login Field

A **login** is a way of ensuring that no unauthorized people can use a computer system. Each authorized user is given a work area on the computer, and is not normally granted permission to look at or change anything in other users' work areas. This provides security and privacy for each user.

A login for a computer of this kind usually consists of two parts:

- the user's identification
- the user's "password"

In our example for NFT, the login field is:

```
karen:open4me
```

The computer recognizes **karen** as someone who is authorized to use it. The password, **open4me**, ensures that an imposter cannot gain access. Some computers require more than one identification, or more than one password.

In this example, a colon (:) separates the user's identification from the password.

Note

The form of the login varies for different NFT partners. For example, the login for an HP 9000 computer uses a colon as shown above. When your partner is an HP 1000 computer, you use a slash (/) instead.

The exact form you need depends on your NFT partner. For details, see the chapter for your NFT partner.

To maintain security and privacy on computers with many users, NFT insists that you be authorized to use any computer that you exchange files with. If Karen wants co-workers to use NFT to exchange files from her work area, she must reveal her password to them.

Note

In many cases, the secrecy of passwords is very important. In such situations, passwords should *never* be visible on a computer screen. If you want the password to be invisible at all times, refer to the section **Security and Password Prompting** later in this chapter.

The File Name Field

A **file system** is a computer's overall scheme for naming and organizing files. You may be familiar with certain aspects of the DOS file system, like file names and extensions, and directories and subdirectories. For example, with the DOS file system, file names are limited to eight characters. You may also be familiar with the hierarchy of DOS directories, subdirectories, and files.

Different computer systems have different file systems. Each file system has its own rules for naming and organizing files.

Default Directory

One common feature is a "default" directory. The DOS concept of a current directory is similar to the default directory on a larger computer. When someone starts up a session with the computer (or "logs in"), the default directory is determined by the computer, according to the name of the user.

When you use NFT, the default directory on a remote computer is the default directory for the user whose login authorized the transfer. If you give *only* a file name in the **file_name** field, the remote computer will assume that you mean to use the default directory for the user. If necessary, you can explicitly state that the remote file exists in a different directory. See the chapter that covers your NFT partner for details on file names.

Note

You must have access rights to another directory before you can transfer files into or out of it. Otherwise, the transfer will fail and you will receive an error message. See the system administrator on the partner system for details.

EXAMPLES

The following examples show how NFT can be used to copy files between the PC workstation and other computers.

Note

These examples assume that the source and target files are ordinary text files containing ASCII data. See the section (in this chapter) **Common Transfers**, and the chapter **Using NFT Options** for information on how to accomplish transfers that do not meet this assumption.

```
dscopy  my_list.txt  DataCentral#karen:open4me#partlist  Enter
```

This is a copy of the last example to illustrate the syntax, or form, of a **DSCOPY** command. Note that the **local** file is in the active drive and directory on the PC workstation. (For details about active drives and directories, see your DOS documentation.) Also note that the **remote** file is in the home directory for the user, **karen.**

```
a:dscopy  my_list.txt  DataCentral#karen:open4me#partlist  Enter
```

This example is the same as the previous one, except that this PC workstation uses flexible discs instead of a hard disc. Notice that the command is `a:dscopy`, not `dscopy`. For simplicity, the other examples in this book assume that the PC workstation uses a hard disc.

```
dscopy  my_list.txt  DataCentral#karen:open4me#inventory/partlist  Enter
```

Note the more complex name for the **remote** file. This specifies that the copy belongs in a different directory than Karen's home directory. The exact meaning of this complex remote file name is not important now. Just remember that you specify a different directory as part of the **file_name** field.

```
dscopy  a:\parts\my_list.txt  DataCentral#karen:open4me#partlist  Enter
```

Note that the **local** file is in a specified drive and subdirectory on the PC workstation. This shows that you can transfer files from (or to) a drive and directory other than the current one.

```
dscopy  e:\parts\my_list.txt  DataCentral#karen:open4me#partlist  Enter
```

In this example, the **local** file is on a shared disc on a server for a PC LAN. On PC workstations with access to a server, you use the shared discs in the same way you use your local disc(s). (Your network may not have a server.)

```
dscopy  DataCentral#karen:open4me#sales  MfgCntrl#jim/6shooter#Q1sales  Enter
```

This example shows how to use NFT to transfer files between two remote computers. Here you can see that both the producer and consumer are remote computers. The PC workstation initiates a network file transfer of the file called **sales** from **DataCentral** to **MfgCntrl**, where it is called **Q1sales.**

```
dscopy  my_list.txt  DataCentral.MetroSite.MegaCorp#karen:open4me#partlist  Enter
```

This example uses an unabbreviated node name for the remote computer. Notice that all three fields are present, including Local Node Name, Environment Name, and Organization Name. A pound-sign follows the node name, as usual.

Common Transfers

NFT is a powerful and flexible file transfer tool. However, it is frequently used in two ways:

1. **Default transfers:**
   Moving text files, like memos and letters, from one computer to use on another. It is likely that someone will edit, print, or otherwise use the copy.

2. **Binary transfers:**
   Binary transfers are used for either of these cases:

   a. Remote storage, where a remote computer stores large or seldom used files. Typically no one uses a file while it is in remote storage; the remote computer is used as a "warehouse" for the file.

   b. Transferring binary data between a PC workstation and a remote computer.

There are many other uses for NFT, but these are common. In general:

- Use a **default** transfer whenever a file is readable and will be used on the remote computer.
- Use a **binary** transfer whenever you want use remote storage, or for moving data files between computers.

Note

There are important exceptions to the previous rules. For example, most people consider word processor files "readable." However, these programs sometimes insert important but invisible characters for their own purposes. These hidden characters could require you to use NFT options to obtain the kind of file you want on the target computer. If you have difficulty with a transfer, see the section **Common Difficulties** for more information.

EXAMPLE:

Below are examples of the two kinds of transfers:

Default Transfer:

```
   dscopy source1.fff DataCentral#karen:open4me#target1 Enter
```

Binary Transfer:

```
   dscopy -B source1.fff DataCentral#karen:open4me#target1 Enter
```

The difference is the **-B** that appears after the keyword **DSCOPY** in the binary transfer. The **-B** is an NFT **option**, namely the **Binary** option.

Note

There are many different options you can use with the **DSCOPY** command. See the chapter **Using NFT Options** for details about other options and their use.

Default and binary transfers are based on these two kinds of computer data:

- **ASCII** data
- **binary** data

ASCII Data

Default transfers assume that the source file contains ASCII (pronounced ask-ee) data. ASCII is short for the *American Standard Code for Information Interchange*. It is a computer code for:

- All the letters of the alphabet, both upper and lower case
- The digits 0 through 9
- All the punctuation marks
- A number of special purpose characters (control characters) that aren't visible on the screen and that do not print on paper

ASCII is a kind of "common language" that most computers share. Typical ASCII data includes letters, memos, electronic mail, and many other kinds of "readable" information.

Despite the common interpretation of ASCII data, different computers *store* ASCII data in different ways. When you use a default transfer, NFT automatically translates the file from one computer's organization to the other's. In some circumstances this automatic translation is inappropriate, and a file attribute error occurs. See **Common Difficulties** for more information.

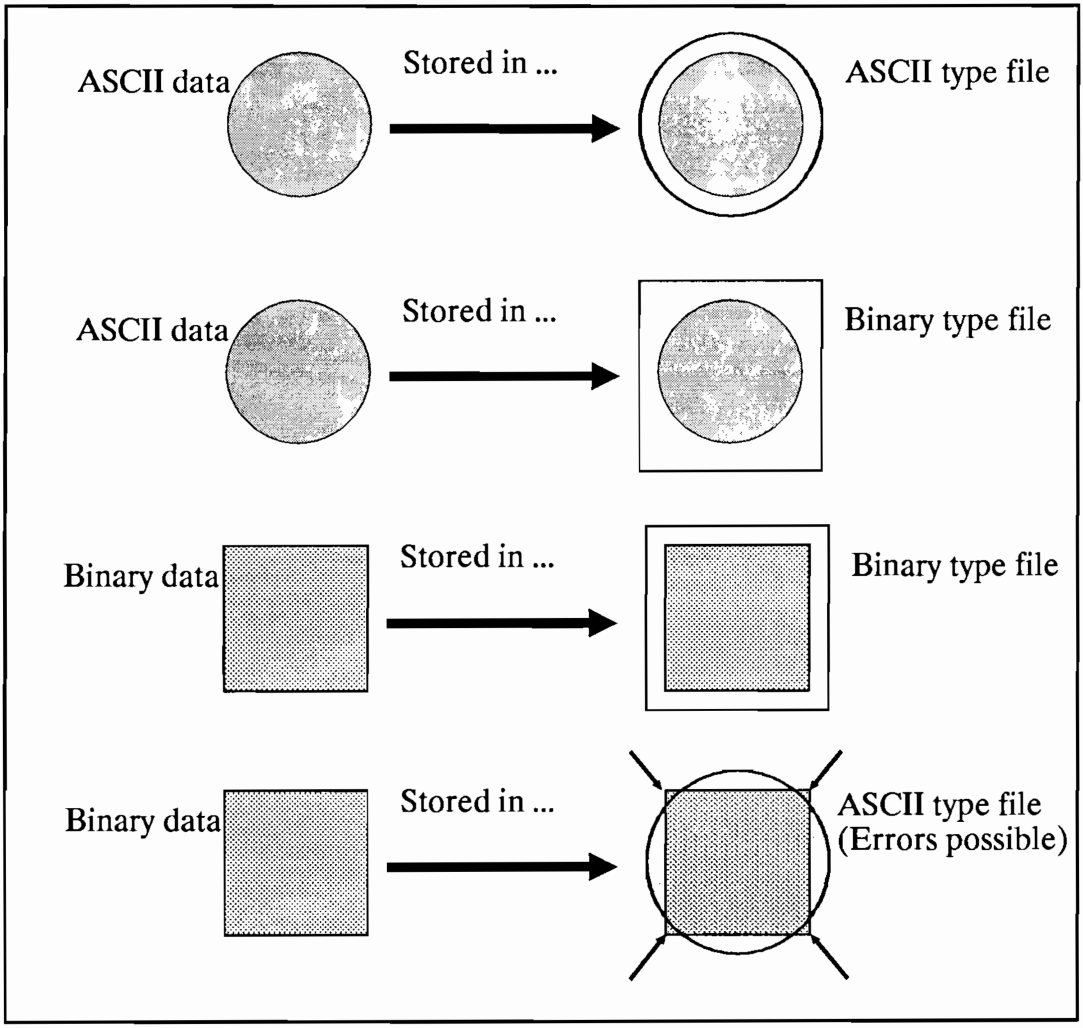
Binary Data

Binary transfers assume that the source file contains **binary data**. If a file contains information that is *not* coded in ASCII, we say it contains binary data. Among the many kinds of binary data are:

- Graphics files
- Executable files
- Data from remote sensing equipment
- Output files from a variety of software packages

Technically, binary data *includes* ASCII data. That is, ASCII data is also binary data, but the reverse is not true.

