DECnet/E
Guide to User Utilities

Order No. AA-H504B-TC

January 1982

The DECnet/E utilities allow a terminal user to: talk to a user locally or at another node, log in at and use another DECnet/E node, manipulate files between nodes, and copy the contents of storage devices between DECnet/E nodes.

Operating System and Version: **RSTS/E V7.1**

Software Version: **DECnet/E V2.0**

To order additional copies of this document, contact your local Digital Equipment Corporation Sales Office.

digital equipment corporation · maynard, massachusetts

First Printing, January 1982

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

Distributed and Mid-Range Systems Publications typeset this manual using DIGITAL's TMS-11 Text Management System.

310ALL

Contents

Chapter 4 NFT: Network File Transfer Utility

Chapter 5 NETCPY: Network Copy between Devices

Chapter 6 NETOFF: Network Shutdown

Appendix A Object Type Codes

Appendix B NSP Reason Codes for Connect Reject and Link Abort

Appendix C NETACT: Automatic Log-in Utility for NFT

Appendix D Error Messages Generated by RMS and DAP

Index

Figures

Preface

Manual Objectives

DECnet/E is a combination of hardware and software that extends the capabilities of a RSTS/E system on a DIGITAL PDP–11 computer to allow communication with other DECnet systems in a network. This manual describes the DECnet/E utilities that allow a terminal user to talk to a user locally or at another node, log in at and use another DECnet/E node, manipulate files between nodes, and copy the contents of entire disks and tapes between devices at different DECnet/E nodes.

Intended Audience

This manual is written for the nontechnical RSTS/E terminal user who wishes to use the DECnet/E facilities to access remote systems within his network. The reader is expected to have a basic understanding of the operation of the local RSTS/E system, as well as that of the remote system being accessed, but is not expected to be a sophisticated computer user or programmer.

Related Documents

The terminal user who wishes to use the DECnet/E utilities will find all the information necessary to do so in this manual. To obtain a general understanding of the DECnet environment, however, the reader is directed to the following manual:

Introduction to DECnet, Order No. AA-J055B-TK

DECnet/E also provides message services that allow a programmer to develop programs that exchange data with programs running on other DECnet systems in a network. These features are described in four companion manuals:

DECnet/E Network Programming in BASIC–PLUS and BASIC–PLUS–2, Order No. AA-H501B-TC
DECnet/E Network Programming in MACRO–11, Order No. AA-L265A-TC
DECnet/E Network Programming in FORTRAN, Order No. AA-L266A-TC
DECnet/E Network Programming in COBOL, Order No. AA-H503B-TC

Two other companion manuals tell how to generate and start a DECnet/E system, control and monitor a running DECnet/E system, and give other information useful to a system manager.

DECnet/E Network Installation Guide, Order No. AA-K714A-TC
DECnet/E System Manager's Guide, Order No. AA-H505B-TC

Structure of This Manual

This manual is structured as a reference guide. It is divided into chapters, one for each available network utility. The function and capabilities of each utility are described and step-by-step instructions are presented for their use.

Documentation Conventions

Throughout this manual, the following documentation conventions are used to describe keyboard input:

UPERCASE	Uppercase letters indicate keywords that must be entered as shown.
lowercase	Lowercase italic letters represent items that must be replaced according to the accompanying description. When an item requires a descriptor or more than one word, the descriptor is hyphenated. For example, *file-specification*.
[ ]	Square brackets enclose an optional field that need not be included. In general, brackets are logical symbols only and should not be included in the input. Brackets surrounding a project-programmer number, however, must be included.
{ }	Braces indicate a choice of input items, one of which must be specified.
red/black	User input is shown in red. Program output (prompts and displays) is shown in black.
CTRL/	The character string "CTRL/" followed by a letter indicates a control character (for example, CTRL/Z). Control characters are generated by depressing the CTRL key and the appropriate letter key at the same time and are echoed on the console display as a caret (^) followed by the letter (for example, ^Z).

Unless shown otherwise, spaces or tabs must separate items in a command string. More than one space or tab between items is treated by the system as a single space. Also, although it is not shown in this manual, you must follow the last item of input in a command string with a carriage return.

Chapter 1
Introduction

This manual tells how to use DECnet/E utilities to perform the following operations:

1. Communicate with someone at a terminal on another DECnet system that has a compatible receiver utility. Each person can type messages that will be displayed at the other's terminal (see TLK, Chapter 2).
2. Do work on another DECnet/E system as if your terminal were connected directly to the other system (see NET, Chapter 3).
3. Copy files to or from other DECnet systems, submit command files to other systems, and delete files at other systems (see NFT, Chapter 4).
4. Copy the contents of an entire storage device to or from another DECnet/E system (see NETCPY, Chapter 5).
5. Shut down the network in an orderly fashion (see NETOFF, Chapter 6).

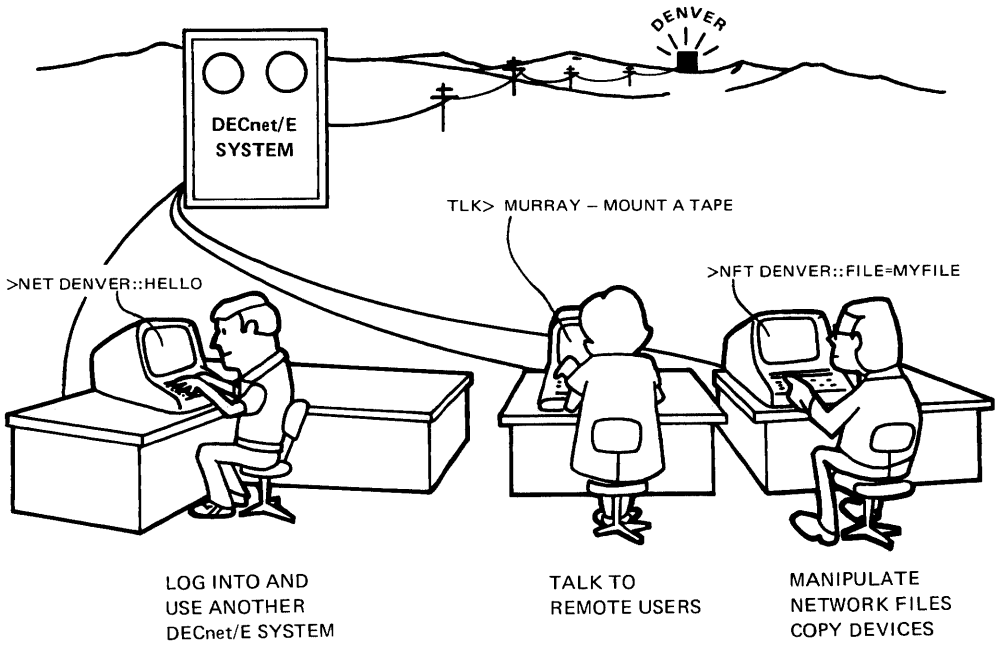


Figure 1–1 DECnet/E Features for the Terminal User

1.1 Basic Terms and Concepts

The term *network* is used here to refer to two or more computer systems connected so that they can exchange information. The computer systems, called *nodes*, are connected by communications paths called *physical links*. Physical links can be established through normal (switched) telephone circuits, several varieties of leased (private) lines, coaxial cables, or even satellite transmission facilities provided by several carriers.

In the network shown in Figure 1–2, a user at node BOSTON would refer to BOSTON as the *local node* and to nodes DENVER, DALLAS, LONDON, and LA as *remote nodes*. At node BOSTON, the nodes DENVER, DALLAS and LONDON are physically *adjacent nodes*. That is, there is a direct physical link to these nodes with no intervening nodes. To a user at node LA, DENVER is the only adjacent node.

A *path* is the route over which data travels from its source to its destination within the network. *Path length* is the distance from the source node to the destination node, measured in hops. A *hop* is equal to a physical line between two adjacent nodes. In Figure 1–2, the path length between nodes DALLAS and BOSTON is one hop and between LA and LONDON is three hops. The *network diameter* is the maximum path length between any two nodes. The network shown in Figure 1–2 has a network diameter of three hops.

The utilities described in this manual establish *logical links* to communicate with their counterparts at remote nodes. A logical link is a software path for the exchange of data between two programs running in a network. DECnet/E Version 2.0 is a Phase III DECnet product. (Refer to *Introduction to DECnet* for a discussion of the differences between Phase II and Phase III DECnet products.) As such, it supports the adaptive routing feature of Phase III, providing the user with the capability of communicating with other Phase III nodes in the network, regardless of whether or not they are directly connected to the local node.

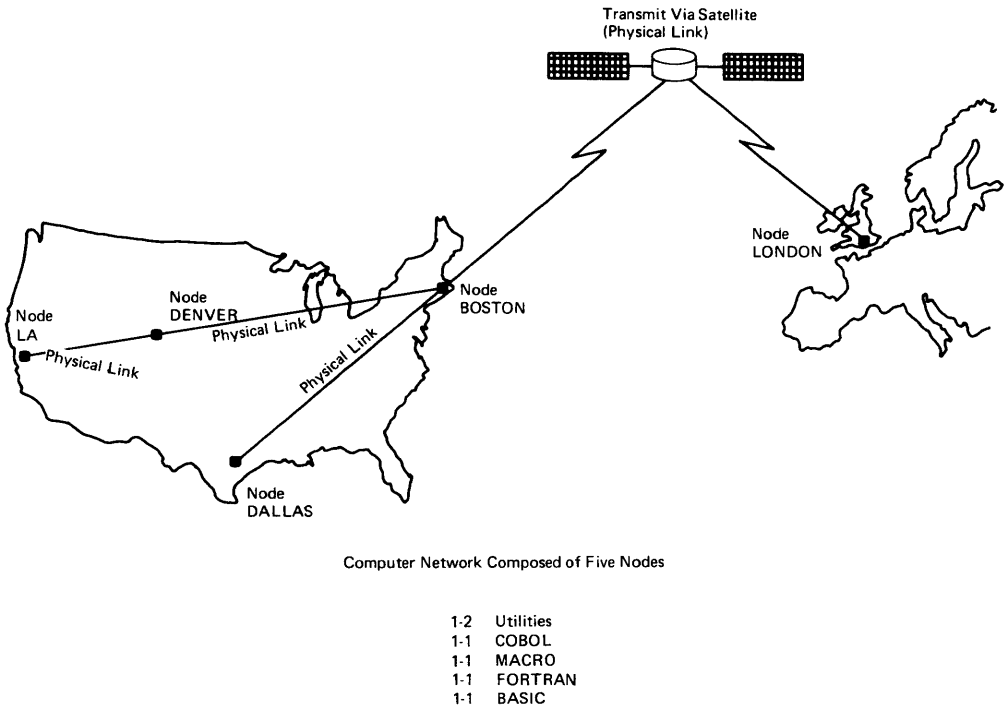


Figure 1–2 Computer Network Composed of Five Nodes.

1.2 DECnet Networks

DECnet as a whole refers to the hardware and software that allows different DIGITAL systems to be connected to form networks. DECnet has been designed to allow each system to function as a separate entity but still allow access to resources that are distributed throughout the various computer systems in the network. A DECnet/E system provides all the capabilities of a normal RSTS/E system plus the networking capabilities described here and in the companion DECnet/E manuals listed in the Preface.