This distinction is important because some computers store binary and ASCII data in different kinds of files. If binary data is stored in a file meant for ASCII data, errors can occur. However, ASCII data can be safely stored in a file meant for binary data. You take advantage of this when you use NFT to store files on a remote system. These drawing illustrate:



Using Remote Storage

With NFT, you can use the large storage capacity of a large remote computer as a "warehouse" for your files. Files stored this way are backed up as regularly as the remote system's own files.

Use NFT's **binary** option, **-B**, to move the file *to* the remote computer for storage. Later, you again use the **-B** option to retrieve the file *from* the remote computer. The **-B** option appears on the command line between the word **DSCOPY** and the source file specification.

Storing a File Remotely

These two steps store a file on a remote computer:

1. Transfer the file from the PC workstation to the remote computer with the binary option:

   ```
   dscopy  -B  bigfile.dat  DataCentral#karen:open4me#savefile  Enter
   ```

   Caution

   Be sure you use the binary option in step 1. Also be sure that no errors are reported from step 1. Otherwise, you may lose data. Step 2 deletes the original file from the PC workstation.

   The file is now safely stored on the remote computer.

2. **Do not perform this step if errors were reported from step 1.** Delete (erase) the original file from your PC workstation. See your DOS documentation for instructions if you don't know how to do this.

Retrieving a File from Remote Storage

Just one step is necessary to recover a file that is in remote storage:

1. Transfer the file back to the PC workstation from the remote computer with the binary option:

   ```
   dscopy  -B  DataCentral#karen:open4me#savefile  bigfile.dat  Enter
   ```

   The file is again available on your PC workstation. The remote copy remains intact, so there is no problem if you forget to use the binary option on this transfer. Execute the command again using the binary option as shown.

Using an Embedded Pound-sign

The pound-sign separates fields in a remote file name. Therefore a login, password, or file name that contains a pound-sign (#) requires special treatment. In principle, you have to mark the pound-sign (with a backslash) so that NFT does not interpret it as a field separator.

To use a pound-sign (#) as part of a name, insert a backslash (\) before the pound-sign.

EXAMPLE

Ken's password is **a#ofcure**. He has a file called **dog#** to transfer to a PC workstation. The following **DSCOPY** command will work:

```
dscopy  DataCentral#ken:a\#ofcure#dog\#  my_file.txt  Enter
```

Note the backslashs preceding the embedded pound-signs.

Interrupting NFT

You can interrupt NFT during a file transfer. When NFT is interrupted, you can:

- Request a progress report, which tells you how much of the file has been transferred and then continues the transfer
- Abort a transfer in progress, and return to the DOS prompt or PAM

To interrupt NFT, you type **CTRL-C** during a file transfer. NFT responds as follows:

```
Abort or Progress ? (A or P):
```

EXAMPLE

To request a status report, at the above prompt type:

`p` **Enter**

NFT reports the progress of the file transfer:

```
28% of current file has been transferred.
```

To abort a transfer in progress, at the above prompt type:

`a` **Enter**

Security and Password Prompting

NFT offers a way to keep passwords invisible during file transfers. Without this feature, passwords that should remain confidential are visible on the screen of the PC workstation when the **DSCOPY** command is issued.

Because an unauthorized person could read a password over your shoulder or while the PC workstation is unattended, you may want to use the password prompting feature of NFT to maintain security.

When you use password prompting, you substitute a **key** in place of each password in the **DSCOPY** command. Then, before NFT copies the file, it requests that you supply the password(s). You respond by typing the password and pressing **Enter**. However, the characters you type do not appear on the screen. NFT recognizes the password you enter, and proceeds accordingly.

Using Password Prompting

Use the rules below to maintain password security. The **key** is either one colon (:) or two (::), according the these rules:.

1. Use a key in place of each password that you want secure.

2. If the remote node is either an HP-UX or DEC VAX system, the key is a single colon.

3. If the remote node is not one of the above, the key is a double colon.

EXAMPLES

In the first example, the remote node is an HP-UX system, so the key is a single colon. When you enter this command, NFT responds on the next line as shown below:

```
dscopy  my_file.txt  DataCentral#karen:#partlist  Enter

target password>
```

At this point, you type the password, and press **Enter**. The characters you type do not appear on your screen. Once NFT knows the password, the transfer proceeds normally.

In the second example, the remote node is an HP 3000 system. The HP 3000 requires two passwords. Because we haven't used an HP 3000 in any previous example, let's first see what a **DSCOPY** command with an HP 3000 partner looks like *without* using password prompting:

```
dscopy  admin#ken/nolimit.sales/sell2day#partlist  my_file.txt  Enter
```

In this example:

- Ken's personal password is **nolimit**.
- Every file in the "sales" group has the group password, **sell2day**.

The exact syntax and meaning of each part of the HP 3000 login field is of no concern now. That information is covered in the chapter for the HP 3000. For now, just compare the way the passwords in the above command relate to the password prompting keys in the following command. By the third rule above, the keys are pairs of colons. Again, NFT responds on the next line as shown below:

```
dscopy  administration#ken/::.sales/::#partlist  my_file.txt  Enter
```

`source password 1>` (Here you type `nolimit`, and press **Enter**.)

`source password 2>` (Type `sell2day`, and press **Enter**.)

File Lockwords

File lockword prompting is a related idea. Some computers (e.g., the HP 3000) allow files to be password, or **lockword**, protected. The exact syntax for such a file appears in the section for the HP 3000 partner. If you need to use lockword prompting for a file lockword, use the **-P** option after the **DSCOPY** command, and include the double colon key in place of the file lockword. NFT will prompt you to enter the file lockword.

For full details on the **-P** option, see the chapter **Using NFT Options.**

Common Difficulties

When you use NFT, you can encounter three types of errors:

- Syntax errors
- DSCOPY errors
- File attribute errors

Syntax Errors

What is a syntax error?	When do they occur?
A syntax error is a mistake in typing that leaves NFT unable to execute the DSCOPY command.	A syntax error is detected before NFT even attempts to execute a file transfer.

What to do

When NFT encounters a syntax error it issues an error message and quits. You can then correct the problem and continue by re-entering the **DSCOPY** command. Syntax errors are the easiest to correct.

EXAMPLES

All the following commands cause syntax errors:

```
dscopy  my_list.txt  Enter
```

Missing target file.

```
dscopy  -w  my_list.txt  DataCentral#Karen:open4me#partlist  Enter
```

The **-w** (after the keyword **DSCOPY**) is not a valid option. (See **Using NFT Options.**)

```
dscopy  my_list.txt  DataCentral#Karen;open4me#partlist  Enter
```

Used a semicolon between the user identification and password. This is not correct syntax for any NFT partner.

```
dscopy  -VB  my_list.txt  DataCentral#Karen:open4me#partlist  Enter
```

The two options, **-V** and **-B**, were concatenated. Concatenating options is illegal. (See **Using NFT Options.**)

In each case, you get an error message and can retype the command using the correct syntax.

DSCOPY Errors

What is a DSCOPY error?	When do they occur?
A DSCOPY error is a condition that may cause a transfer in progress to fail.	DSCOPY errors are detected after NFT has begun to execute, but before the transfer is finished. You always receive an error message that describes the problem.

What to do

See **Appendix A** for a complete list of DSCOPY errors, including explanations and appropriate ways to fix or avoid each problem.

EXAMPLES

Any of the following conditions cause DSCOPY errors:

- The partner system is not currently available.
- The source file and target file are identical.
- The remote computer experiences system failure during a transfer.
- The remote computer rejects the transfer because of an incorrect password.

There are many other conditions that can cause a **DSCOPY** command to fail. Refer to **Appendix A** for details.

File Attribute Errors

What is a File Attribute error?

NFT works between computers with sometimes very different file systems. It must perform a kind of translation from one computer's file system to the other's.

File attribute errors happen when this automatic translation is incorrect. The DSCOPY command finishes with apparent success, but the target file cannot be used for its intended purpose.

When do they occur?

File attribute errors are not detected by NFT at all. Rather, they become apparent when someone encounters difficulties with the target file. There are two potential causes for the incorrect translation between file systems during a transfer:

1. NFT made an incorrect assumption about the transfer, **or**

2. You applied one or more NFT options incorrectly.

Both causes may be present at once.

What to do

If you transfer a file and later discover that the target file does not work for its intended purpose, you can create a target file with the proper attributes by using NFT options correctly. See the chapter **Using NFT Options** for details. Detecting, analyzing, and correcting a file attribute error may require considerable knowledge of the different file systems, the attributes of the source file, and the intended purpose of the target file.

EXAMPLE

The presence a file attribute error strictly depends on the intended purpose of the target file. A simple example of this can be described with a transfer between a PC workstation and an HP 9000 partner: Karen has been editing a text file on the HP 9000 called **draft1**. She wants to transfer this file to her PC workstation to do some further editing. She executes the following **DSCOPY** command:

```
dscopy -B DataCentral#karen:open4me#draft1 draft2.txt Enter
```

Here, Karen has incorrectly used the binary option, **-B.** As a result, she will have difficulty trying to edit the file **draft2.txt.** The source file on the HP 9000 computer is a **text** file, and Karen intends to edit the copy as a **text** file on her PC workstation. She needs to perform a **text**, or default transfer. To get the file she wants, Karen should execute the command again *without* the binary option.

Other file attribute errors can be more subtle. If you transfer a file between computers and discover that the target file is not acceptable for its intended purpose, a file attribute error is a possible problem. See the chapter **Using NFT Options** for details on options you can use to correct the situation.

Notes

3

NFT with the HP 1000 Computer

Note

You must understand how to use the **DSCOPY** command before you can use the information in this chapter. See the chapter **Using NFT** if you have questions.

Quick Reference

Logins for HP 1000 computers	**Syntax:**	username/password
	Example:	bob/keylock
File Names for HP 1000 computers	**Syntax:**	[/][directory_chain/]filename
	Example:	bob/products/kits/toyboat

To use password prompting, include a slash before the double colon key that replaces the password.

The square brackets in the file name syntax indicate optional fields. If you omit the initial slash, the directory chain originates from the user's home directory. If you include the initial slash, the directory chain begins at a root directory. A slash separates directories on the directory chain.

Review

Recall the basic form of a **DSCOPY** command:

```
dscopy [options] <source_file> <target_file> Enter
```

Either the **source_file** or the **target_file** (or both) belongs to a remote computer. In this chapter, we work with a remote HP 1000.

Local Files

You specify the PC file the same way you specify any other file on the PC workstation. See your DOS documentation for details.

Remote Files

A remote file name has three parts:

- node_name
- login
- file_name

These fields are separated by pound-signs (#):

```
node_name#login#file_name
```

See **Using NFT** for details.

A Typical DSCOPY Command

This **DSCOPY** command transfers a local file to a remote system:

```
dscopy local.fil remote_node#login#remote_file Enter
```

About This Chapter

Two fields in the remote file specification change depending on the kind of remote computer involved:

- the **login**
- the **file_name**

This chapter shows you how to specify the **login** and **file_name** when the remote computer is an HP 1000. It also has examples of transfers with an HP 1000.