All DECnet implementations are based on a design structure called the DIGITAL Network Architecture (DNA). This architecture allows network implementations on the various DIGITAL systems to evolve (to provide more and more capability as time goes on) within a basic common structure. The structure ensures that implementations providing fewer features will always work with implementations providing more.

Perhaps a more immediate aspect of this process of evolution, however, is that at any given time capabilities can indeed differ from system to system. The DECnet implementations for all the different DIGITAL systems are not described here. Nevertheless, to use the DECnet/E utilities, you do need to give some thought to other DIGITAL systems in your network.

If your network consists entirely of DECnet/E Version 2.0 nodes, this document provides all the information you need to use the DECnet/E utilities. If your network contains other DECnet systems, however, you will need to check the DECnet documentation for the other systems to be sure that the corresponding capabilities exist.

The DECnet/E utilities TLK and NFT are designed and implemented according to the protocols defined as part of the DIGITAL Network Architecture. You can use them with non-DECnet/E systems as long as the corresponding feature is implemented in that system. The DECnet/E utilities NET and NETCPY, on the other hand, are networking features unique to DECnet/E and will work only with other DECnet/E systems.

A brief overview of the functional capabilities of the various DECnet implementations can be found in *Introduction to DECnet.*

1.3 The DIGITAL Network Architecture (DNA)

DECnet/E is implemented according to a set of rules, or protocols, governing the format, control, and sequencing of data exchange from the user program level down through the physical link level. These protocols are defined by the DIGITAL Network Architecture (DNA) – a logical structure that provides the model for all DECnet implementations.

DNA consists of several layers, each defining a distinct set of network functions and a set of rules for implementing those functions. Each DECnet implementation consists of software modules that perform these DNA-defined functions according to DNA-defined protocols.

At the lowest layer, physical link control is achieved in DECnet by implementation of the Digital Data Communications Message Protocol (DDCMP). DDCMP ensures an error-free, sequential data path over a generally error-prone medium. This is accomplished with the Cyclic Redundancy Check (CRC) for error detection, retransmission for error corrections, and numbered data segments to ensure sequential transmission of data.

The Network Services Protocol (NSP) defines the rules for communication between programs in a network over paths called *logical links*. Logical links are described in detail in the network programming manuals for the various DECnet/E language interfaces, as named in the Preface. Briefly, logical links allow many different data streams to be multiplexed for transmission over the physical link and separated at the receiving node for delivery to the appropriate program. The utilities described in this manual communicate using logical links.

The Transport protocol defines the rules for determining the actual physical path, or *route*, along which data travels to its destination.

The Data Access Protocol (DAP) defines conventions for DECnet utilities that transfer and manipulate files. This protocol includes a specific language that these utilities use to transfer commands and data. The NFT and FAL utilities described in this manual communicate according to the DAP protocol.

1.4 Network Routing

DECnet/E Version 2.0 is a Phase III DECnet product. (Refer to *Introduction to DECnet* for a discussion of the differences between Phase II and Phase III DECnet products.) As such, it supports the adaptive routing feature of Phase III, providing the user with the capability of communicating with other nodes in the network regardless of whether or not they are directly connected to the local node.

Networks can consist of the following three types of nodes:

- **Routing nodes** – Phase III nodes connected to multiple communications lines, supporting route-through capabilities.
- **Nonrouting nodes (end nodes)** — Phase III nodes connected to a single communications line.
- **Phase III nodes** – nodes running a previous generation of the DECnet architecture.

Phase III nodes can communicate with and are compatible with Phase II nodes. However, Phase II nodes do not acquire any new capabilities by being connected to Phase III nodes, and the restriction that Phase II nodes can only communicate with adjacent nodes continues to apply. Thus, a Phase III node can communicate with any other Phase III node, *providing the path goes through Phase III nodes only.*

The adaptive routing feature is implemented through an algorithm that chooses the routing path with the lowest associated cost. Cost is computed as the sum of the costs of the lines over which the message is transmitted. The individual line cost parameters are assigned by the system manager and input into the system using the Network Control Program (NCP). (See the *DECnet/E System Manager's Guide* for a full discussion of line costs.) The algorithm automatically adjusts the routing when network topology or line cost changes occur.

1.5 Overview of DECnet/E Structure

You do not really need to know how DECnet/E is implemented to use the utilities described in this manual. It is probably useful, however, to have some appreciation for how the utilities themselves fit into the overall DECnet design and implementation.

As mentioned before, the utilities described in this manual communicate over the network using logical links. The DECnet/E implementation of the Network Services Protocol, that provides the capability for establishing logical links, is a part of the RSTS/E monitor. Although it is not a separate entity or program in DECnet/E, the term NSP is used in this manual and the companion DECnet/E manuals to refer to the software that allows a user program to establish and exchange data over logical links. In other words, NSP is the implementation of the Network Services Protocol.

NSP, the utilities described in this manual, and NCP, the Network Control Program (described in the *DECnet/E System Manager's Guide*), are visible to the user. To code a program that exchanges information with another program, a programmer uses calls that NSP executes. A terminal user runs TLK, NFT, NETCPY, and NET to use their services. The system manager runs NCP to start, monitor and control the local node and other nodes in the network.

Some elements of DECnet/E, however, are not directly accessible to any user. These lower level elements of DECnet/E include the Transport module (TRN), that implements the Transport protocol, several hardware devices (the DMC11, DMR11, DMV11, and DMP11) and their associated software device drivers.

The device drivers are responsible for handling messages received from and sent over the physical communication lines. NSP gives an outgoing message to Transport, and Transport selects an appropriate data path based on the destination of the message. Transport then passes the message to the proper device driver for actual transmission. The drivers manipulate the device registers to cause message transmission to occur.

To handle incoming messages, the drivers include buffer management functions that allow messages to be received from the physical links. When a message is received, it is passed to Transport, and Transport checks the destination of the message. If the message is for the local node, Transport passes it to NSP for further processing. If the message is destined for another node in the network, Transport selects a data path and passes the message to one of the drivers for transmission.

The hardware devices are intelligent communication controllers that connect the PDP–11 to physical communication lines. The DMC11 and the DMR11 control lines that connect to only one other system in the network (point-to-point lines) while the DMV11 and the DMP11 control lines that can connect to more than one other system (multipoint lines). The physical lines can include cables, modems and telephone circuits, or even satellite transmission facilities. Each controller contains a microprocessor, its own memory, and microcode that implements the DDCMP protocol. The implementation of DDCMP in firmware considerably reduces the software overhead for the DECnet/E system.

Chapter 2
TLK: Terminal Communication Utility

The TLK utility lets you send messages to another person at any other terminal in the network, including the Operator Services Console (OSC), provided that the system to which the remote terminal is connected has a utility designed to receive such messages. Two communication modes are available.

1. One-line mode allows transmission of a one-line message to another terminal.
2. Dialog mode allows two-way communication between two terminals. The dialog can be stopped at any time by the person at either terminal.

The TLK utility is implemented as two programs in the RSTS/E environment: TLK (Talk) and LSN (Listen). Figure 2–1 shows a sample conversation in dialog mode, and also illustrates the general relationship of TLK and LSN. TLK is executed at a RSTS/E node at the direct request of one of the local terminal users. LSN is executed to complete a connection requested by a TLK program.

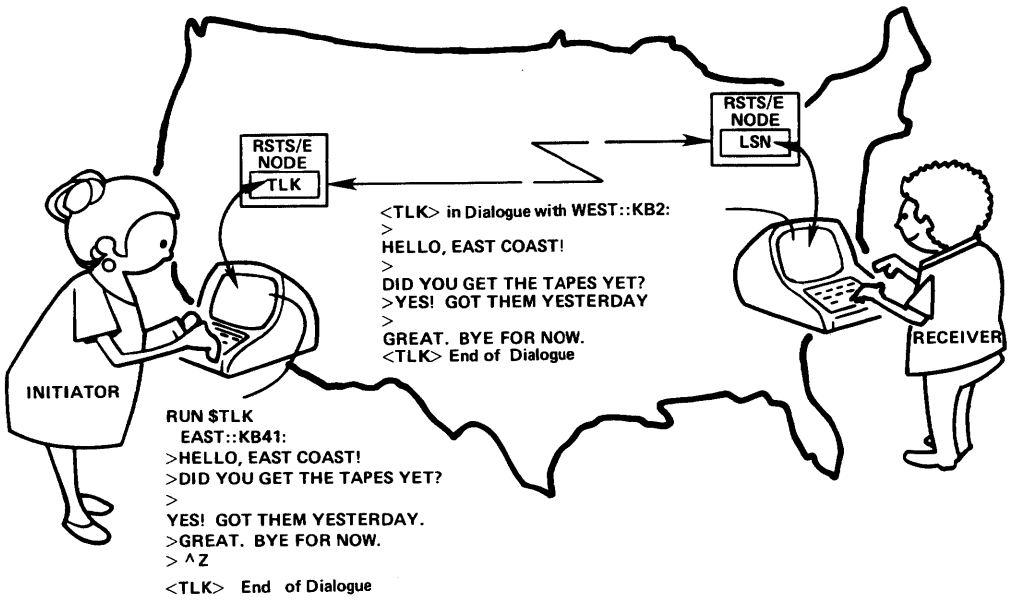


Figure 2–1: The TLK Utility: TLK and LSN Provide Communication

2.1 One-line Mode

Using the TLK utility in one-line mode, you can transmit one line of information to another terminal. The following subsections describe the input required to initiate a message and the output displayed at the receiver's terminal.

2.1.1 Initiator's Input

To send a one-line message to another terminal, type:

```
RUN $TLK
TLK V2.0 RSTS V7.1 Timesharing NSP
*[target-node::][target-terminal:]  'message
```

ONE-LINE MODE: MESSAGE, PRECEDED BY A SINGLE QUOTE, ON SAME LINE AS TARGET INFORMATION

If TLK has been installed as a Concise Command Language (CCL) command, you can also type

```
TLK [target-node::][target-terminal:]  'message
```

In either case, the length of a line cannot exceed 132 characters.

The *target-node* is the 1- to 6-character ASCII node name to which the message is to be sent. If the local node is the target, the node name can be omitted. If a node name is given, the two colons (::) must be used. (The names of nodes that are currently active on the network can be determined with the NCP command, SHOW ACTIVE NODES. See the *DECnet/E System Manager's Guide.*)

The *target-terminal* is the device designation of the terminal to which the message is to be sent (for example, TT5: or KB41:). If the operator's console is the target, the target terminal can be omitted. TLK will then automatically send the message to the operator's console at the node specified. (Under the RSTS/E system, for example, the operator's service console can be assigned to different physical devices at different times for the operator's convenience. By not specifying a specific device, such as KB0:, you ensure that the message reaches the current operator's console device.)

If the node specified is the local node, TLK will check to make sure the target terminal specified is a terminal known to the system. The usual designation for terminals under the RSTS/E system is KB*n*: or TT*n*:, where *n* is the physical terminal number. However, logical names established with an ASSIGN command can also be used for target terminals at the local node. (See the *RSTS/E System User's Guide* for a discussion of logical device names.)