Note

For simplicity, many DSCOPY examples in this chapter use an abbreviation. These examples use dots to fill fields that are not discussed in the example. For instance, if **remote file names** are the current topic, the example looks like this:

```
     dscopy  ...  ...#...#remote_file
```

The **node_name**, **login**, and **local_file** fields are occupied by dots.

Logins on the HP 1000 Computer

Logins on HP 1000 computers take this form:

```
username/password
```

The **username** is the identification of an authorized user. If the user has no password, omit the slash (/) and password. If you use password prompting, be sure to include the slash (/) before the double-colon key.

EXAMPLES

The following **DSCOPY** commands show legal login fields that could be used with a remote HP 1000:

```
dscopy ...  ...#bob/keylock#...  Enter
```

This is a typical example of a login field for an HP 1000 partner.

```
dscopy ...  ...#bob/::#...  Enter
```

In the second example, **bob** is using password prompting for security, so the double colon key appears after the slash (/).

```
dscopy ...  ...#guest#...  Enter
```

In the third example, **guest** has no password, so the slash and password are omitted.

Specifying Files on the HP 1000 Computer

The way the HP 1000 organizes files is similar to the organization DOS uses. That is, files are catalogued in directories. Directories can also contain subdirectories, which can catalog files and deeper subdirectories.

Default Directory

Each user of an HP 1000 has a "home" or default working directory. The home directory defines the user's personal work area on the HP 1000. All subdirectories of the home directory are also part of the work area. Files in the home directory are the easiest to specify: just type the name in the **file_name** field.

EXAMPLE

```
dscopy ...  ...#...#suppliers  Enter
```

User's Subdirectories

To specify a file in a subdirectory under the user's home directory, type the name of the subdirectory, then a slash, then the file name. If the file is in a deeper subdirectory, type the subdirectory names in order, beginning from the home directory. Separate them with a slash, and use a slash before the file name at the end.

EXAMPLES

```
dscopy  ...  ...#...#products/volleyballs  Enter
```

The file **volleyballs** is catalogued in a subdirectory, called **products**, of the home directory.

```
dscopy  ...  ...#...#products/kits/toyboat  Enter
```

The file **toyboat** is catalogued in a directory called **kits**. In turn, **kits** is a subdirectory of **products**, which itself is a subdirectory of the user's home directory.

Other Directories on the HP 1000 Computer

Directories outside the user's work area can be specified by starting out with a slash. Then the HP 1000 starts at a root directory for the system (one that is not a subdirectory of *any* directory), and follows the chain of directory names to the subdirectory where the file is found.

EXAMPLE

```
dscopy  ...  ...#...#/marketing/karen/products/kits/toyboat  Enter
```

In this example, the HP 1000 starts at the **marketing** root directory, and follows the chain of subdirectories from **marketing** to **karen** to **products** to **kits**, and finally finds the file **toyboat**.

More Examples

The following examples show whole **DSCOPY** commands for transfers to an HP 1000:

1. `dscopy  my_list.txt  DataCentral#bob/keylock#partlist` **Enter**

2. `dscopy  my_list.txt  DataCentral#bob/::#partlist` **Enter**

   The first and second examples are identical, except that the second uses password prompting to maintain password security. Note the double-colon key in place of the password in the second example.

3. `dscopy  DataCentral#bob/keylock#stock/panel  panel.txt` **Enter**

4. `dscopy  DataCentral#bob/keylock#stock/bulk/wire  wire.txt` **Enter**

   The third and fourth examples show how to specify remote files that exist in the user's subdirectories

5. `dscopy  DataCentral#bob/::#/users/jim/legalnote  law.txt` **Enter**

   The fifth example shows how to specify a remote file outside the user's work area. Note the initial slash (/) in the remote file name.

6. `dscopy  -B  graph1.pcx  DataCentral#bob/keylock#graph1` **Enter**

7. `dscopy  -B  DataCentral#bob/keylock#graph1  graph1.pcx` **Enter**

   The sixth example shows how to use remote storage on a remote HP 1000. The seventh example shows how to later retrieve the file that was stored in example 6. Note the consistent use of the binary, **-B**, option.

4

NFT with the HP 3000 Computer

Note

You must understand how to use the **DSCOPY** command before you can use the information in this chapter. See the chapter **Using NFT** if you have questions.

Quick Reference

Logins for HP 3000 computers	**Syntax:**	user[/usrpass].account[/acctpass][,group[/grppass]]
	Example:	bob/keylock.sales/sell2day
File Names for HP 3000 computers	**Syntax:**	filename[/lockword][.group[.account]]
	Example:	clients/private.legal.admin

The square brackets indicate optional fields.

To use password prompting, include a slash before the double colon key(s) that replaces the password(s).

To use file lockword prompting, include a slash before the double colon key that replaces the file lockword.

Review

Recall the basic form of a **DSCOPY** command:

```
dscopy [options] <source_file> <target_file> Enter
```

Either the **source_file** or the **target_file** (or both) belongs to a remote computer. In this chapter, we work with a remote HP 3000.

Local Files

You specify the PC file the same way you specify any other file on the PC workstation. See your DOS documentation for details.

Remote Files

A remote file name has three parts:

- node_name
- login
- file_name

These fields are separated by pound-signs (#):

```
node_name#login#file_name
```

See **Using NFT** for details.

A Typical DSCOPY Command

This **DSCOPY** command transfers a local file to a remote system:

```
dscopy local.fil remote_node#login#remote_file Enter
```

About This Chapter

Two fields in the remote file specification change depending on the remote computer involved:

- the **login**
- the **file_name**

This chapter shows you how to specify the **login** and **file_name** when the remote computer is an HP 3000. It also has examples of transfers with an HP 3000.

Note

For simplicity, many DSCOPY examples in this chapter use an abbreviation. These examples use dots to fill fields that are not discussed in the example. For instance, if **remote file names** are the current topic, the example looks like this:

```
     dscopy  ...  ...#...#remote_file
```

The **node_name**, **login**, and **local_file** fields are occupied by dots.

Logins on the HP 3000 Computer

Logins on HP 3000 computers take this form:

```
user[/usrpass].account[/acctpass][,group[/grppass]]
```

The **user** is the identification of an authorized user. The **account** is the name of the account the user belongs to. The **group** is the name of the group within the account. Passwords may be associated with any of these fields. They are enclosed in brackets to indicate they are optional. If you have to use a password, precede it with a slash (/). If you use password prompting, include the double-colon key after the slash.

EXAMPLES

The following **DSCOPY** commands show legal login fields that could be used with a remote HP 3000:

```
dscopy  ...  ...#ken/nolimit.sales/sell2day#...  Enter
```

This is a typical example of a login field for an HP 3000 partner.

```
dscopy  ...  ...#ken/::.sales/::#...  Enter
```

In the second example, **ken** is using password prompting for security, so the colon key appears in place of each password.

```
dscopy  ...  ...#ken/nolimit.sales/sell2day,retail/1buy1#...  Enter
```

The third example is like the first, but the group, **retail**, and its password, **1buy1**, are specified.

Specifying Files on the HP 3000 Computer

Default Group

Each account on an HP 3000 has one or more groups assigned to it. The default group for a user is the group he or she accesses after logging on. Files in the default group are the easiest to specify: just type the name in the **file_name** field.

EXAMPLE

```
dscopy  ...  ...#...#pricelist Enter
```

Other Groups on the HP 3000

Groups outside the user's work area can be specified by following the file name with the a period, then the group name. A particular account can be specified by following the group name with a period and then an account name.

EXAMPLES

```
dscopy  ...  ...#...#pricelist.manufact  Enter
```

In this example, the HP 3000 places the file in the main directory for the **manufact** group.

```
dscopy  ...  ...#...#pricelist.manufact.shop1  Enter
```

In this example, the HP 3000 places the file in the directory for the **shop1** account under the **manufact** group.

File Lockwords

On an HP 3000, files can be individually password protected. These file passwords are called "lockwords". To access a file that is lockword protected, follow the file name with a slash (/), then the lockword.

EXAMPLE

```
dscopy  ...  ...#...#pricelist/secret  Enter
```

In this example, the file **pricelist** can only be accessed by people who know the password, **secret**. This transfer creates a file that is secure from tampering by others who have access to this directory.

File Lockword Prompting

You can maintain the security of file passwords by using **lockword prompting**, which is like password prompting described earlier. As usual, you substitute a double-colon (::) key in place of the file password, or lockword. In addition, you **must** use the file lockword option, **-P**, before the source file specification.

EXAMPLE

```
dscopy  -P  ...  ...#...#pricelist/::  Enter
```

Note the file lockword option, **-P**, and the double-colon key in place of the file lockword.

More Examples

These examples show **DSCOPY** commands with an HP 3000 partner:

1. `dscopy my_list.txt admin#ken/nolimit.sales/sell2day#memo` **Enter**

2. `dscopy my_list.txt admin#ken/::.sales/::#memo` **Enter**

   The first and second examples are identical, except that the second uses password prompting to maintain password security. Note the double-colon key (::) in place of each password in the second example.

3. `dscopy admin#ken/nolimit.sales/sell2day#wire.stock wire.txt` **Enter**

4. `dscopy admin#ken/nolimit.sales/sell2day#wire.stock.mfg wire.txt` **Enter**

   The third and fourth examples show how to specify remote files that exist outside the user's default directory.

5. `dscopy admin#ken/nolimit.sales/sell2day#clients/private names.txt` **Enter**

6. `dscopy -P admin#ken/nolimit.sales/sell2day#clients/:: names.txt` **Enter**

   The fifth example shows how to access a remote file that is lockword protected. The sixth example shows how to use lockword prompting on a lockword protected file. Note the file lockword option, **-P**, and the double-colon key (::) in place of the file lockword.

7. `dscopy -B graph1.pcx admin#ken/nolimit.sales/sell2day#graph1` **Enter**

8. `dscopy -B admin#ken/nolimit.sales.mktg/sell2day#graph1 graph1.pcx` **Enter**

   The seventh example shows how to use remote storage on a remote HP 3000. The eighth example shows how to later retrieve the file that was stored in example 7. Note the consistent use of the binary, **-B**, option to both store and retrieve the file.

5

NFT with the HP 9000 Computer

Note

You must understand how to use the **DSCOPY** command before you can use the information in this chapter. See the chapter **Using NFT** if you have questions.

Quick Reference

Logins for HP 9000 computers	**Syntax:**	username:userpass
	Example:	bob:keylock
File Names for HP 9000 computers	**Syntax:**	[/][directory_chain/]filename
	Example:	/users/bob/products/kits/toyboat

To use password prompting, include the single-colon key after the user's name (i.e., omit the password).

The square brackets in the file name syntax indicate optional fields. If you omit the initial slash, the directory chain originates from the user's default directory. If you include the initial slash, the directory chain begins at the root directory for the system. The directory chain must end with a slash.

Review

Recall the basic form of a **DSCOPY** command:

```
dscopy [options] <source_file> <target_file> Enter
```

Either the **source_file** or the **target_file** (or both) belongs to a remote computer. In this chapter, we work with a remote HP 9000.

Local Files

You specify the PC file the same way you specify any other file on the PC workstation. See your DOS documentation for details.