If the node specified is a remote node, TLK extracts only the decimal terminal number from the target terminal specification and forwards that number to the TLK (or LSN) program at the remote node. The remote node's TLK (or LSN) utility will return an error if the device number is invalid for that node.

The *message* is any ASCII string; it is the message to be transmitted. The message must be preceded by a single quote and cannot extend beyond the end of the current line. Leading and trailing blanks and tabs will be deleted. (The entire line cannot exceed 132 characters.)

Example:

The following example shows a one-line message sent from the local node (known as MAYNRD on the network) to terminal KB5: at a remote node called SANFRN.

```
RUN $TLK
TLK V2.0 RSTS V7.1 Timesharing NSP
*SANFRN::KB5: 'Relay my calls to (617) 123-4567.  Sheldon.
```

The following example shows a one-line message sent to the operator's service console at the local node:

```
TLK 'MOUNT TAPE MM015 ON UNIT MM1:
```

2.1.2 Receiver's Output

A one-line message targeted to a RSTS/E terminal is displayed as follows:

<TLK> *source-node*:: *source-terminal*: *message*

The *source-node* is the name of the node where the message originated. It is omitted if the message came from the local node. The *source-terminal* is the device designation of the terminal from which the message originated. The *message* is the ASCII message.

The LSN program at the RSTS/E node sends the message to the designated terminal. If the terminal is busy, the executing program is not halted but the message is displayed regardless of whatever else may be going on. That is, the line can be displayed with other text being displayed at the terminal.

A message arriving for the local node's operator services console when OPSER (the operator services program) is running is displayed in the usual OPSER format:

MESSAGE *nnnnn:date time* JOB:*nn jobname*[*ppn*]
<TLK> *source-node*:: *source-terminal*:
'*message*

If OPSER was not installed as part of the RSTS/E system, or if for some reason it is not running, the message will be broadcast to KB0: in the ordinary format.

(OPSER provides various capabilities useful to a computer operator or system manager. The program is described in detail in the *RSTS/E System Manager's Guide.*)

Example:

Assuming that the first one-line message in the previous example arrived at a RSTS/E node, the message would appear as follows:

```
<TLK> MAYNRD:: KB4: 'Relay my calls to (617)123-4567,  Sheldon,
```

If OPSER is running, the second one-line example would appear at the local node as follows:

```
MESSAGE 23402 : 19-Mar-82 12:45 PM JOB:13 TLK[122,100]
        <TLK> MAYNRD:: KB4:
        'MOUNT TAPE MM015 ON MM1:
```

2.2 Dialog Mode

Using the TLK utility in dialog mode, you can enter into an exchange with a user at another terminal. The following subsections describe the input required to initiate such a dialog, as well as the output displayed at the receiver's terminal and the input required from the receiver to continue the conversation.

2.2.1 Initiator's Input

To initiate a dialog with a person at another terminal, type the following:

RUN $TLK
TLK V2.0 RSTS V7.1 Timesharing NSP
*[*target-node*::][*target-terminal*:]
>*message*

DIALOG MODE: FIRST MESSAGE ON LINE AFTER TARGET INFORMATION, NO SINGLE-QUOTE NEEDED

If TLK has been installed as a Concise Command Language (CCL) command, you can also type the following:

TLK [*target-node*::][*target-terminal*:]
>*message*

The *target-node*, *target-terminal*, and *message* input parameters are the same as for one-line mode except that the message does not require the preceding single quote.

Wait for the > prompt from TLK before typing the first line of the message. TLK will continue prompting for new lines with this character. Returned messages from the receiver end of the dialog will be displayed as lines interspersed with the initiator's lines (see the example that follows) until one or the other types a CTRL/Z in response to a > prompt.

Example:

The following example shows a typical dialog and how the conversation proceeds from the initiator's end.

```
TLK MAYNRD:: KB4:
>Sheldon--a call came long-distance; couldn't be transferred.
>
Oh. Who was it, did they leave a number to call back?
>Yes--Gladys Freen of Fern, Fern, Freen, and Fern. Call
>(415)987-6543 after 10:00 (1:00 PM your time).
>
OK--thanks again.  Sheldon.
<TLK> End of Dialog

Ready
```

(The dialog was terminated by the receiver in this example.)

2.2.2 Receiver's Interaction

If the target terminal is not busy (no one is logged in), the RSTS/E LSN program displays the first line of a dialog-mode message as follows:

<TLK> in Dialog with *source-node*:: *source-terminal*:
>
message-1

You can then continue the dialog by typing a return line in response to a > prompt.

If the receiving terminal is busy (someone is logged in at the terminal), the following display is made:

<TLK> in Dialog with *source-node*:: *source-terminal*:
Type 'TLK/*n source-node*:: *source-terminal*:' to continue dialog

The material in single quotes is what should be typed to continue the dialog. For example, TLK/6 NOD5::KB12:. LSN will then wait up to two minutes for the dialog to be continued. You can stop what you are doing at the terminal and type the indicated command to continue the dialog, or you can go to another terminal, log in, and type the indicated command to continue the dialog. The number *n* following the slash is the job number of the copy of the local LSN that is waiting for the response to continue the dialog. At least one space must be typed after the job number.

You actually run TLK when you type the command for continuing the dialog. TLK tells the waiting LSN utiltiy that the dialog will continue and

identifies the terminal from which it will continue. TLK then detaches from the terminal so that LSN can control the interactive dialog. TLK and LSN are separate jobs in this case. Both are terminated when the dialog is stopped by either user. When the dialog is complete, LSN kills itself and TLK reattaches to the terminal and exits.

You can also choose to continue the dialog with the RUN $TLK specification:

```
RUN $TLK
TLK V2.0 RSTS V7.1 Timesharing NSP
*/6 NOD5::KB12:
```

Once the line is typed, the first message is displayed and the dialog continues in the usual manner until either person types a CTRL/Z in response to a > prompt. LSN then breaks the connection and displays the following message:

```
<TLK> End of Dialog
```

If the receiving terminal is the local operator services console and the OPSER routine is running, the display for a busy terminal is made in the standard OPSER format:

```
MESSAGE nnnnn:date time JOB:nn jobname-ppn]
        <TLK> in Dialog with source-node:: source-terminal:
        Type 'TLK/n source-node:: source-terminal:' to continue dialog
```

Operation will then proceed as for a busy terminal.

Example:

The following example illustrates how the conversation from the previous example might have looked from the receiver's end. (The user is logged in, entering text with an editing program.)

```
          .
          .
          .
*I/And so, after/
<TLK> in Dialog with SANFRN:: KB5:
Type 'TLK/13 SANFRN:: KB5:' to continue dialog

*exit           \
                 \
                  \     (user exits from the editor
                  /           and starts TLK)
                 /
Ready           /
```

```
TLK/13 SANFRN::KB5:
>
Sheldon--a call came long-distance; couldn't be transferred.
>Oh. Who was it, did they leave a number to call back?
>
Yes--Gladys Freen of Fern, Fern, Freen, and Fern. Call
>
(415)987-6543 after 10:00 (1:00 PM your time).
>OK--thanks again. Sheldon.
>CTRL/
<TLK> End of Dialog
```

2.3 TLK Error Messages

Command syntax error – Illegal job number – /*n*

The job number you typed to continue a dialog is not a valid RSTS/E job number. Check the display that TLK printed and retype the line to continue the dialog.

Command syntax error – Illegal message

This error occurs if you try to type a message on the same line as the execute line for continuing a dialog. Retype the command line. When TLK prints the > prompt, you can then type the message you want sent.

Command syntax error – Illegal node name – *nodename*

The node name specified in a command line is not recognized by TLK. Retype the line with the correct node name.

Command syntax error – Illegal terminal specification – *spec*

The terminal specification shown (*spec*) is incorrect. Retype the command line.

Command syntax error – Job *n* is not a message receiver

The job number typed to continue a dialog is a valid job number but the job is not a message receiver. Check the display that TLK printed and retype the line to continue the dialog.

Command syntax error – Job *n* does not meet dialog requirements

The job number typed to continue a dialog is a valid job number, and the job is a message receiver, but the job is not the same one that initiated the dialog. Check the display that TLK printed and retype the line to continue the dialog.

Command syntax error – Message too long

The maximum line that TLK will accept is 132 characters. Retype the message with shorter lines.

Connect Initiate error – *text*

TLK tried to set up a logical link with the remote TLK or LSN to exchange messages, but it got an error on the initial connection request. The *text* is the error text TLK got in response to its attempt to send a Connect Initiate Message. (See *DECnet/E Network Programming in BASIC–PLUS and BASIC–PLUS–2* for further description of the error text.)

Connection rejected by NSP – Code:*n*

TLK tried to set up a logical link with the remote TLK or LSN to exchange messages but the connection was rejected by the remote NSP. The code *n* is the reason code that the remote NSP returned in its rejection. Appendix B lists the meanings for the various NSP error codes.

Connection rejected by remote TLK – Code:*n*

TLK tried to set up a logical link with the remote TLK or LSN to exchange messages but the connection was rejected by the remote TLK or LSN. The code *n* was returned in the rejection and its meaning depends on the remote TLK. Contact a DIGITAL Software Support representative if this error occurs repeatedly.

Connection rejected by remote TLK – Illegal value in TLK mode byte

TLK tried to set up a logical link with the remote TLK or LSN to exchange messages but the connection was rejected by the remote TLK or LSN. Contact a DIGITAL Software Support representative if this error occurs repeatedly.

Connection rejected by remote TLK – Terminal is busy

This error occurs when you have initiated a dialog and the remote terminal is busy, but no one has responded to continue the dialog. (The remote user is allowed two minutes to respond and continue the dialog; in this case, no one did.)

Connection rejected by remote TLK – Terminal is nonexistent

The remote terminal requested does not exist at the remote node. Check for possible error in the command line.

Connection rejected by remote TLK – *text*

The remote TLK or LSN could not accept the logical link connection that would allow the one-line or dialog exchange. The *text* is the error message that the remote TLK or LSN got in trying to issue its connect confirmation. If this error occurs often, contact a DIGITAL Software Support representative.

Connection rejected by remote TLK – Unknown format

The remote TLK did not recognize the parameters passed in the request to set up a logical link that would allow the one-line or dialog exchange. This could occur if the remote TLK is part of a Phase I implementation of DECnet – an earlier implementation than the local TLK/DECnet. Contact a DIGITAL Software Support representative.

Declare Receiver error – *text*

The local TLK could not begin to set up message exchange facilities at the local node. The *text* is the text of the error message received in its Declare Receiver system call.

DECnet is not installed

DECnet was not installed at system generation time. No network functions can be executed.

Dialog already continued by job *n*, account[*ppn*], from KB*n*:

You or someone else has already typed the line requesting a dialog continuation. The account number and terminal number can be used to determine who continued the dialog and from which terminal.

Disconnected by remote TLK – Code:*n*

The logical link allowing the one-line or dialog exchange has been disconnected by the remote TLK. The code *n* was returned by the remote TLK in its disconnection. Contact a DIGITAL Software Support representative if this error occurs repeatedly.

Disconnected by remote TLK – Transfer error

An error occurred at the remote site in transferring a message to or from the remote terminal. Contact a DIGITAL Software Support representative if this error occurs often.

Job *n* is not in dialog with specified target

In continuing a dialog, you typed the wrong terminal or node name. Check the display that TLK printed and retype the command line.

Link aborted by NSP – Code:*n*

The logical link allowing the one-line or dialog exchange has been aborted. The code*n* is the reason code supplied by NSP in the Link Abort Message terminating the link. Appendix B lists the meanings for the various NSP error codes.

Link could not be established – Timeout

Two minutes went by after the remote TLK printed its message for continuing a dialog and no one responded. The dialog was timed out.

Link could not be established – *text*

The local TLK got an error in receiving a Connect Confirm Message from the remote TLK (or LSN). The logical link that would provide the one-line or dialog exchange could not be set up. The *text* is the NSP error message that the local TLK got on its Receive system call. Contact a DIGITAL Software Support representative.

Message too long

A line typed in dialog mode was longer than 132 characters. Retype the message using shorter lines.

NSP is not enabled

NSP, the part of DECnet that would allow the logical link connections providing the one-line or dialog exchange, has not been enabled at the local node. See the system manager.

Receive error – *text*

The local TLK (or LSN) got an error in receiving a message from the remote end. The *text* is the error message returned on the Receive system call. Contact a DIGITAL Software Support representative.

Send data error – *text*

The local TLK (or LSN) got an error in sending a message to the remote end. The *text* is the error message returned on the Send Network Data Message system call. Refer to *DECnet/E Network Programming in BASIC–PLUS and BASIC–PLUS–2* for further details of the error text, and contact a DIGITAL Software Support representative.

Chapter 3
NET: Network Command Terminal Utility

The NET utility lets you access another DECnet/E system in the network and do work there as if your terminal were directly connected to the other system.

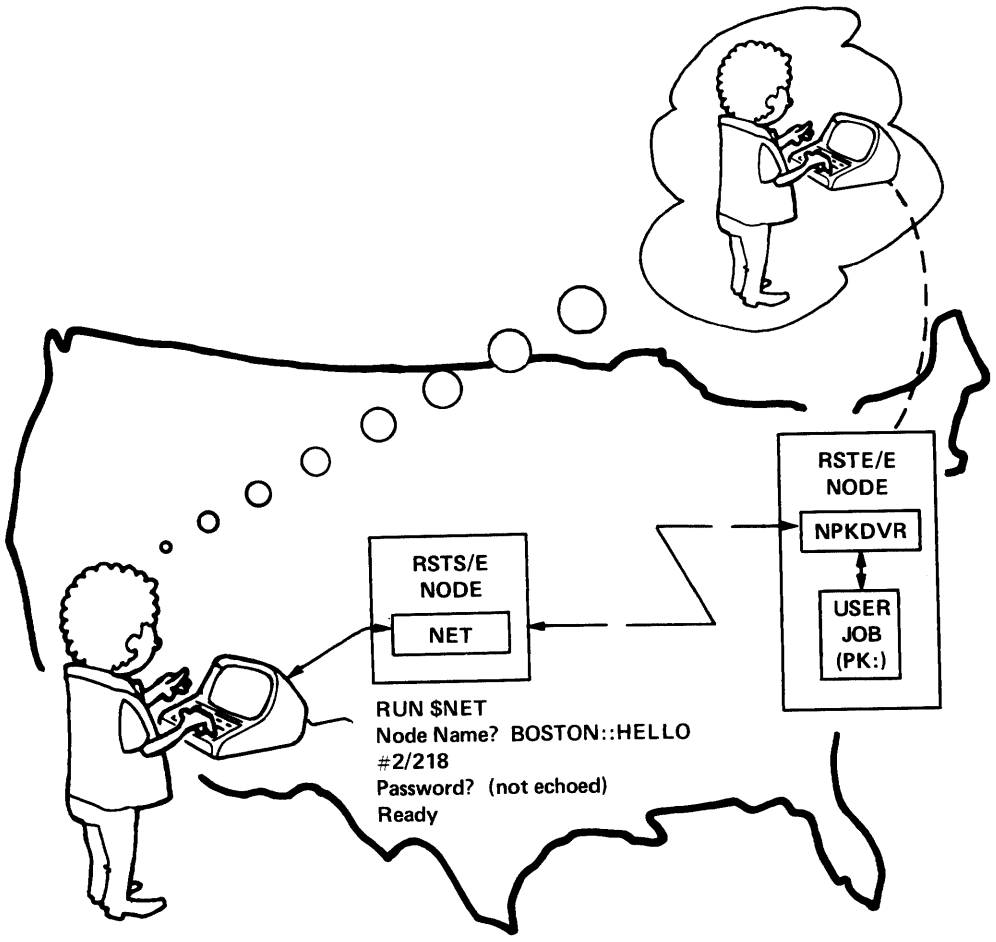


Figure 3–1 NET Provides Access to Another DECnet/E Node

NET is implemented as two programs: NET and NPKDVR. Figure 3–1 shows a user running NET at node DENVER. NET requests a logical link with NPKDVR at the remote node BOSTON. NPKDVR opens a pseudo-keyboard at node BOSTON and accepts the connection requested by NET. Once the connection is established, information that the user types at the

keyboard in DENVER is relayed over the logical link to NPKDVR, which relays it to the pseudo-keyboard. The RSTS/E system at BOSTON treats a pseudo-keyboard as a normal keyboard. When the RSTS/E monitor receives information from the pseudo-keyboard (whatever the user types at node DENVER), it creates a job and runs LOGIN to process the input. The responses made by the monitor, LOGIN, and whatever else the user runs, are directed to the pseudo-keyboard. NPKDVR, in turn, passes responses directed to the pseudo-keyboard back over the logical link to the NET program. NET then displays it at the user's terminal.

NOTE

Except for the special CTRL/P character and its related commands (discussed in the sections that follow), user input is passed directly through to NPKDVR on the remote system. NET makes no attempt to interpret any of the data it passes through. Thus, if the user types a system-level command, for example, the command is executed at the remote system and has no effect on the local system.

NPKDVR is itself a job. To reduce overhead, a system manager can install one or more permanent copies of NPKDVR at a node. Each such permanently installed copy can handle multiple logical links. When such copies are all busy, an incoming connection request causes a temporary copy to be automatically started. (See the discussion of the SET/DEFINE OBJECT command in *DECnet/E System Manager's Guide* for further information on installing programs for automatic startup.)

3.1 Running NET

You can run NET from any terminal in a DECnet/E system by typing the following command:

```
RUN $NET
```

NET responds by identifying itself and then outputing the following prompt:

```
Node Name?
```

Type a node name, followed by two colons and, optionally, an initial string of characters to be forwarded to the remote node. When the connection is established, NET prints the following message:

```
Connection established to RSTS/E Node nodename
```

You can then proceed as though directly connected to the remote RSTS/E node.

If NET has been installed as a Concise Command Language (CCL) command, you can also run NET by typing the following command:

```
NET [nodename[::[string]]]
```

If you are running under the DCL Run-Time System, you can also connect to a remote node by typing the following command:

SET HOST [*nodename*[::[*string*]]]

Once the connection to the remote DECnet/E node is established, you are logically connected to the remote node. You can log in, run programs (including NET to establish a connection at yet another DECnet/E node), and log out. You can use the special character CTRL/P (Control-P) followed by the RETURN key to escape to the NET program at the local node without breaking the connection to the remote node. NET will then accept one of the commands described in Section 3.2.

NOTE

To run a program at the remote node that opens the keyboard using field control, echo control, or binary modes (as described in the *RSTS/E Programming Manual*), you must first use the ODT command to enter a special transmission mode (Section 3.2.6). This places a heavy load on the network, however, and should only be used as needed.

3.2 NET Commands

If you type CTRL/P, followed by RETURN, while using a remote RSTS/E node, you will temporarily return to the NET program at the local node. The connection to the remote node is not broken, but any characters you type will be processed by NET at the local node rather than being forwarded to the remote node. NET displays the following prompt:

nodename::NET>

The parameter *nodename* is the name of the "local" node. (If you are using NET to run NET at another remote node, this is the name of the node at which the remote NET is processing a CTRL/P command. See Section 3.2.2.) You can then type one of the following six commands. These commands are listed below and described in the sections that follow:

CONTINUE	Continue transmission of input to remote node.
CTRL/P	Transmit a Control-P character to the remote node
EXIT	Terminate transmission to the remote node and exit from the NET program.
HELP	Display instructional information.
NORMAL	Resume normal transmission to remote node, one line at a time.
ODT	Begin transmission to remote node, one character at a time.

3.2.1 CONTINUE Command

This command causes the NET program to stop displaying its "NET>" prompt and resume sending the data typed at the local terminal over the connection to the remote node.

Example:

```
RUN $NET
NET V2.0 RSTS V7.1 DECnet System
Node name? BOSTON
Connection established to RSTS/E Node BOSTON
HELLO
RSTS V7.1 Timesharing Job 31 KB33 15-Aug-81 8:00AM
#2/218
Password: (not echoed)


Ready


^P
DENVER::NET>HELP
          .
          .
          .
(help display)
          .
          .
          .
DENVER::NET>CONTINUE
RUN $DIRECTORY
          .
          .
(DIRECTORY display for user
      at node BOSTON)
          .
          .
          .
```

3.2.2 CTRL/P Command

The CTRL/P command is used to pass on a Control-P character to the remote node. This is useful when you are running NET at one remote node to connect to yet another node.

Example:

Consider a network having three DECnet/E nodes named DENVER, BOSTON, and TULSA, with BOSTON as the central node. A user at node DENVER runs NET as follows:

```
RUN $NET
Node Name? BOSTON::HELLO
Connection established to RSTS/E Node BOSTON
RSTS V7.1 Timesharing Job 31 KB33 15-Aug-81 8:00AM
#2/218
Password: (not echoed)

Ready
```

```
RUN $NET
Node Name? TULSA::HELLO
Connection established to RSTS/E Node TULSA
RSTS V7.1 Timesharing Job 31 KB33 15-Aug-81 9:01AM
#1/150
Password: (not echoed)


Ready


RUN $ODT
#^P
DENVER::NET>CTRL/P
BOSTON::NET>ODT
^P
DENVER::NET>ODT
          .
          .
          .
```
(TULSA's ODT waiting for input)

In this example, the user runs NET at the local node (DENVER) and establishes a connection with node BOSTON. At node BOSTON, the user again runs NET, establishes a connection with node TULSA, and runs ODT at TULSA.

When ODT displays its prompt, the user types a CTRL/P combination followed by a RETURN, escaping to the local NET's command level. The user then types the command CTRL/P, which sends a Control-P character to node BOSTON, causing BOSTON's NET to display its "NET>" prompt. The user types ODT, telling BOSTON's NET to send information to TULSA in the single-character mode. The local NET must still be told to send information in ODT mode, so the user again types a CTRL/P combination followed by a RETURN, escaping to the local NET's command level. The user then types the ODT command and is then ready to type commands to the ODT running at TULSA.