Remote Files

A remote file name has three parts:

- node_name
- login
- file_name

These fields are separated by pound-signs (#):

```
node_name#login#file_name
```

See the chapter **Using NFT** for details.

A Typical DSCOPY Command

This **DSCOPY** command transfers a local file to a remote system:

```
dscopy local.fil remote_node#login#remote_file Enter
```

About This Chapter

Two fields in the remote file specification change depending on the remote computer involved:

- the **login**
- the **file_name**

This chapter shows you how to specify the **login** and **file_name** when the remote computer is an HP 9000. It also has examples of transfers with an HP 9000.

Note

For simplicity, many DSCOPY examples in this chapter use an abbreviation. These examples use dots to fill fields that are not discussed in the example. For instance, if **remote file names** are the current topic, the example looks like this:

```
    dscopy  ...  ...#...#remote_file
```

The **node_name**, **login**, and **local_file** fields are occupied by dots.

Logins on the HP 9000 Computer

Logins on HP 9000 computers take the form:

```
username:password
```

The **username** is the identification of an authorized user. If the user has no password, omit the colon (:) and password. If you use password prompting, be sure to include the colon key.

EXAMPLES

The following **DSCOPY** commands show legal login fields that could be used with a remote HP 9000:

```
dscopy  ...  ...#karen:open4me#...
```

This is a typical login field for an HP 9000 partner.

```
dscopy  ...  ...#karen:#...
```

In the second example, **karen** is using password prompting for security, so the colon key appears without the password.

```
dscopy  ...  ...#guest#...
```

In the third example, **guest** has no password, so the colon and password are omitted.

Specifying Files on the HP 9000 Computer

The way the HP 9000 organizes files is similar to the organization DOS uses. That is, files are catalogued in directories. Directories can also contain subdirectories, which can catalog files and deeper subdirectories.

Default Directory

Each user of an HP 9000 has a "home" or default working directory. The default directory defines the user's personal work area on the HP 9000. All subdirectories of the default directory are also part of the work area. Files in the default directory are the easiest to specify: just type the name in the **file_name** field.

EXAMPLE

```
dscopy  ...  ...#...#partlist  Enter
```

User's Subdirectories

To specify a file in a subdirectory under the user's home directory, type the name of the subdirectory, then a slash, then the file name. If the file is in a deeper subdirectory, type the subdirectory names in order, beginning from the home directory. Separate them with a slash, and use a slash before the file name at the end.

EXAMPLES

```
dscopy ...  ...#...#products/volleyballs  Enter
```

The file **volleyballs** is catalogued in a subdirectory, called **products**, of the home directory.

```
dscopy ...  ...#...#products/kits/toyboat  Enter
```

The file **toyboat** is catalogued in a directory called **kits.** In turn, **kits** is a subdirectory of **products**, which itself is a subdirectory of the user's home directory.

Other Directories on the HP 9000 Computer

Directories outside the user's work area can be specified by starting out with a slash. Then the HP 9000 starts at the highest directory in the system (the one that is not a subdirectory for *any* directory), and follows the chain of directory names to the subdirectory where the file is found.

EXAMPLE

```
dscopy ...  ...#...#/users/marketing/karen/products/kits/toyboat  Enter
```

In this example, the HP 9000 starts at the top of the hierarchy of directories, and follows the chain of subdirectories from **users** to **marketing** to **karen** to **products** to **kits**, and finally finds the file **toyboat.**

More Examples

The following examples show whole **DSCOPY** commands for transfers to an HP 9000:

1. `dscopy  my_list.txt  DataCentral#karen:open4me#partlist` **Enter**

2. `dscopy  my_list.txt  DataCentral#karen:#partlist` **Enter**

   The first and second examples are identical, except that the second uses password prompting to maintain password security. Note the colon key in place of the password in the second example.

3. `dscopy  DataCentral#karen:open4me#stock/panel  panel.txt` **Enter**

4. `dscopy  DataCentral#karen:open4me#stock/bulk/wire  wire.txt` **Enter**

   The third and fourth examples show how to specify remote files that exist in the user's subdirectories

5. `dscopy  DataCentral#karen:#/users/jim/legalnote  law.txt` **Enter**

   The fifth example shows how to specify a remote file outside the user's work area. Note the initial slash (/) in the remote file name.

6. `dscopy  -B  graph1.pcx  DataCentral#karen:open4me#graph1` **Enter**

7. `dscopy  -B  DataCentral#karen:open4me#graph1  graph1.pcx` **Enter**

   The sixth example shows how to remote storage on a remote HP 9000. The seventh example shows how to later retrieve the file that was stored in example 6. Note the consistent use of the binary, **-B**, option.

6

NFT with the DEC VAX Computer

Note

You must understand how to use the **DSCOPY** command before you can use the information in this chapter. See the chapter **Using NFT** if you have questions.

Quick Reference

Logins for DEC VAX computers	**Syntax:**	username:userpass
	Example:	karen:open4me
File Names DEC VAX computers	**Syntax:**	device:[directory_chain]filename.type;version
	Example:	sys$admin:[karen.data.sales]thirdquarter.dat;3

To use password prompting, include the colon key after the user's name (i.e., omit the password).

The **device** is a device name; the **directory_chain** is a path on the device. Both are optional. Periods always separate names on the directory_chain. Either the **file_name** or the **type** must be present; usually both are present. The period between them is mandatory. The **version** is an integer. If you omit the semicolon and version, the highest numbered version of the file is used.

Review

Recall the basic form of a **DSCOPY** command:

```
dscopy  [options]  <source_file>  <target_file>  Enter
```

Either the **source_file** or the **target_file** (or both) belongs to a remote computer. In this chapter, we work with a remote DEC VAX computer.

Local Files

You specify the PC file the same way you specify any other file on the PC workstation. See your DOS documentation for details.

Remote Files

A remote file name has three parts:

- node_name
- login
- file_name

These fields are separated by pound-signs (#):

```
node_name#login#file_name
```

See the chapter **Using NFT** for details.

A Typical DSCOPY Command

This **DSCOPY** command transfers a local file to a remote system:

```
dscopy  local.fil  remote_node#login#remote_file  Enter
```

About This Chapter

Two fields in the remote file specification change depending on the kind of remote computer involved:

- the **login**
- the **file_name**

This chapter shows you how to specify the **login** and **file_name** when the remote computer is a DEC VAX computer. It also has examples of transfers with a DEC VAX computer.

Note

For simplicity, many DSCOPY examples in this chapter use an abbreviation. These examples use dots to fill fields that are not discussed in the example. For instance, if **remote file names** are the current topic, the example looks like this:

```
     dscopy  ...  ...#...#remote_file
```

The **node_name**, **login**, and **local_file** fields are occupied by dots.

Logins on the DEC VAX Computer

Logins on DEC VAX computers take this form:

```
username:password
```

The **username** is the identification of an authorized user. If the user has no password, omit the colon (:) and password. If you use password prompting, be sure to include the colon key.

EXAMPLES

The following **DSCOPY** commands show legal login fields that could be used with a remote DEC VAX computer:

```
dscopy  ...  ...#karen:open4me#...
```

This is a typical login field for a DEC VAX computer.

```
dscopy  ...  ...#karen:#...
```

In the second example, **karen** is using password prompting for security, so the colon key appears in place of the password.

```
dscopy  ...  ...#guest#...
```

In the third example, **guest** has no password, so the colon and password are omitted.

Specifying Files on the DEC VAX Computer

The way the DEC VAX computer organizes files is similar to the organization DOS uses, though the syntax is somewhat different. There are **devices** (similar to disc drives on the PC workstation), **directories**, and **files**. Files and directories are ways of organizing storage space on devices. Files are cataloged in directories. Directories can also catalog subdirectories, which can catalog files and deeper subdirectories.

Like DOS, file names on DEC VAX computers may include a period followed by a file extension. In addition, every file on a VAX computer has a version number. The version number is increased by one every time the file is modified, and all the old versions are saved (until explicitly removed).

EXAMPLE

This example illustrates a simple file name on a DEC VAX computer:

```
TOYBOAT.NAV
```

However, as the following sections explain, you may need to use a full file name. For example, the unabbreviated name of this file could be:

```
SYS$MANUFAC:[KAREN.PRODUCTS.KITS]TOYBOAT.NAV;17
```

About LOGIN.COM and LOGOUT.COM

When a VAX system acts as the NFT producer or consumer, it runs the LOGIN.COM command procedure for the user named in the login field of the file specification.

Thus, any logical names defined by the LOGIN.COM are also defined for NFT. These logical names can be used in file specifications.

Also, if the LOGIN.COM command procedure performs a "set default" command to change the default directory, file specifications for that user should be relative to that directory.

If a LOGIN.COM file executes a time-consuming operation or does something inappropriate for a non-interactive process, you should insert the following in the LOGIN.COM file:

```
  IF  F$MODE()  .NES.  "INTERACTIVE"  THEN  EXIT
```

This statement causes the LOGIN.COM procedure to exit immediately for a non-interactive process. Place this statement after any logical name definitions and set default commands you want active for NFT.

If the LOGOUT command has been redefined, ensure that a non-interactive process will not be adversely affected. To avoid potential problems, insert the above statement in the LOGOUT.COM command procedure as well.

Default Directory

Each user of a DEC VAX computer has a "home" or default working directory on a default device. The default directory defines the user's personal work area on the DEC VAX computer. All subdirectories of the home directory are also part of the work area. Files in the default directory are the easiest to specify: just type the name in the file_name field. If you want to specify a particular version of a file, attach a semicolon (;) and the version number to the file name.

EXAMPLE

```
dscopy  ...#...#partlist.edt  ...
```

The first example takes the file with the specified name and the *highest* version number available.

```
dscopy  ...#...#partlist.edt;7  ...
```

The second example explicitly requests version 7 of the file.

User's Subdirectories

To specify a file in a subdirectory under the user's home directory, type a left square bracket ([), a period, then the name of the subdirectory. Finally, type a right square bracket (]), then the file name.

If the file is in a deeper subdirectory, type the subdirectory names in order, beginning from the home directory. Separate them with a period, and use a right square bracket (]) before the file name at the end.

EXAMPLE

```
dscopy  ...#...#[.products]volleyballs.edt  ...
```

In the first example, the file **volleyballs.edt** is in a directory, called **products**, that is a subdirectory of the user's home directory. Note the period before the subdirectory name; this specifies that it is a subdirectory of the current (usually *home*) directory.

```
dscopy  ...#...#[.products.kits]modelboat.edt;8  ...
```

In the second example, version 8 of the file **modelboat.edt** is in a directory called **kits**, which is a subdirectory of **products**. In turn, **products** is a subdirectory of the user's home directory. Again, note the initial period in the directory chain. The initial period makes the directory chain relative to the current directory.

Other Directories on the DEC VAX Computer

Directories outside the user's work area may share the same device as the user, or a different device.

Specifying Files on the User's Device

This is similar to specifying user's subdirectories. Type a left square bracket ([), then *without a leading period*, type the first directory name. Chain together subdirectory names with periods. Finally, type a right square bracket (]), and then the file name.