NOTE

Note that this command consists of the actual character string "CTRL/P" rather than the CTRL and P key combination.

3.2.3 EXIT Command

There are two ways to terminate the NET program. You can simply log out at the remote node, or you can use the EXIT command. The EXIT command severs the connection with the remote node. NPKDVR at the remote node automatically kills the job to which the user is connected. NET displays the following message:

Link to Node *nodename* disconnected

This message is followed by two bell characters (an audible noise at your terminal). NET then exits.

Example:

```
RUN $NET
Node Name? BOSTON::
Connection established to RSTS/E Node BOSTON
HELLO
RSTS V7.1 Timesharing Job 31 KB33 15-Aug-81 8:00AM
#2/218
Password: (not echoed)

Ready (node BOSTON's Ready prompt)

RUN $NET
Node Name? TULSA::HELLO
Connection established to RSTS/E Node TULSA
RSTS V7.1 Timesharing Job 22 KB17 15-Aug-81 9:01AM
#2,120
Password: (not echoed)

Ready (node TULSA's Ready prompt)

RUN $SYSTAT

(systat display)

Ready (node TULSA's Ready prompt)

BYE/F

Link to node TULSA disconnected

Ready (node BOSTON's Ready prompt)

^P
DENVER::NET>EXIT
Link to node BOSTON disconnected

Ready (node DENVER's Ready prompt)
```

In this example, the user simply logs off at node TULSA and uses the EXIT command to break the link to node BOSTON.

3.2.4 HELP Command

If you type HELP in response to the "NET>" prompt, you will get a help display at the terminal. After the help information is displayed, NET displays another prompt and you can type another NET command.

Example:

```
RUN $NET
NET V2.0 RSTS V7.1 DECnet System
Node name? BOSTON::HELLO
Connection established to RSTS/E Node BOSTON
RSTS V7.1 Timesharing Job 25 KB27 14-Aug-81 10:23AM
#2/218
Password: (not echoed)


Ready


          .
          .
          .
(work at node BOSTON)
          .
          .
          .
^P
DENVER::NET>HELP
          .
          .
          .
(help display)
          .
          .
          .
DENVER::NET>
```

In the preceding example, the user runs NET at the local node (DENVER), requesting a connection to node BOSTON. After the connection is made, the user types a standard log-in sequence, the RSTS system at BOSTON displays its identifying line and "Ready" prompt, and the user does whatever work there he wishes. He then types CTRL/P followed by RETURN to return to the local NET program to display the HELP text. After the display, NET displays a prompt for another command.

3.2.5 NORMAL Command

The NORMAL command is used to stop the single-character input requested by the ODT command. It should be used immediately after exiting from a program that needs this kind of input.

Example:

```
(ODT processing at remote node)
          .
          .
          .
#^Z

Ready

^P
DENVER::NET>NORMAL
          .
          .
          .
(continue work at node BOSTON)
```

In the preceding example, the user terminates ODT at node BOSTON by typing CTRL/Z (displayed as ^Z on the terminal), and the BOSTON RSTS/E system displays its "Ready" prompt. The user then types CTRL/P to return to the local NET program at node DENVER and NET displays its "NET>" prompt. (It is not necessary to type the RETURN key after CTRL/P in this case. Since the NET program is forwarding each character as it is typed, it recognizes the CTRL/P immediately.) The user types NORMAL to stop ODT-mode transfers and returns to work at node BOSTON.

3.2.6 ODT Command

Normally, RSTS/E programs receiving input from terminals accept an entire line at a time – a line being a string of characters terminated by a carriage-return/line-feed combination, an escape character, and so forth. A few programs, such as ODT (Octal Debugging Tool), do not wait for an entire line but process characters as they are typed. When you run such a program at the remote node, you must first issue the ODT command to the NET program at the local node. NET will then forward characters to the remote node as you type them.

NOTE

Using this feature places a heavy load on the network since DECnet must now send many more messages over the link than it would normally. ODT mode should be used with discretion.

Example:

```
NET BOSTON::HELLO
Connection established to RSTS/E Node BOSTON
RSTS V7.1 Timesharing Job 31 KB33 15-Aug-81 8:00AM
#2/218
Password: (not echoed)

Ready
```

```
RUN $ODT
#^P
DENVER::NET>ODT
210/ 1077
#
          .
          .
          .
```

In the preceding example, the user runs NET as a CCL command, requesting a connection to node BOSTON. The string "HELLO" is given as part of the command line. Once the connection is accepted, the RSTS/E system displays its normal prompt for a project-programmer number and the user proceeds with logging in. The user then runs ODT at node BOSTON and when ODT prints its pound sign prompt for input, the user types the CTRL/P combination followed by RETURN to return to the NET command level at the local node (DENVER). NET responds with its "NET>" prompt and the user types ODT. NET processes the command and returns the user to the remote node. (ODT was waiting for input.) The user types an ODT command and processing continues as normal.

3.3 NET Error Messages

?Connect Failure – Invalid node name *nodename*

The node name given in response to the "Node name?" prompt is not valid. There is no such node in the network. Run NET again, specifying a valid node name.

?Connect Failure – Node shutting down

The node requested is scheduled for shutdown. No new logical links can be established to the node. Try again later.

?Connect Failure – Too many links to node *nodename*

Network traffic to node *nodename* is too heavy at the moment to establish another logical link. Try again later.

?Connect Reject – NSP reason code = *n*

The remote node rejected NET's request for a logical link. NSP reason codes are given in Appendix B.

Illegal Command

The command you typed in response to a "NET>" prompt is not one of those recognized by the NET utility. Type HELP to see a display of valid commands.

Chapter 4
NFT: Network File Transfer Utility

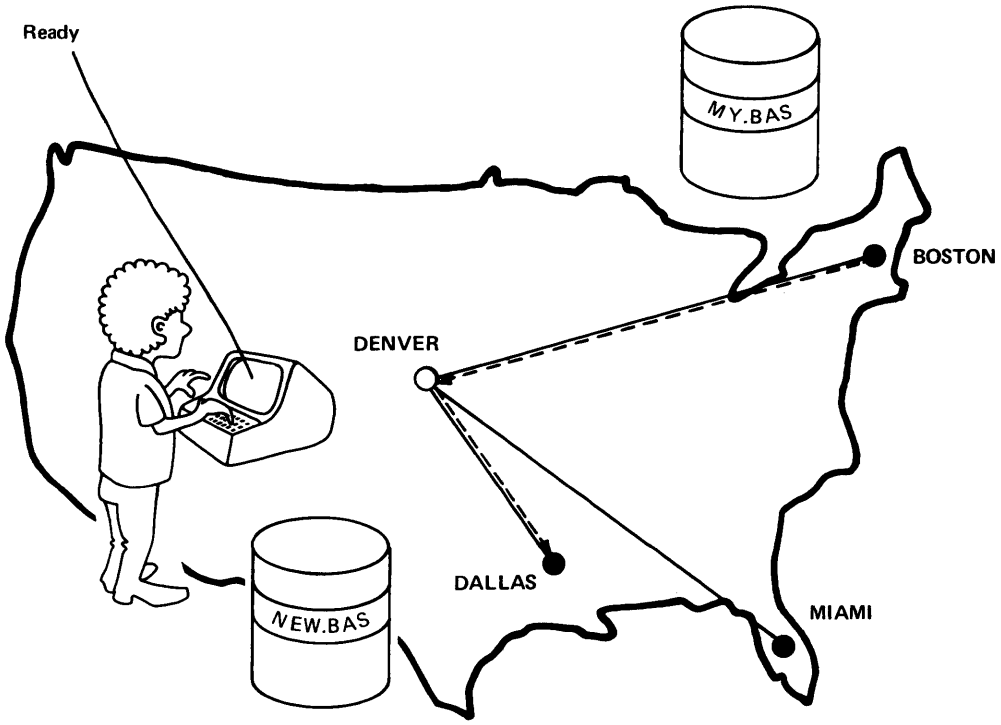
As the name implies, the main function of the Network File Transfer utility (NFT) is to move files from one node to another (see Figure 4–1). You can run NFT from a terminal to perform the following functions:

- Copy files from a storage device at one node to a storage device or output device at another node.
- Copy and append one or more files stored at one node to a file stored at another node.
- Copy and submit a command file stored at one node to another node's batch processor or indirect command interpreter for execution. On RSTS/E nodes, such files are submitted to the batch processor.
- Execute an existing command file at a remote node.
- Delete files stored on a disk at a remote node.
- Copy and spool print files from one node to another node's printer
- Obtain a directory listing of files stored on disk at a remote node.

NOTE

Even though NFT is written to conform to the Data Access Protocol (DAP), Version 5.6, it still supports only the executable bit during file transfers; file protection codes are not passed. Thus, files created as a result of NFT commands have a protection code of either <60> or <124>, depending on the presence of the executable bit.

```
RUN $NFT
NFT>COPY DALLAS::NEW.BAS=BOSTON::MY.BAS
NODE:                BOSTON
PPN:                 120,123
PASSWORD:            (not echoed)
ACCOUNT:             (none needed)
NODE:                DALLAS
PPN:                 104,160
PASSWORD:            (not echoed)
ACCOUNT:             (none needed)
NFT>
```



NFT provides file protection by prompting for log-in information for remote nodes.

Figure 4–1 NFT Transfers Files to, from, and between Remote Nodes

4.1 Background

It is easier to understand what NFT does by beginning with a brief description of how the file transfer capability is implemented in DECnet/E.

The features listed in the introduction to this chapter are actually provided by three software modules – NFT, the File Access Listener (FAL), and the Record Management Services (RMS).

NFT establishes a logical link with a FAL program at the remote node. It then accepts commands typed by the local user and transmits them to FAL. The FAL program, a part of DECnet, does whatever is necessary on its end to execute the commands. NFT and FAL communicate with each other in a language defined by the Data Access Protocol (DAP) for DECnet in general. Thus, even though the user commands can be different for various DECnet implementations of NFT, all DECnet systems having NFT and FAL can communicate to do file handling.

RMS is a file service feature used by NFT and FAL at DECnet/E nodes to execute input and output to the local I/O devices. This is of interest mainly because non-RSTS/E nodes can use other means to do their local file I/O and can require some of the special switches described in Section 4.5. These switches ensure that files transferred between nodes with different operating systems and file systems are transferred in a usable format. Figure 4–2 illustrates the interaction between NFT, FAL, and RMS at two DECnet/E nodes.