EXAMPLE

```
dscopy  ...#...#[dave.memos]policy.edt;10  ...
```

In the first example, version 10 of the file **policy.edt** is in a directory, called **memos**, which is a subdirectory of the default directory for a user called **dave**. Because there is no initial period in the directory chain, the VAX computer expects to find a directory for **dave**.

```
dscopy  ...#...#[dave.memos.legal]audit.edt  ...
```

In the second example, the file **audit.edt** is in a directory called **legal,** which is a subdirectory of the directory **memos**. In turn, **memos** is a subdirectory of the home directory for a user called **dave**. Again, there is no initial period in the directory chain.

Specifying Files on a Different Device

To specify files on a different device, type the name of the device, then a colon, and then proceed as described in the previous section.

EXAMPLES

```
dscopy  ...#...#sys$mfg:[bob.data]inventory.edt;1  ...
```

In the first example, version 1 of the file **inventory.edt** is in a directory, called **data**, which is a subdirectory of the home directory for a user called **bob**. The home directory for this user, **bob**, is on the device called **sys$mfg**. Note the colon that follows the device name.

```
dscopy  ...#...#sys$admin:[kim.payroll.timecards]policy.edt  ...
```

In the second example, the file **policy.edt** is in a directory called **timecards** , which is a subdirectory of the directory **payroll.** In turn, **payroll** is a subdirectory of the home directory for a user called **kim.** The home directory for this user, **kim,** is on a device called **sys$admin.**

Summary: File Name Syntax for the DEC VAX Computer

Under the VAX/VMS operating system, files exist in a hierarchy of devices, directories, and subdirectories. The device is the physical unit (e.g., the disk) where the file exists. Files can be catalogued in any directory or subdirectory on the device. A subdirectory is a directory that is cataloged in the next higher level directory or subdirectory.

The general syntax for a VAX file specification is:

```
device:[directory_chain]file_name.type;version
```

- The **device** is a device name, which if omitted (along with the colon) defaults to the current (usually logon) device.
- The **directory_chain** can be an absolute path on the device, or relative to the current (usually home) directory on the device. Relative paths begin with a period. Periods always separate names on the directory_chain.
- Either the **file_name** or the **type** must be present; usually both are. The period between them is mandatory.
- The **version** is an integer. If you omit the semicolon and version, the highest number version of the file is used.

DECnet VAX File Names

If your VAX system is also connected to DECnet, you can exchange files with any DECnet computer. This "between networks" transfer requires you to explicitly route the copy through the VAX node that is on both networks.

To work with a DECnet VAX partner:

1. Use the **node_name** and **login** fields of the remote file name to specify the VAX computer that is on *both* networks.

2. Use the **file_name** field of the remote file name to specify the DECnet computer, along with the device, directory, etc. as described in the previous section.

The general syntax for a DECnet remote file is:

```
shared_vax#login#far_node"\"access_control\""::VAX_file_specification
```

- **shared_vax** is the name of the VAX computer that is on both networks.

- **login** is the authorization to use the shared VAX computer.

- **far_node** is the name of the DECnet node.

- The **access_control** is the authorization to use the DECnet VAX computer. It consists of a user name, followed by a space, followed by the user's password. You must include a backslashes and quote marks before and after, as shown.

Note

The access control string includes backslash-quote characters (\") before and after. In addition, the field is enclosed by quotation marks.

- The **VAX_file_specification** is covered in the previous section. It is a normal file specification for a DEC VAX system.

EXAMPLES

This is a sample file name for a file on a VAX node that is on DECnet, but which is not on AdvanceNet:

```
share_vax#karen:open4me#far_vax"\"jim  facts\""::jims_file.edt
```

This example file name illustrates the following: a computer named **share_vax** is on both the AdvanceNet network and the DECnet network. **share_vax** has a user called **karen**, whose password is **open4me**. A VAX computer called **far_vax** on the DECnet network has a user called **jim**, whose password is **facts**. The file **jims_file.edt** exists in the **jim**'s home directory. Notice the quotation marks (") at both ends of the access control field, and the backslash-quotes (\") around the access control string.

```
share_vax#karen:open4me#far_vax"\"sam  ivory\""::sys$mfg:[sue]memo.edt
```

In this example file name, a DEC VAX computer named **share_vax** is the same as the previous example. The VAX computer called **far_vax** (on the DECnet network) has a user called **sam**, whose password is **ivory**. The file **memo.edt** exists in the home directory of a different user, **sue**, whose workspace is on a different device than **sam**'s, a device named **sys$mfg**.

More Examples

The following examples show whole **DSCOPY** commands for transfers to a DEC VAX computer:

1. `dscopy my_list.txt VX1#karen:open4me#partlist.edt` **Enter**

2. `dscopy my_list.txt VX1#karen:#partlist.edt` **Enter**

   The first and second examples are identical, except that the second uses password prompting to maintain password security. Note the colon key in place of the password in the second example.

3. `dscopy VX1#karen:open4me#[.stock]panel.edt panel.txt` **Enter**

4. `dscopy VX1#karen:open4me#[.stock.bulk]wire.edt wire.txt` **Enter**

   The third and fourth examples show how to specify remote files that exist in the user's subdirectories. Note the initial period in the directory chain, and the period between directories in the chain.

5. `dscopy VX1#karen:open4me#[jim]memo.edt jimsnote.txt` **Enter**

   The fifth example shows how to specify a remote file in another user's work area. Note that the directory chain has no initial period.

6. `dscopy VX1#karen:open4me#[jim.law]legalnote.edt law.txt` **Enter**

   The sixth example shows how to specify a remote file in another user's subdirectory. Again, the directory chain has no initial period.

7. `dscopy VX1#karen:open4me#VXtax"\"bill taxman\""::FY89.edt tax.txt` **Enter**

   The seventh example uses a DECnet VAX computer. The computer that is on both the AdvanceNet and DECnet networks is **VX1**. Using **karen**'s authorization, it contacts **VXtax**, which is a VAX computer on DECnet. **VXtax** recognizes the user **bill** and his password, **taxman**, and authorizes NFT to copy the file **FY89.edt** to the PC workstation. A more complex file specification could be used after the two colons in the remote file name. Apply the rules in **Summary: File Name Syntax for the DEC VAX computer** above.

Notes

7 Using NFT Options

Recall that the syntax for the **DSCOPY** command is:

```
dscopy [options] <source_file> <target_file> Enter
```

In this chapter we explore NFT options.

What You Should Know:

Before you read this chapter, you should understand these concepts:

- The difference between **remote** and **local** computers
- The role of the **initiator**, **producer**, and **consumer** computers
- How to specify **local** and **remote** files for the systems you exchange files with
- What to do if errors occur

Turn to the chapter **Using NFT** for details on these and other topics.

Note

Some of the options described in this chapter require some technical knowledge to use effectively.

Quick Reference

The following table summarizes the options described in this chapter:

Conven-tional Options	Interactive	**-i**	Start an interactive session with NFT by typing: `dscopy -i` **Enter**
	Print	**-p**	Prints summary data about the transfer on your screen.
	File Lockword	**-P**	Allows file lockword prompting.
	Replace	**-r**	Target file replaces existing file of the same name.
Data-Type Options	ASCII	**-A**	Causes NFT to treat the data as **ASCII** data, regardless of its source.
	Binary	**-B**	Causes NFT to treat the data as **binary** data, regardless of its source.
Record Structure Options	Variable	**-V**	Causes NFT to create a target file with variable length records.
	Fixed	**-F**	Causes NFT to create a target file with fixed length records.
	Record Length	**-L**	Specifies the maximum record length for the transfer. Requires integer parameter.
Record Delimiter Options	Search Character	**-s**	Specifies the record delimiter character in the source file. Requires a literal or decimal ASCII parameter
	Delimiter Character	**-d**	Specifies the record delimiter character in the target file. Requires a literal or decimal ASCII parameter.

About This Chapter

This chapter introduces you to NFT options: what they are and how they are used. It contains:

- An Introduction to NFT Options.
- A description of how each option in the four groups works. Each description includes at least one example. An appropriate partner is used to illustrate the action of each option.
- A summary of the NFT options.

Introduction to NFT Options

If you have used remote storage (as described in **Using NFT**), you have already used an NFT option: the binary option. There are many others that you can use with the **DSCOPY** command.

What Is an NFT Option?

An option consists of a dash followed by a letter: for example, **-p**. Options always appear between the **DSCOPY** command and the source file specification. One or more options can be used on any transfer; options can be given in any order, but each one has to be set off with spaces.

What Are NFT Options For?

Using NFT options allows you to "customize" NFT to match your needs more completely. Some options affect the amount and kind of interaction you have with NFT during a transfer. Others manipulate the attributes of the file, so that the target file has the characteristics you desire.

Some options are very simple to understand and use. Others have effects which are technically important, but more subtle. The following example shows you what a **DSCOPY** command that uses options looks like. These and the other options are fully explained later. For now, only the form of the command is important.

EXAMPLE

```
dscopy  -p  -B  aerial1.pic  DataCentral#Karen:open4me#photos/aerial1  Enter
```

In this example, the source file is a photograph stored in binary form for display on a high-resolution monitor. Such files are typically very large, so it was moved to a remote computer with a large disc. Then the user can erase **aerial1.pic** from the PC workstation, to make room for other files.

```
dscopy  -p  -B  DataCentral#Karen:open4me#photos/aerial1  aerial1.pic  Enter
```

When the aerial photograph is needed again, this command copies the photograph back to the PC workstation as a file called **aerial1.pic**. Using a remote computer for remote storage is covered in the chapter **Using NFT**.

In both transfers, the **-p** option causes NFT to print the final outcome of the transfer on the display screen when the transfer is complete. The **-B** option tells NFT that the file contains binary data.

The Four Kinds of NFT Options

NFT options can be classified in four key functional groups:

- **Conventional** options
- **Data Type** options
- **Record Structure** options
- **Record Delimiter** options

Conventional options affect the way you work with NFT, and their effects do not depend on the contents of the source file. These options are especially easy to use.

Data Type, **Record Structure**, and **Record Delimiter** options are designed to manipulate certain attributes of the file during the transfer. Understanding the effects of these options and when these effects are desirable requires more technical knowledge.

The following sections cover the four groups in turn.

Conventional Options

In general, the effects of the conventional options do not depend on the contents of the source file. There are four options in this group:

- **-i**, the **interactive** option
- **-p**, the **print** option
- **-P**, the **Prompt for file lockword** option
- **-r**, the **replace** option

Interactive: -i

What the option does

Ordinarily, if you have several files to transfer, NFT has to re-establish communications between the computers every time you issue the DSCOPY command.

The **interactive** option, **-i**, starts up an interactive session with PC-NFT and allows you to execute more than one file transfer without breaking the connection between initiator, producer and consumer. By using the **interactive** option, you save the time it takes to keep re-establishing these connections. You also save typing whenever you transfer several files between the same computers.

When to use the option

Use the **interactive** option when you have several files to transfer. When you use this option, you get the benefit of faster transfers and less typing.