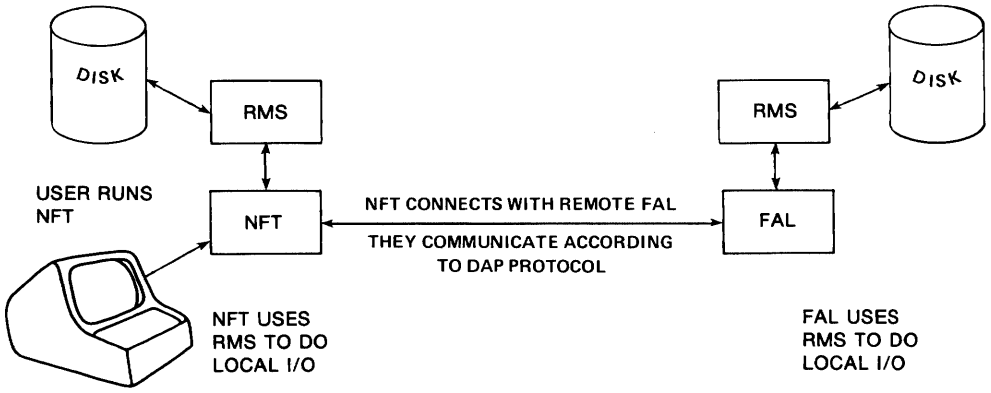


Figure 4–2 At DECnet/E Nodes, NFT, FAL, and RMS Work Together to Allow File Operations

4.2 Running NFT

You can run NFT from any terminal in a DECnet/E system by typing:

```
RUN $NFT
```

Or, if NFT has been installed by the system manager as a Concise Command Language (CCL) command, you can simply type:

```
NFT
```

In either case, NFT responds with the prompt NFT> and will accept one command per line until CTRL/Z or "EXIT" is typed in response to a prompt:

```
NFT> command-1
NFT> command-2
  .
  .
  .
NFT> command-n
NFT> ^Z
```

If NFT has been installed as a CCL command and you want to use only one NFT command, you can type NFT and the command on one line, as follows:

```
NFT command
```

NFT executes the command and immediately returns control to the RSTS/E monitor.

NOTE

If you are running under the DCL Run-Time System, the NFT functions can be invoked by issuing the appropriate DCL command (e.g., COPY, DELETE, PRINT, etc.). Since the syntax of these DCL commands differs significantly from the NFT commands described in this chapter, refer to the *RSTS/E DCL User's Guide* for details on using the DCL commands.

To protect files from unauthorized use or destruction, an NFT user must supply log-in information for any node referred to in a command. There are five methods for supplying this information:

1. You can use the NETACT utility to provide the information in a file that NFT can use each time it is run (see Appendix C).
2. You can use the NODESPECIFICATION command (see Section 4.4.7).
3. You can supply the information as an extension of the remote node specification (see Section 4.3.1).
4. You can use the /IDENTIFY switch to request that NFT prompt for the information on a one-time basis (see Section 4.5.8).

5. Or, you can simply do nothing and force NFT to prompt for the information. (NFT will prompt you if you refer to a remote node in a command and have not specified the log-in information in any other fashion.)

In three of these cases (NETACT file, NODESPECIFICATION command, and NFT prompting), once log-in information is specified for a particular node, it is used throughout the NFT run each time that node is referenced, unless it is changed, overridden, or rejected by the remote node. In the other two cases (node name extension and the /IDENTIFY switch), the information is used to override the existing log-in information for one command only.

However specified, NFT uses the log-in information to gain access to the remote file system before executing any command affecting a file at a remote node. If the information given is valid, you can access any files that you could when directly logged in to the remote system under the specified account.

4.3 General Format of NFT Commands

A DECnet/E NFT command consists of a command keyword (defining an action) and variable parameters (defining the nodes and files involved in the action). In addition, switches can be used when necessary to further define the action to be performed. For example:

```
COPY DENVER::FILE.LST/SU=DALLAS::MYFILE.TXT
```

COPY is the command keyword, indicating that the file on the right of the equal sign (MYFILE.TXT at node DALLAS) is to be copied to the node named DENVER under the name FILE.LST. The /SU switch indicates that if a file already exists on node DENVER called FILE.LST, it will be superseded by the file being transferred.

Command and switch keywords can be abbreviated. In general, the first two characters of the keyword are all that is necessary. Some switches, however, require four characters to identify the switch. For example: /NOHEADING, /NOSUPERSEDE, and /NODELETE.

The commands and switches recognized by the DECnet/E implementation of NFT are described in Sections 4.4 and 4.5. Most of the user-supplied arguments identify files to be manipulated by NFT. The following general comments apply.

4.3.1 Node Names

DECnet/E NFT can be used to manipulate files to, from, and between remote nodes. NFT will also work with files at the local node, although it is generally more efficient to use the RSTS/E utilities designed to manipulate files at the local node.

The network shown in Figure 4–3 will be used as an example throughout this chapter. To illustrate file transfers among DECnet nodes with different operating systems, BOSTON and DENVER are DECnet/E nodes (RSTS/E operating system); DALLAS is a DECnet/11M node (RSX–11M operating system); and MIAMI is a DECnet/RT node (RT–11 operating system).

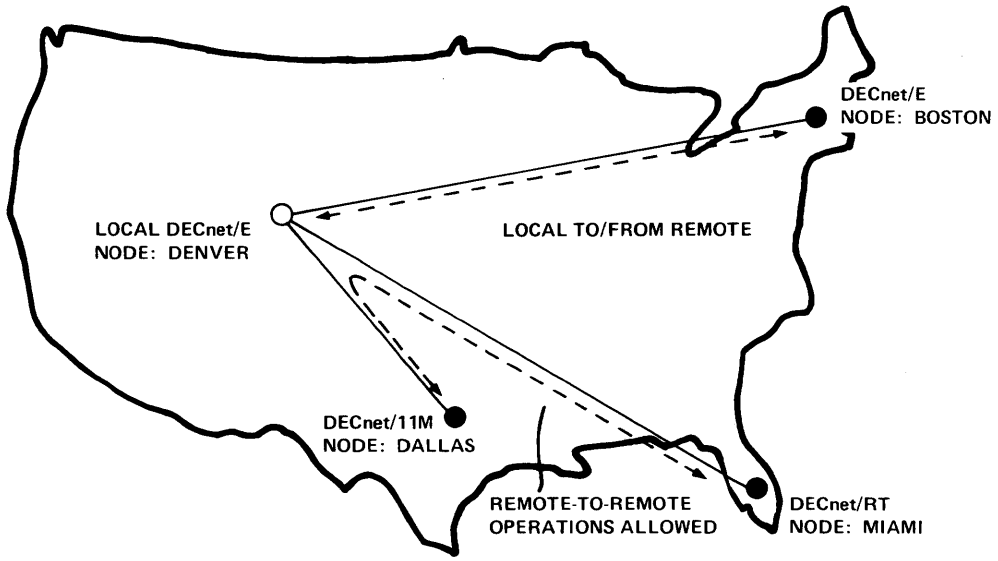


Figure 4–3 Sample Network Showing Transfers between Nodes with DECnet/E NFT

To identify the remote node where a particular file is located or to which it is to be moved, the 1- to 6-character, alphanumeric node name is given. The node name must contain at least one letter and is followed by two colons (::) to separate it from the file specification. (Names of currently active nodes in the network can be determined with the NCP command SHOW ACTIVE NODES, described in the *DECnet/E System Manager's Guide.*) If the local node name is specified in an NFT command, FAL is run at the local node.

As mentioned in Section 4.3.1, the log-in information required by NFT to access files at the remote node can be specified as an extension of the node name. If included, this information must be in the following format:

nodename "[*user-id*] [*password*] [*account-id*]"::

The parameters *user-id*, *password*, and *account-id* are the same as described with the NODESPECIFICATION command (see Section 4.4.7). They are separated by either a space or a horizontal tab character and are enclosed in double quotes.

As with quoted strings (see Section 4.3.4), NFT simply removes the quotes and passes the characters they enclose to the remote system without examination. In some circumstances, this means that the accounting information

is processed differently than if it had been specified with the NODESPECIFICATION command, through the NETACT program, or with prompting. For example, consider the case where the remote system is running the VAX/VMS operating system. Since VMS does not permit nonoctal user identification codes, NFT ordinarily converts the specified *user-id* to the VMS-compatible octal equivalent. However, if the log-in information is included with the node name, enclosed in quotes, no conversion is done and errors may result.

4.3.2 File Specifications

The way in which files are identified depends on the operating system at the node where the file is located. For file specification formats used with non-RSTS/E nodes, consult the documentation for the appropriate operating system.

For files at RSTS/E nodes, the file specification is a subset of the full RSTS/E file specification. The elements used (device, project-programmer number, file name, and type) follow the rules for RSTS/E file specification.

dev:[*ppn*]*filename.typ*

The *dev* item is a device designation. If this is omitted, the system disk (SY:) is assumed.

Although you can refer to specific devices at the local or remote nodes, you are responsible for seeing that tapes are mounted and positioned properly, and so forth. If a device is busy, NFT will return an error message. You can use the ASSIGN and DEASSIGN commands at the local node to reserve local devices.

NOTE

It is currently more practical to use NFT to move files between disk structures (either public or normally mounted private disks) at each node than to request transfer to or from another type of device. For example, suppose you are using NFT to transfer a file to the local node for transfer to a magnetic tape. Use NFT to transfer the file to public disk and then use PIP to transfer the file to magtape. (The *RSTS/E System User's Guide* describes how to use PIP.)

The *ppn* is the project-programmer number identifying the directory in which the file is located – for example, [100,201]. (On some DIGITAL systems, the *ppn* is called a user identification code, or UIC.) If omitted, the *ppn* is assumed to be the user ID specified during the NFT log-in sequence (with the NODESPECIFICATION command, NFT's prompting, or through the NETACT utility) for the node being referenced. Access to another *ppn* depends on the user's privilege at the node in question. (Note that some DIGITAL systems permit alphanumeric directory names to be assigned

equivalent to the project-programmer number. If the remote system permits such names, they can be used in the file specification.)

NOTE

If a *ppn* is given, the square brackets (or parentheses) must be used.

The *filename.typ* is the file name and type. For files on RSTS/E nodes, the *filename* can be up to 6 alphanumeric characters, and *typ* can be up to 3 alphanumeric characters. If a command accepts both an input and an output file specification and the output specification is omitted, NFT will default the output file specification to that supplied for the input file. If *typ* is omitted, NFT does not supply a default.

4.3.3 Wildcards

NFT permits wildcard characters in file specifications, allowing the user to select a set of files for sequential file retrieval, file deletion, file spooling, directory listings, or command file execution. The format of the wildcard syntax depends on the capabilities of the remote node to process the file specification.

In an input file specification, any valid wildcard that is acceptable to the remote or local system is acceptable. For example, if the remote system is a RSTS/E system, an input file specification of A?????.LST selects as input all those files with an type of .LST and a file name beginning with the letter A. The *ppn* in an input specification can be entered as either [*], for named directories, or [*,*], for numeric project-programmer numbers.

The only valid wildcard character supported for output, however, is the asterisk (*). An asterisk can be used to replace one entity in the file specification. Thus, "A?????.LST" in the previous example is invalid as an output file specification but "*.LST" is permitted.

When wildcards appear in the output file specification of certain commands, names are duplicated on a one-to-one basis during the operation as specified by the wildcards. For example, consider the following COPY commands:

```
COPY *.DAT=DALLAS::[COWBOY]*.LST
COPY *.DAT=DENVER::[1,34]FILEA.LST,[3,12]FILEB.LST
```

On receiving the first command, NFT copies every file from the directory COWBOY on node DALLAS with the type .LST to a file on the local node in the current directory with the same file name but with a type of .DAT.

The second command causes NFT to copy FILEA.LST from directory [1,34] on node DENVER to FILEA.DAT in the current directory on the local node and FILEB.LST from directory [3,12] on node DENVER to FILEB.DAT, also in the current directory on the local node.

In most cases, when the input file specification contains wildcards but the output file specification contains none, the input files are all concatenated into the one output file. For example, consider the first example shown previously, slightly modified:

```
COPY MASTER.DAT=DALLAS::[COWBOY]*.LST
```

This command causes NFT to copy every file with the type .LST from directory COWBOY on node DALLAS to a single file MASTER.DAT in the current directory on the local node. Each of the files located with the type .LST is appended, in sequence, into a single output file named MASTER.DAT.

In this circumstance, as with the APPEND command (see Section 4.4.1) and the /APPEND switch (see Section 4.5.1), the user must be aware of the attributes of the individual source files. Concatenating an image file with an ASCII file, for example, can have unpredictable results, depending on how the final destination file is to be used; an error message may or may not be generated. Thus, when using wildcards in the source file specification in this fashion, it can be useful to apply the /INQUIRY switch (see Section 4.5.10). As each input file is located, the user can then make a decision as to whether or not the file should be included in the operation.

4.3.4 Quoted Strings

Quote characters can be used in commands to tell the local NFT not to examine the characters enclosed by the quotes. In general, when NFT encounters paired double quotes ("text") or single quotes ('text'), it removes the quotes and passes the characters they enclose to the remote system without examination. This is useful when it is necessary to include characters in a file specification for a remote, non-RSTS system that the local NFT recognizes in its own analysis of a command.

For example, suppose you want to copy a file to a DECnet/RT node and you want to use the RT–11 file system's switch for allocating a specific amount of storage space for the file. This switch is of the form /B:*n*. To prevent the local NFT from interpretting the slash and trying to parse the switch as one of its own (Section 4.5), you can enclose the file specification in quotes:

```
COPY RTNODE::"FILE.LST/B:200"=LOCFIL.1
```

The local (RSTS/E) NFT also treats pairs of square brackets – [ ] – parentheses – ( ) – and angle brackets – < > – in a similar manner. It does not remove these pairs, however, but passes them on, along with the characters they enclose, without examination. These characters can be used for access to either the local or a remote node. (At a DECnet/E node, local or remote, the characters enclosed would be passed on directly to RMS.)

4.4 NFT Commands

There are twelve NFT commands available. Each command is discussed in detail in the following subsections. The commands available are:

APPEND	Append file(s) from one node to a file on another.
COPY	Copy file(s) from one node to another.
DELETE	Delete file(s) at a remote node.
DIRECTORY	Obtain a directory listing of files at a remote node.
EXIT	Exit from NFT.
HELP	Obtain instruction information.
NODESPECIFICATION	Enter remote node log-in information.
PRINT	Spool file(s) from one node to the line printer of another node.
SUBMIT	Submit file(s) from one node for execution on another node.
TYPE	Display file(s) on the terminal console.
VERSION	Display the local NFT or remote FAL version number.

4.4.1 APPEND Command

The APPEND command transfers one or more files from one node to another, appending them to the end of an existing file at the target node.

NOTE

When invoking this command, the user should be aware of the attributes of the files to be appended and ensure that the files are compatible. Appending a variable-length record file to one with fixed-length records, for example, can have unpredictable results. An error message may or may not be output.

Command Format:

APPEND [*dstnode*::]*outfile* = [*srcnode*::]*file1*[,...,*filen*]

Parameter Descriptions:

dstnode

The *dstnode* is the name of the node containing the file to which the other files are to be appended. If omitted, the local node is assumed.

outfile

The *outfile* is the file specification of the existing file to which the others are to be appended.

=

The equal sign separates the destination and source specifications.

srcnode

The *srcnode* is the name of the node from which the files are to be copied. If omitted, the local node is assumed.

file1,...

These are the file specifications for the files to be transferred and appended. They will be appended to the end of *outfile*, in the order specified.

Example:

```
NFT> APPEND DALLAS::ACCT.BAS=AACCT.BAS,BACCT.BAS
```

In this example, the files AACCT.BAS and BACCT.BAS are copied from the local node and appended to the file ACCT.BAS at node DALLAS (Figure 4–4).

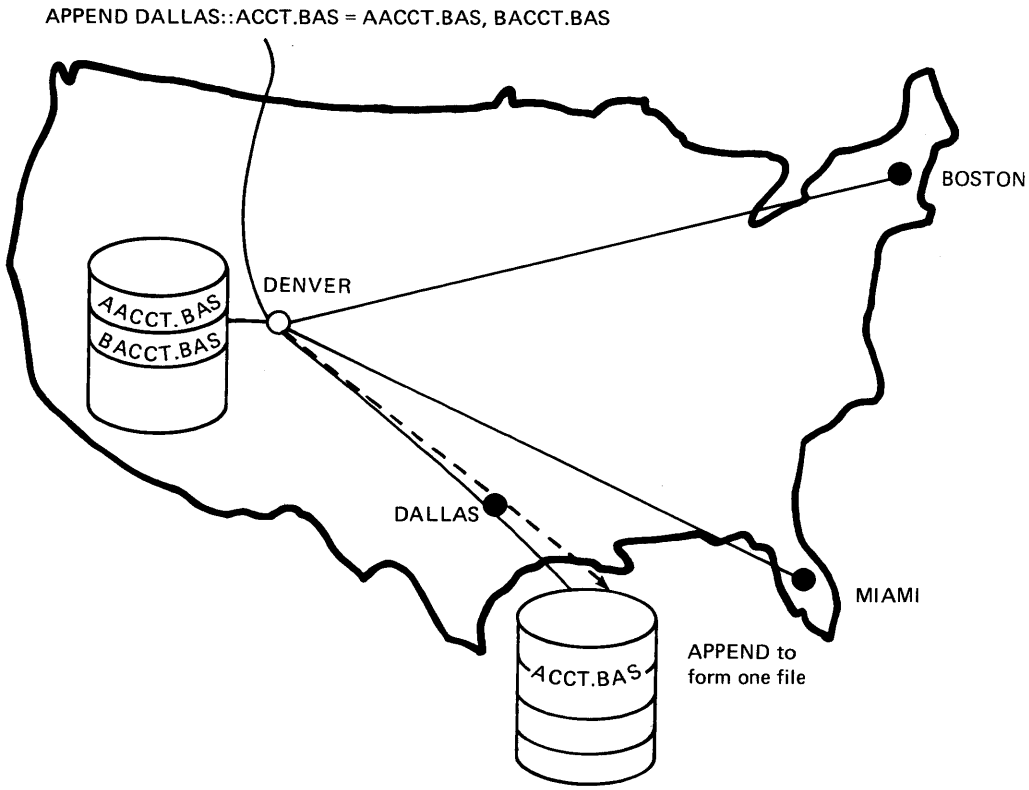


Figure 4–4 Example of the APPEND Command

4.4.2 COPY Command

The COPY command copies one or more files from one node to one or more files at another node. The files are not deleted at the source node. COPY is the default command for NFT; the command keyword COPY can be omitted entirely.

Command Format:

[COPY] [*dstnode*::][*outfile*][=][*srcnode*::]*file1*[,...,*filen*]

Parameter Descriptions:

dstnode

The *dstnode* is the name of the destination node to which the file(s) are to be transferred. If omitted, the local node is assumed.

outfile

The *outfile* is the file specification to be given to the transferred file(s) at the destination node. If no wildcards are used, the input files are concatenated in the order specified to form *outfile*. If wildcards are used, each output file created at the destination node will have the *outfile* specification, as specified by the wildcards in the input file names.

=

The equal sign separates the destination and source specifications.

srcnode

The *srcnode* is the name of the source node from which the file(s) are to be copied. If omitted, the local node is assumed.

file1,...

These are the file specifications for the files stored at the source node that are to be copied.

The equal sign is only required if a *dstnode* or *outfile* specification is included in the command. If all three items are omitted, the input files (*filen*) will be copied to the user's keyboard console. If *dstnode* and *outfile* are omitted but the equal sign is not, the output file is assumed to be "*.*".

Example:

```
NFT> AFIL.BAS=BOSTON::AFIL.BAS
NFT> BFIL.BAS=BOSTON::BFIL.BAS
NFT> CO DK1:MYFILE.BAS=BOSTON::AFIL.BAS,BFIL.BAS
NFT> COPY DALLAS::NEWFIL.BAS=BOSTON::AFIL.BAS,BFIL.BAS
```

In all these examples, files AFIL.BAS and BFIL.BAS are copied from the node named BOSTON (see Figure 4–5). In the first two commands, the files are copied to the local node's system disk as separate files named AFIL.BAS and BFIL.BAS, the same names they had at node BOSTON. In the third command, they are copied and concatenated to form a file named MYFILE.BAS on disk DK1 at the local node. In the fourth command, they are copied to form a file named NEWFIL.BAS on the public disk at node DALLAS.

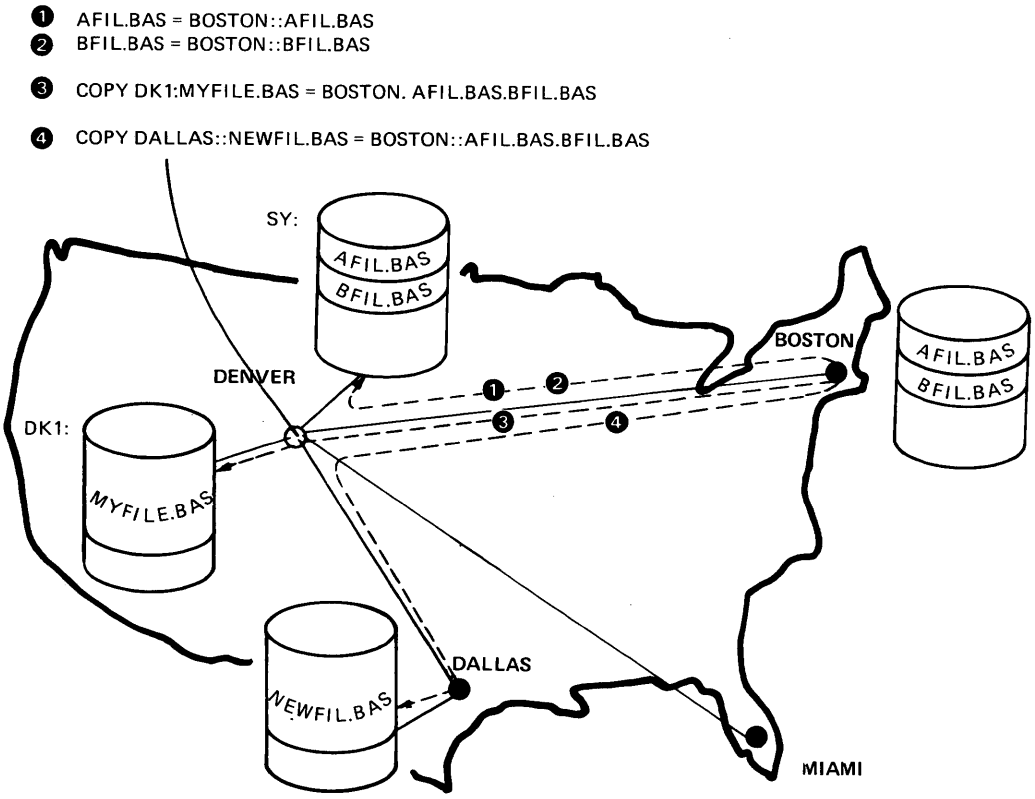


Figure 4–5 Examples of the COPY Command

4.4.3 DELETE Command

The DELETE command deletes the specified files. These files must exist on disk; tape files cannot be deleted with this command.