Using the Option

You do not specify a source or target file when you initiate an interactive NFT session. To begin an interactive session with NFT, type the following:

```
dscopy  -i  Enter
```

NFT will then begin prompting you for transfers. This is the prompt:

```
dscopy>
```

At the prompt, type in any other option you want to use, then a source file name, then a target file name.

You can think of an interactive session with NFT this way: the computer types the word **DSCOPY** for you, and adds the angle to let you know it's your turn. Then you take over, and type in the options and file names you want.

Using the Same Computers for Multiple Transfers

Interactive NFT keeps track of which computers you are accessing, and saves you typing the node names and logins for every transfer. For the first transfer, you type in the full file specification for the producer and consumer computers, as described in the chapters for the NFT partners. After the first transfer, interactive NFT keeps track of the computers involved. You only supply the file names for subsequent transfers; NFT assumes you are using the same computers in the same roles.

Changing to a Different Computer

During an interactive session with NFT, you may want to exchange several files with one computer, then switch to a another computer and exchange files with it. For example, you may want to copy three files to your PC workstation from **Computer1**, two files from **Computer2**, and then copy another four files *to* **Computer2** *from* the PC. Each step involves putting different computers in the roles of producer and consumer.

You can connect to different computers for either the producer or consumer roles (or both) at any time. The example for this section illustrates these two rules:

1. To connect to a new **remote** computer in either role, use a complete remote file specification, including the **node_name, login**, and **file_name** fields. This tells NFT to drop the current connection for the role, and establish a connection to the remote computer.

2. To switch either role from a remote computer to the PC workstation, type a pound-sign (#) before the local file name. This tells NFT to drop the connection to the remote computer and use a local PC file.

Ending an Interactive Session

When you are done using interactive NFT, press the CTRL key, and hold it down while you press C. NFT stops immediately. If a transfer is in progress, NFT asks whether you want to abort the transfer, or a progress report, as explained under **Interrupting NFT** in the chapter **Using NFT**.

EXAMPLE

Gray shaded text is entered by the user.

```
dscopy -i Enter
```

The user types this command at the DOS prompt, or from the PAM main screen. This starts the interactive session with NFT.

```
dscopy> my_list.txt  DataCentral#karen:#partlist  Enter
```

`target password>` (Type in, then press **Enter**. Remains invisible.)

Because the user is conscious of security, she uses password prompting (see chapter 3). The first transfer request names the PC as producer and a computer called **DataCentral** as consumer. The source file on the PC is **my_list.txt**; the file **partlist** is created on the **DataCentral** computer in **karen**'s home directory.

```
dscopy> proposal.txt  newventure  Enter
```

The file **newventure** is created in **karen**'s home directory on **DataCentral**.

```
dscopy> newhire.txt  employees/hiringplans  Enter
```

The file **hiringplans** is created on **DataCentral**, this time in **karen**'s subdirectory called **employees**.

```
dscopy> DataCentral#karen:#products/kits/toyboat  #toyboat.txt  Enter
```

`source password>` (Type in, then press **Enter**. Password remains invisible.)

Now the roles of producer and consumer are switched. The producer is now **DataCentral**, and the consumer is the PC. Note the pound-sign before the local file name. This pound-sign alerts interactive NFT that this file is local. Again, for security reasons, the user has employed password prompting.

```
dscopy> -B  graphics/graphic1.pcx  graph1.pcx  Enter
```

NFT remembers the new producer and consumer computers. The file **graphic1.pcx** is copied from its subdirectory on **DataCentral** to the file **graph1.pcx** on the PC. Note that the binary option, **-B**, is present: this transfer recovers a file that was stored remotely. You can use options in interactive NFT in the usual way.

Print: -p

What the option does:	When to use the option:
The **print** option causes NFT to print the final outcome of the file transfer on your screen. This offers you a way to check that the transfer completed successfully, and that the file you intended to copy across the network was copied.	You can use the **print** option on any transfer, at your discretion. The **print** option is never required, but always available. Use it whenever you want confirmation on your transfer.

Using the Option

Include the **print** option, **-p**, after the keyword **DSCOPY**, but before the source file specification. The option must have a blank space before and after.

EXAMPLE

If you enter the following command:

```
dscopy  -p  memo.txt  DataCentral#karen:open4me#taxmemo.edt  Enter
```

NFT responds, when the transfer is done, as follows:

```
81 records transferred for the source target pair:

memo.txt >> taxmemo.edt
```

Prompt for File Lockword : -P

What the option does

For those file systems that allow individual files to be password protected, the **prompt for file lockword** option provides password security. **Lockword** is the name applied to a file password. As with login passwords (see **Security and Password Prompting** in the chapter **Using NFT**), you use a key instead of a lockword in the file name argument. NFT then prompts you to enter the lockword; the lockword is never visible.

When to use the option

Use the **prompt for file lockword** option to maintain file lockword security on the remote partner you are exchanging files with.

Note

The HP 3000 is the only NFT partner that currently supports lockwords on individual files.

Using the Option

Include the **prompt for file lockword** option, **-P**, after the keyword **DSCOPY**, but before the source file specification. The option must have a space before and after. Use two colons (::) in place of the file lockword.

EXAMPLE

Since the HP 3000 is the only NFT partner that currently uses file lockwords, this example uses an HP 3000 to illustrate. The remote file is named **salarys**, and can only be accessed by persons who know the password, **harvest**.

```
dscopy  -P  admin#ken/::.sales/::#salarys/::  salarys.txt  Enter

source password 1>  nolimit  Enter  (not visible)

source password 2>  sell2day  Enter  (not visible)

source file password>  harvest  Enter  (not visible)
```

Because the user in this example is conscious of security, he is using password prompting throughout the command. Each pair of colons in the remote file specification is a password prompting key. Each key causes NFT to prompt for a password, which remains unseen while the user types. The last prompt is for the file lockword, which the user enters. When the last password is verified, the transfer begins.

Replace: -r

What the option does

Tells NFT what to do if a target file with the same name already exists. If the DSCOPY command specifies a target file that already exists on the consumer, NFT normally notifies you of the situation, and aborts without altering the existing target file.

The **replace** option tells NFT that you want the transfer to erase the existing file, and replace it with the file you are transferring. In either case, the source file is unaltered.

When to use the option

Use the **replace** option only when you want to replace the target file with the file you are transferring. The **replace** option is necessary when you want to replace an existing file with a new one of the same name.

Caution

If an existing file on the consumer is replaced during a transfer, the original contents are lost.

Using the Option

Include the **replace** option, **-r**, after the keyword **DSCOPY**, but before the source file specification. The option must have a blank space before and after.

EXAMPLE

Karen works with an HP 9000 partner called **DataCentral.** There is a file on that computer called **dailydata,** which she needs to update every day. There is no reason to save the previous day's data. She uses the following command to replace the old file with new data:

```
dscopy -r newdata.txt DataCentral#karen:open4me#dailydata  Enter
```

The old file named **dailydata** is erased. It is replaced by a new file called **dailydata.** This file contains the data copied from **newdata.txt**

Data Type Options

Recall the discussion in the chapter **Using NFT** about ASCII data and binary data. NFT offers two options to specify the kind of data a file contains.

Technically, DOS does not distinguish between the two data types. However, by convention the end of an ASCII record is marked by a pair of ASCII characters: a carriage-return (CR) followed by a linefeed (LF). This pair of characters is called the **record delimiter**. You can request a different record delimiter; see **Record Delimiter options** later in this chapter.

On systems that distinguish between these two kinds of data, you may want to use a Data Type option to create the appropriate kind of file.

There are two options in this group:

- **-A**, the **ASCII** option
- **-B**, the **binary** option

ASCII: -A

What the option does	When to use the option
For files copied *from* the PC, the **ASCII** option, **-A**, has no effect. For files copied *to* the PC, the option causes the file to be stored as if it were ASCII data, regardless of the attributes of the source file. That is, even if the original file claims to contain binary data, the ASCII CR-LF pair of characters is inserted between incoming records.	The ASCII option is available for transfers between systems other than DOS and HP-UX, which do not use it. The ASCII option may be necessary to overcome file attribute errors between various systems. For details, see the **NS Cross-System NFT Reference Manual.**

Using the Option

Include the **ASCII** option, **-A**, after the keyword **DSCOPY**, but before the source file specification. The option must have a blank space before and after.

EXAMPLE

This example uses an HP 1000 as the NFT producer.

```
dscopy  -A  MfgCntrl#karen/open4me#welddata  welding.dat  Enter
```

The source file in this example, **welddata**, is contains binary data. However, the target file, **welding.dat**, is stored like an ASCII file: it has the ASCII characters CR-LF between records.

Binary: -B

What the option does	When to use the option
For files copied *from* the PC, the binary option causes the file to be copied without regard to whether or not there are record delimiters. Everything in the file is transferred, including record delimiters. If the consumer distinguishes ASCII data from binary data, the copy is stored in a file suitable for binary data. For files copied *to* the PC, the option causes the file to be stored without delimiters, regardless of the attributes of the source file. That is, even if the original file claims to contain ASCII data, no delimiters are inserted between incoming records.	Use the **binary** option for remote storage transfers. The option may also be necessary to overcome file attribute errors between various systems. For details, see the **NS Cross-System NFT Reference Manual.**

Using the Option

Include the **binary** option, **-B**, after the keyword **DSCOPY**, but before the source file specification. The option must have a blank space before and after.

EXAMPLE

This example uses remote storage on an HP 9000 partner:

```
dscopy  -B  xport.plt  DataCentral#karen:open4me#graphix\exportchart.sav  Enter
```

The source file, **xport.plt**, contains binary data to drive a plotter. To store the file remotely on the HP 9000 partner, the user applies the binary option to the transfer. Later, the remotely stored file can be recovered from the HP 9000 by copying **exportchart.sav** back to the PC workstation, again with the binary option. See the chapter **Using NFT** for details about using remote storage.

Record Structure Options

Options in this group affect the way individual records are stored in the target file. NFT accommodates two record formats:

FIXED LENGTH:

Fixed-length records are of the same length throughout the file. If necessary, a record that is too short gets padded out to meet the required length. Records of binary data are padded with the ASCII NULL character. Records of ASCII data are padded with the ASCII SPACE character. In a file of fixed-length records, every record *must* be the same length.

VARIABLE LENGTH:

Variable-length records are allowed to be of many different lengths. A file of variable-length records may have every record a different length. Note that there is no reason that the records *must* be different lengths, only that they *may* be. Also note that in a given file there may be an upper limit on the length of a variable-length record.

Options in the Record Structure group let you specify the format of the records, and their maximum lengths. There are three options in this group:

- **-V**, the **Variable-length** option
- **-F**, the **Fixed-length** option
- **-L**, the **Record Length** option

Variable-length: -V

What the option does	When to use the option
The **variable-length** option, -V, has no effect with transfers involving a PC as either the producer or consumer. It is only used in transfers between two remote computers.	This option is only required when the PC acts as initiator for a file transfer between two remote computers. For transfers of this kind, refer to: **The NS Cross-System NFT Reference Manual.**

Using the Option

Include the **variable-length** option, **-V**, after the keyword **DSCOPY**, but before the source file specification. The option must have a blank space before and after. The **NS Cross-System NFT Reference Manual** covers the exact effects of the variable-length option for every pair of NFT partners.

EXAMPLE

This example shows a transfer between two remote partners, an HP 3000 producer and an HP 1000 consumer.

```
dscopy  -V  BayStore#bob/::,sales/::#orders  MfgCntrl#jim/planwell#incoming  Enter
```

In this example the **-V** option forces the HP 1000 to place the file called **incoming** in a file suitable for variable-length records.

Fixed-length: -F

What the option does	When to use the option
The effect of the **fixed-length** option, **-F**, is to cause the target file to have records of identical lengths. Some NFT partners have special file types for files of fixed-length records. If such a computer is the consumer, it creates a fixed-record type file as the target.	Use this option when the target file must be a file for fixed-length records. This typically necessary only to overcome file attribute errors when the consumer is a remote computer.

Using the Option

Include the **fixed-length** option, **-F**, after the keyword **DSCOPY**, but before the source file specification. The option must have a blank space before and after. Unless you use the **record length** option (discussed below) NFT assumes fixed records are 160 bytes long; if you also use the **binary** option, the length is 256 bytes. The **NS Cross-System NFT Reference Manual** covers the exact effects of the fixed-length option for every pair of NFT partners.

EXAMPLE

This example shows a transfer that uses the fixed-length option for a transfer to an HP 1000:

```
dscopy  -F  errdata.dat  MfgCntrl#karen/open4me#error_data  Enter
```

The target file, **error_data**, contains fixed-length records of 160 bytes. If any records in **errdata.dat** are longer than 160 bytes, the extra information is lost. The transfer completes normally, but NFT issues a warning that a record was truncated. If any records are shorter than 160 bytes, NFT fills the extra space with ASCII blanks, which act as padding. If you also employ the **binary**, **-B**, option NFT pads as necessary with ASCII NULL characters.

Record Length: -L

What the option does

The **record length** option, **-L**, allows you to specify the maximum length of records in the target file. Note the difference between fixed-length and variable-length records:

For fixed-length records, this option effectively specifies the length of every record in the file.

For variable length records, this option specifies the *greatest* record length permissible for the file. Many or all records may be shorter.

When to use the option

Use this option along with the **fixed-length** option to explicitly state the length of fixed-length records. This option may also be useful to overcome file attribute errors.

See the **NS Cross-System NFT Reference Manual** for details about how the record length option affects transfers between various partners.

Using the Option

Include the **record-length** option, **-L**, after the keyword **DSCOPY**, but before the source file specification. Without adding a blank space after the option, type the length you want. The option must have a blank space before the dash, and after the length.

EXAMPLE

The first example uses the record length option alone in a transfer to an HP 3000 partner. Assume that the application that will use the target file, **partlist**, cannot handle records longer than 80 bytes. However, a regular text transfer places a 160-byte upper limit on the file. The HP 3000 notes this, and the application later rejects the file based on the 160-byte record. This command corrects the problem:

```
dscopy  -L80  my_list.txt  BayStore#bob/::,sales/::#partlist  Enter
```

In this example, the HP 3000 creates a file that contains ASCII data, with every record 80 bytes long or less. Note that while the maximum length is set at 80, it is possible that all the source records are shorter anyway, so none get truncated. Nevertheless, the target file is registered as having records no longer than 80 bytes, and the application will therefore accept the target file as input.

The second example uses the **record length** option along with the **fixed-length** option to explicitly set the length of fixed records on an HP 1000:

```
dscopy  -L65  -F  errdata.dat  MfgCntrl#karen/open4me#error_data  Enter
```

The target file, **error_data**, has fixed-length records of 65 bytes each. Any records shorter than 65 bytes are padded to meet that length. Any records longer than 65 bytes are truncated, and the transfer continues. NFT issues a warning if records are truncated during a transfer.

Record Delimiter Options

Options in this group affect the way the computer distinguishes one record from another. There are two options in this group:

- **-s**, the **search character** option
- **-d**, the **delimiter character** option

Note

The Record Delimiter options only affect stream-type files. They have no effect in transfers to computers that do not support stream files. Currently, the only NFT partners that support stream files are the HP 9000 and DEC VAX computers, and the PC.

DOS Stream Files

As far as DOS is concerned, every file is just a continuous stream of undifferentiated bytes. Files of this kind are called **stream files.**

However, many application use the concept of a **record**. A record is a sequence of bytes that have some specific function. For example, in a text file, a record might consist of everything on a single line of text. The text editor manipulates text line-by-line, so the one-record-per-line idea is convenient. A file typically consists of many records, one after another.

Since DOS does not provide record-level support for file access, each application must use a convention to find where one record ends and the next begins. Under one common scheme, records are separated (or **delimited**) by an ASCII carriage-return (CR) character, followed by an ASCII linefeed (LF) character. This is the method NFT uses.

On other computers that use stream files, other delimiter conventions exist. NFT attempts to simplify transfers by translating between these conventions. The **record delimiter** options are useful in unusual circumstances when NFT's automatic translation is incorrect.

Search Character: -s

What the option does	When to use the option
With the **search character** option, -s, you can override the default record delimiter (as described previously under **DOS Stream Files**) in a source file. The search character option lets you declare any single character as the record delimiter a stream-type source file. NFT then searches for each record by scanning until it finds the next search character, or **delimiter**. Only the information *between* delimiters is transferred. The delimiter itself is not part of the data, and is *not* transferred to the target file. This option only affects the way NFT looks at the source file, and does not affect the contents of the source file.	Use the search character option when the source file was created by an application that uses an unusual record delimiter convention for the computer involved. You may need to use this option to correct certain file attribute errors.

Using the Option

Note

This option is ignored if you also use the **binary** option.

Include the **search character** option, -s, after the keyword **DSCOPY** but before the source file specification. Without adding a blank space after the option, type the search character you want to use. There must be a blank space before the dash, and after the search character. See the **NS Cross-System NFT Reference Manual** for more details about the effect of this option with your NFT partner.

There are two ways you can specify a search character:

- Literal character

  The easy way to specify a printing character as the search character is to type it immediately after the option:

  ```
  dscopy  -s%  my_file.txt  DataCentral#karen:open4me#new_file  Enter
  ```

  In this example, the search character is the percent symbol, %. Any printing character can be entered this way.

- Decimal ASCII value

  You can also specify any ASCII character by entering its ASCII value in base 10. Prefix the value with 0, to distinguish it from a literal value, as described above:

  ```
  dscopy  -s012  my_file.txt  DataCentral#karen:open4me#new_file  Enter
  ```

  In this example, the search character is the ASCII character FormFeed, (FF), whose decimal ASCII value is 12.

EXAMPLE

This example uses an HP 9000 NFT partner to illustrate the search character option:

```
dscopy  -s$  test.odd  DataCentral#karen:open4me#testfile  Enter
```

The target file, **testfile**, looks like the source file, except that every dollar-sign (**$**) in the source file marks the end of a record in the HP 9000 target file. The target file has no dollar-signs at all; they are all converted to record delimiters according to the HP 9000 convention.

Delimiter Character: -d

What the option does	When to use the option
With the **delimiter character** option, **-d**, you can override the default record delimiter (as described above) in a target file. The delimiter character option lets you declare any single character as the record delimiter in a stream-type target file. NFT inserts the delimiter character you declare between each record.	Use the delimiter character option when the target file will be used by an application that employs an unusual record delimiter convention. You may need to use this option to correct certain file attribute errors.

Using the Option

Note

This option is ignored if you also use the **binary** option.

Include the delimiter character option, -d, after the keyword **DSCOPY** but before the target file specification. Without adding a blank space after the option, type the delimiter character you want to use. There must be a blank space before the dash, and after the delimiter character. See the **NS Cross-System NFT Reference Manual** for more details about the effect of this option with your NFT partner.

There are two ways you can specify a delimiter character:

- Literal character

  The easy way to specify a printing character as the delimiter character is to type it immediately after the option:

  ```
  dscopy  -d%  my_file.txt  DataCentral#karen:open4me#new_file  Enter
  ```

  In this example, the delimiter character is the percent symbol, %. Any printing character can be entered this way.

- Decimal ASCII value

  You can also specify any ASCII character by entering its ASCII value in base 10. Prefix the value with 0, to distinguish it from a literal value, as described above:

  ```
  dscopy  -d012  my_file.txt  DataCentral#karen:open4me#new_file  Enter
  ```

  In this example, the delimiter character is the ASCII character FormFeed, **FF**, whose decimal ASCII value is 12.

EXAMPLE

This example uses an HP 9000 NFT partner to illustrate the delimiter character option:

```
dscopy  -d$  list.odd  DataCentral#karen:open4me#testlist  Enter
```

The target file, **testlist**, looks like the source file, except that every delimiter in the source file is converted to a dollar-sign ($) in the HP 9000 target file. The target file has no conventional record delimiters; instead, it has dollar-signs.

Notes

A

NFT Error Messages

This appendix contains a list of the error messages that you may receive when using the DSCOPY command.

Following each message is a description of its meaning and the probable cause of the problem. If the message resulted from an error, corrective action is described.

Note

All NFT errors begin as follows:

```
        dscopy:  ...
```

If you get an error message that does not have this form, refer to the network troubleshooting information included with this product.

Also note that in the following error messages, the producer is called the **source node**. The consumer is called the **target node**.

```
dscopy: internal NFT error
  (NFT error 1)
```

Explanation: Internal error.

User Action: Contact qualified HP personnel.

```
dscopy: unable to logon to source node
  (NFT error 2)
```

Explanation: Login string specified is invalid for the source node.

User Action: Check login entry and retry command.

```
dscopy: unable to logon to target node
  (NFT error 3)
```

Explanation: Login string specified is invalid for the target node.

User Action: Check login entry and retry the command.

```
dscopy: unable to access source file/device
  (NFT error 4)
```

Explanation:

1. Device is not ready or has failed; **or**
2. User does not have access privileges to the file or device.

User Action: Check the entry for correct syntax. If entry is correct, determine if the file/device has protected access.

```
dscopy: unable to connect to source node
  (NFT error 5)
```

Explanation: An attempt to make a connection from the initiator node to the source node failed. One or more of the following problems may exist:

1. Node name specified incorrectly, or other fields of node name defaulting incorrectly. Try specifying all three fields.

2. Network is not functioning on source node.

3. Source node is incorrectly configured.

4. Source node does not have enough memory space available to establish a connection.

5. The source node may be unable to service any more transfers than are currently taking place.

User Action: Determine the cause of the problem (starting with step 1) and correct the situation.

```
dscopy: unable to connect to target node
  (NFT error 6)
```

Explanation: An attempt to make a connection from the source node to the target node failed. One or more of the following problems exists:

1. Node name specified incorrectly, or other fields of node name defaulting incorrectly. Try specifying all three fields.

2. Network is not functioning on target node.

3. Target node is incorrectly configured.

4. Target node does not have enough memory space available to establish a connection.

5. The target node may be unable to service any more transfers than are currently taking place.

User Action: Determine the cause of the problem (starting with step 1) and correct the situation.

```
dscopy: insufficient resources are available on source node
  (NFT error 7)
```

Explanation: Source node does not have enough available resources to complete the file transfer.

User Action: Retry transfer when source node contains more memory.

```
dscopy: insufficient resources are available on target node
  (NFT error 8)
```

Explanation: Target node does not have enough available resources to complete the file transfer.

User Action: Retry transfer when target node contains more memory.

```
dscopy: source file was not found
  (NFT error 9)
```

Explanation: Source file does not exist or the source file path name or login was incorrectly specified.

User Action: Check your entry of the source file and retry the command.

```
dscopy: target file was not found
  (NFT error 10)
```

Explanation: Target file path name or login was incorrectly specified.

User Action: Check your entry of the target file and retry the command.

```
dscopy: transfer aborted per user request
  (NFT error 11)
```

Explanation: The transfer was aborted at your request.

User Action: None.

```
dscopy: requested data type was refused
  (NFT error 14)
```

Explanation: Either the producer or consumer cannot support the transfer of the data type requested.

User Action: Check the producer and consumer node manuals to determine which data types they support and retry command.

```
dscopy: requested record type was refused
  (NFT error 15)
```

Explanation: Target node could not support requested record type or source record type not supported by NFT producer.

User Action: Refer to the *NS Cross-System NFT Reference Manual* for supported file types.

```
dscopy: requested file type was refused
  (NFT error 16)
```

Explanation: Either the producer or consumer cannot support the transfer of the file type requested.

User Action: Try transferring the file as a different file type.

```
dscopy: requested record size was refused
  (NFT error 17)
```

Explanation: Requested or default record size exceeds the maximum record size for the target node.

User Action: Reduce the size of the records for the transfer.

```
dscopy: requested file size was refused
  (NFT error 18)
```

Explanation: File size specified would exceed the space available to the target login.

User Action: A smaller file size value must be specified for the transfer.

```
dscopy: target record size is invalid
  (NFT error 20)
```

Explanation: The record size entered by the user exceeds the maximum integer value.

User Action: Retry command with a valid record size.

```
dscopy: duplicate target file
  (NFT error 22)
```

Explanation:

1. Target file already exists and the REPLACE option was not specified; **or**

2. Source and target file names represent the same file.

User Action:

1. Retry the transfer with the REPLACE option.

2. Check to see if the source file name and target file name refer to the same file.

```
dscopy: need password to access source file
  (NFT error 23)
```

Explanation: You must supply a password to access the source file.

User Action: Reenter the **DSCOPY** command, and supply the file lockword. You may want to use the file lockword option.

```
dscopy: need password to access target file
  (NFT error 24)
```

Explanation: You must supply a password to access the target file.

User Action: Reenter the **DSCOPY** command, and supply the file lockword. You may want to use the file lockword option.

```
dscopy: out of disc space
  (NFT error 25)
```

Explanation: Target node ran out of disc space during the file transfer.

User Action: Remove unneeded files from the target node.

```
dscopy: connection to source node has gone down
  (NFT error 26)
```

Explanation: Connection between the initiating node and the source node was established, but failed before the transfer was complete.

User Action: Check to see if the producer node is still operating on the network. Check its network log file for additional information.

```
dscopy: connection to target node has gone down
  (NFT error 27)
```

Explanation: Connection between the source node and the target node was established, but failed before the transfer was complete.

User Action: Check to see if the consumer node is still operating on the network. If a timeout occurred, the remote system may be busy. Try the command again when the remote system is not so busy. If necessary, check its network log file for additional information.

```
dscopy: unable to purge target file
  (NFT error 28)
```

Explanation: Replace option used, but NFT could not delete existing target file.

User Action: Check permission of target file.

```
dscopy: invalid target file id
  (NFT error 29)
```

Explanation: Target file name specified had an invalid syntax; **or** the target file is a directory.

User Action: Check your entry for the target file, and retry the command.

```
dscopy: read from source file failed
  (NFT error 31)
```

Explanation: Source file was opened but a read from it failed during the transfer. This occurs when a flexible disc is removed while being read.

User Action: Re-insert the flexible disc and try again. If the PC is not the producer node, notify the system manager for the producer node of the file system error.

```
dscopy: write to target file failed
  (NFT error 32)
```

Explanation: Target file was opened but a write to it failed during the transfer. This error will occur when a flexible disc is removed while being written.

User Action: Re-insert the flexible disc. Then retry the command. If the PC is not the consumer node, notify the system administrator of the consumer node that a file system error has occurred.

```
dscopy: unable to create or open target file
  (NFT error 33)
```

Explanation: Target file could not be opened for writing. Target disc drive may be empty or the drive door may be open.

User Action: Insert a formatted flexible disc in the drive and close the drive door. Then retry the command.

```
dscopy: invalid or unsupported source device
  (NFT error 34)
```

Explanation: Source file is a device which NFT does not support for reading or the device does not exist on the system.

User Action: Check your entry of the source device.

```
dscopy: invalid or unsupported target device
  (NFT error 35)
```

Explanation: Target file is a device which NFT does not support for writing or the device does not exist on the system.

User Action: Check your entry of the target device.

```
dscopy: unable to close target file
  (NFT error 36)
```

Explanation: Target file could not be closed after the transfer.

User Action: Notify system manager of the target node that a file system error has occurred.

```
dscopy: incorrect source file password
  (NFT error 38)
```

Explanation: Password string specified is invalid for the source file.

User Action: Reenter the **DSCOPY** command, and supply the correct password.

```
dscopy: incorrect target file password
  (NFT error 39)
```

Explanation: Password string specified is invalid for the target file.

User Action: Reenter the **DSCOPY** command, and supply the correct password.

```
dscopy: target file name has been truncated
  (NFT error 42)
```

Explanation: Target file was not specified and the source file name was too long or contained illegal characters, or the target file specified was too long or contained illegal characters. Illegal characters were removed and/or the name was truncated.

User Action: Use a shorter file name and/or ensure that the file name does not contain illegal characters.

```
dscopy: (warning) source and target file attributes differ
  (NFT error 43)
```

Explanation: Target node counter-offered file attributes which differ from those of the source file. A set of attributes was chosen and the transfer continued. For example, a direct access file was transferred to the PC in sequential mode.

User Action: Check the attributes of the target file.

```
dscopy: (warning) record(s) truncated to fit maximum target file record size
  (NFT error 44)
```

Explanation: At least one source logical record is larger than the maximum size specified. Trailing data in the record beyond the maximum size were truncated.

User Action: Use the record length option to specify a larger record length if you want to transfer the entire file.

```
dscopy: unable to access target file/device
  (NFT error 48)
```

Explanation:

1. Device does not exist or has failed; **or**
2. User does not have access privileges to the file or device.
3. The specified target file name is the name of a directory on the PC.

User Action: Check the entry for correct syntax. If entry is correct, determine if the file/device has protected access.

```
dscopy: invalid source file id
  (NFT error 49)
```

Explanation:

1. Source file path name contained illegal or incorrect characters or had illegal or incorrect syntax; **or**
2. Path name specified was bad.

User Action: Check your entry for the source node and retry the command.

```
dscopy: overwrite not supported
  (NFT error 52)
```

Explanation: The replace option is not supported on the consumer.

User Action: Reenter the **DSCOPY** command without the replace option.

```
dscopy: unable to start nft service on the source node
  (NFT error 55)
```

Explanation: Cannot create producer process at this time.

User Action: Retry the transfer. If you experience this failure continually, contact the system manager or qualified HP personnel.

```
dscopy: unable to start nft service on the target node
  (NFT error 56)
```

Explanation: Cannot create consumer process at this time.

User Action: Retry the transfer. If you experience this failure continually, contact the system manager or qualified HP personnel.

```
dscopy: found end of file mark in target file
  (NFT error 66)
```

Explanation: The consumer was unable to write to the target file because End-of-File was encountered

User Action: Reallocate space for the file on the target computer.

Notes for Programmers

Like other applications and commands, you can load and execute NFT from another application. There are a number of ways to accomplish this, and it is not the intent of these notes to describe these techniques.

These notes cover information that will be of interest if you are writing an application that uses NFT, regardless of the method you use.

Return Codes

When NFT terminates, you may find the return code useful:

- If it is zero, NFT has terminated normally.
- If it is non-zero, a error has been detected.

Return Code	Meaning
0 (Zero)	Success.
1 through 66	Return codes in this range correspond directly to the error message numbers in Appendix A.
-1 **(255 unsigned)**	The load and execute failed, and NFT did not attempt a transfer.
-2 **(254 unsigned)**	NFT detected a syntax error in the DSCOPY command.

Screen Ownership

While NFT is executing, it has ownership of the screen and may write several lines of information. NFT has written to the screen if any of the following are true:

- the return code is non-zero
- you used the print option, **-p**
- you used password or file lockword prompting
- the user typed **CTRL-C** during a transfer.

Bear this in mind as you develop your application; you should take some appropriate action when NFT returns. Two alternatives follow. There may be others:

1. Repaint the screen immediately when NFT returns, since even a return code of zero does not guarantee an uncorrupted screen.
2. Give NFT a window to write in, and see that the cursor is still located in the window after NFT returns. Under most circumstances, NFT will write 6 or fewer lines. However, there are cases where NFT could write more. For example, a user could make sev^ral status requests during a transfer, or a single transfer could encounter multiple errors.

Interrupt Handling

NFT installs its own INT 23 and INT 24 handlers. INT 23 handles **CTRL-C** interrupts, and INT 24 handles critical errors. If your application has conflicting handlers installed, they will be inactive while NFT executes. They will be reinstalled when NFT returns.

This means that while a file transfer is in progress, a **CTRL-C** from the user will be handled by NFT, as noted in the chapters **Using NFT** and **Using NFT Options.**

C

Glossary

Consumer This computer is the destination for the copy. It consumes the copy that the producer sends, and stores it for later use. The file that NFT creates on the consumer is called the target file.

Initiator This computer reads and interprets the **DSCOPY** command. It initiates the process of copying the file from one computer to another. Its only function is to read your command and later notify you of the results. The PC workstation is always the initiator. It is frequently the producer or consumer as well.

Local computer The local computer is the one you are working on. If you did not have a network, you would still have your local computer. Your PC workstation is always the local computer.

Local file A file on the PC is called a local file.

NFT Partner A general term that refers to any computer with which you can exchange files. The NFT partner may be the producer or consumer.

Node A node is just another name for any computer on the network. It can be either a PC workstation or another computer.

Options NFT options are qualifiers inserted after the **DSCOPY** key word. These qualifiers modify the characteristics of NFT in a variety of ways.

Producer This computer has the original file that you want to copy. It produces a copy of the file in the same sense that a printer produces a copy of a book. The original file is called the source file; NFT does not affect the source file in any way.

Remote computer All the other computers on the network are remote. The resources of remote computers are only available over the network. A PC workstation is never the remote computer for NFT.

Remote file A file on a remote computer is called a remote file.

Remote storage The practice of using NFT to move files to and from a larger computer's disc. Remote storage is a way to extend the amount of storage available to you, by placing some of your files on another system's disc.

Index