Command Format:

DELETE [*dstnode*::]*file1*[,...,*filen*]

Parameter Descriptions:

dstnode

The *dstnode* is the name of the remote node at which the files to be deleted are located. If omitted, the local node is assumed.

file1,...

These are the file specifications for those files to be deleted. These files must reside on disk; no tape file deletions are allowed.

Example:

```
NFT> DELETE MIAMI::FIL.RNO,BINFIL.SAV,RM.BAS
```

The files FIL.RNO, BINFIL.SAV, and RM.BAS are deleted from the public disk at node MIAMI (see Figure 4–6).

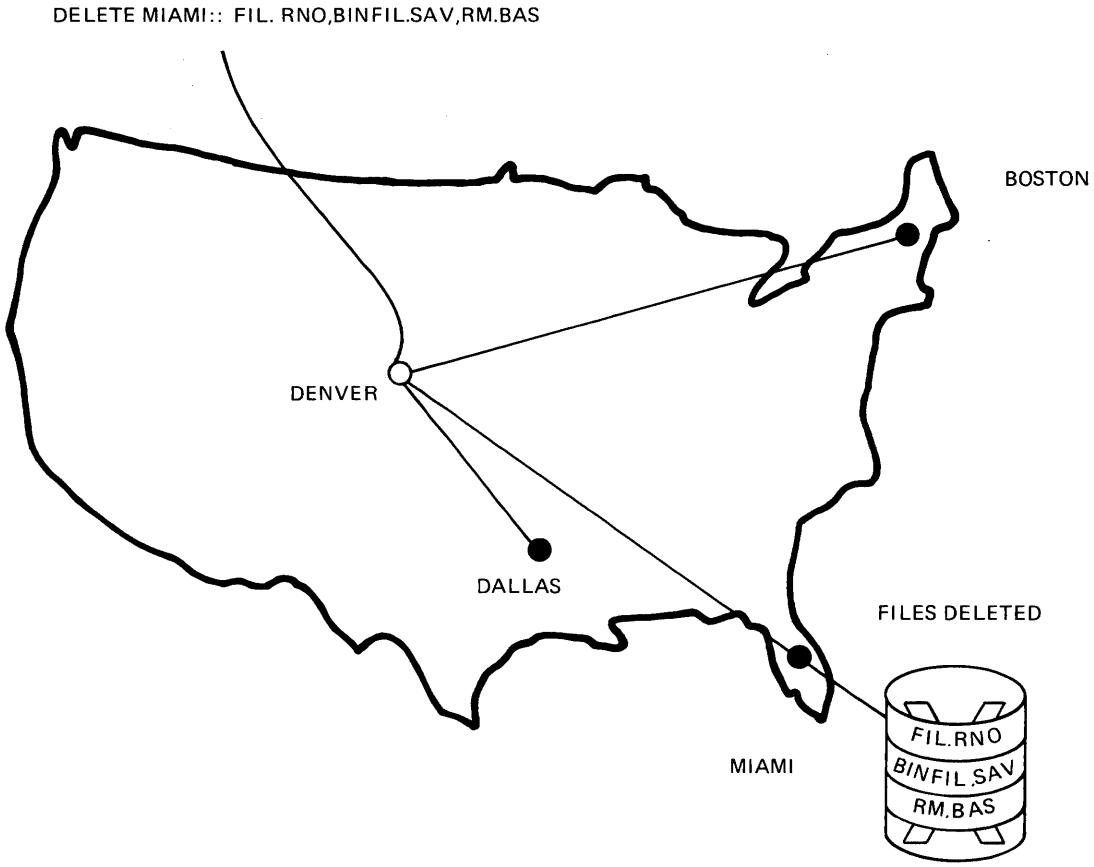


Figure 4–6 Example of the DELETE Command

4.4.4 DIRECTORY Command

The DIRECTORY command displays a directory listing of files located at a remote node.

Command Format:

DIRECTORY [*dstnode*::][*outfile*][=][*srcnode*::]*file1*[,...,*filen*]

Parameter Descriptions:

dstnode

The *dstnode* is the name of the destination node on which the directory listing is to be created. If omitted, the local node is assumed.

outfile

The *outfile* is the specification of the file to receive the output listing. If it is omitted, the listing is output to the console on the destination node. If the file currently exists, its contents are overwritten unless the /APPEND switch is used (see Section 4.5.1).

=

The equal sign separates the destination and source specifications.

srcnode

The *srcnode* is the name of the node at which the files to be listed are located.

file1,...

These are the specifications of the files to be listed. They must exist at the source node.

The equal sign is only required if a *dstnode* or *outfile* specification is included in the command. If all three items are omitted, the directory listing is output to the user's console.

Example:

```
DIRECT DENVER::[7,214]*.*
```

A complete listing of all files in account [7,214] on the system disk of node DENVER is displayed at the local console.

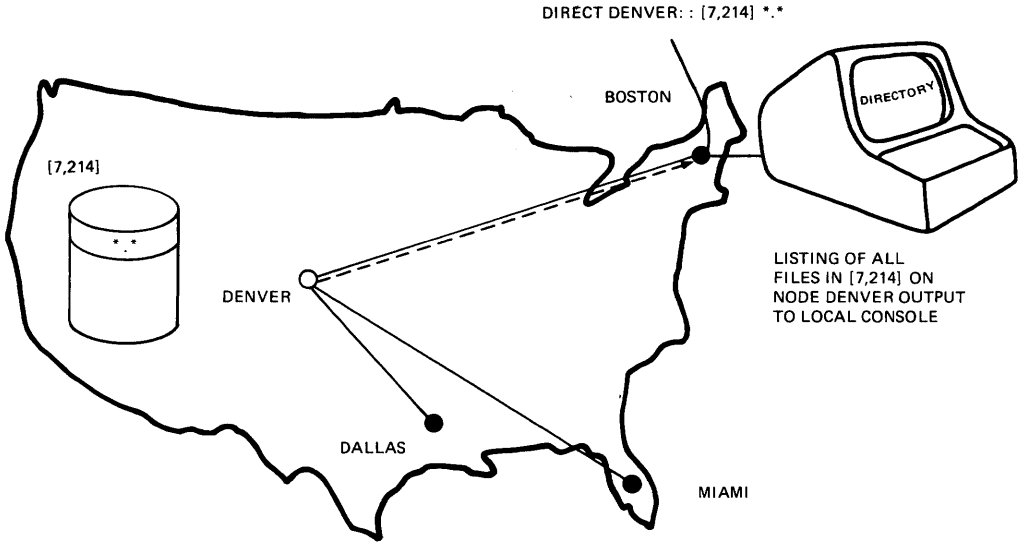


Figure 4–7 Example of the DIRECTORY Command

4.4.5 EXIT Command

The EXIT command is used to terminate execution of NFT. It is equivalent to typing CTRL/Z.

Command Format:

EXIT

4.4.6 HELP Command

The HELP command is used to display information about the commands and switches available with DECnet/E NFT.

Command Format:

```
HELP    COMMANDS
        SWITCHES
```

If neither COMMANDS or SWITCHES is specified, both sets of information are displayed.

The HELP command does not involve network access.

4.4.7 NODESPECIFICATION Command

The NODESPECIFICATION command allows you to enter or change the log-in information required to establish you as a valid user at a remote node. This information is kept and used in subsequent references to the node for this particular execution of NFT.

You do not have to use this command to define initial log-in values, however. NFT prompts for log-in information each time you refer to a node for which you have not yet given any log-in information. This log-in information is also kept and used for the entire run or until it is changed with another NODESPECIFICATION command.

(Log-in information can be altered on a one-time basis for a single command with either the node name extension (see Section 4.3.1) or with the /IDENTIFY switch (see Section 4.5.8).)

Command Format:

NODESPECIFICATION [*nodename*::]

Parameter Descriptions:

nodename

The *nodename* is the name of the node at which you wish to be validated as an established user. If omitted, NFT lists the nodes you have already specified, if any, along with the log-in information you gave for the nodes.

If *nodename* is given, NFT prompts for log-in information, first redisplaying the node requested.

```
Node:       nodename
User:       ppn
Password:   password
Account:    account-id
```

After each prompt, you type the appropriate information. The *ppn* is the project-programmer number (sometimes referred to as a user identification code or UIC) used for logging in at the remote node. It can be up to 16 characters long. If the remote node permits alphanumeric named directories in addition to or in place of numeric project-programmer numbers, you can specify the *ppn* in that form.

The *password* is a 1- to 8-character string; the *account-id* can be up to 16 characters. If a password is entered, it is not echoed at the terminal.

If a particular item, such as the *account-id*, is not used for log-in at the remote node, you can omit it by entering only a carriage return in response to the prompt.

Example:

```
NFT> NO BOSTON::
Node:                BOSTON
User:                120,4
Password:
Account:              Ret

NFT>
```

NFT, having accepted the log-in information, is ready to accept another command. Note that that password, although entered, did not echo on the terminal.

4.4.8 PRINT Command

The PRINT command copies one or more files from disk at one node to one or more temporary files at another node for input to the line printer spooler. After the temporary file is printed, it is deleted from the destination node unless the /NODELETE switch is used. The destination node must, of course, have a line printer spooler.

Command Format:

PRINT [*dstnode*::][*outfile*][=][*srcnode*::]*file1*[,...,*filen*]

Parameter Descriptions:

dstnode

The *dstnode* is the name of the node to which the files are to be copied, printed, and deleted. If the node name is omitted, the local node is assumed.

outfile

The *outfile* is the name of the temporary file at *dstnode*. If this specification contains no wildcards, the input files will be concatentated into a single output file and printed. If wildcards are specified, multiple temporary files will be created and printed at the destination node, with file names created according to the wildcards in the input specification. If the destination is a RSTS/E node, *outfile* must be a disk file.

=

The equal sign separates the destination and source specifications.

srcnode

The *srcnode* is the name of the node from which the files are to be copied. If omitted, the local node is assumed.

file1,...

These are the files to be copied from *srcnode* and printed. These files must be on disk.

The equal sign is only required if the *dstnode* or *outfile* is specified. If both are omitted, the input files will be queued to the line printer spooler at the source node. In this case, the file is *not* deleted after printing unless the /DELETE switch is used (see Section 4.5.6).

Example:

```
NFT> PRINT DENVER::TEXT.DOC=MIAMI::BOOK?.DOC
```

All files at node MIAMI with file names beginning with BOOK and with the type .DOC are concatenated and copied to a temporary file named TEXT.DOC at node DENVER. File TEXT.DOC is then printed and deleted (see Figure 4–8).

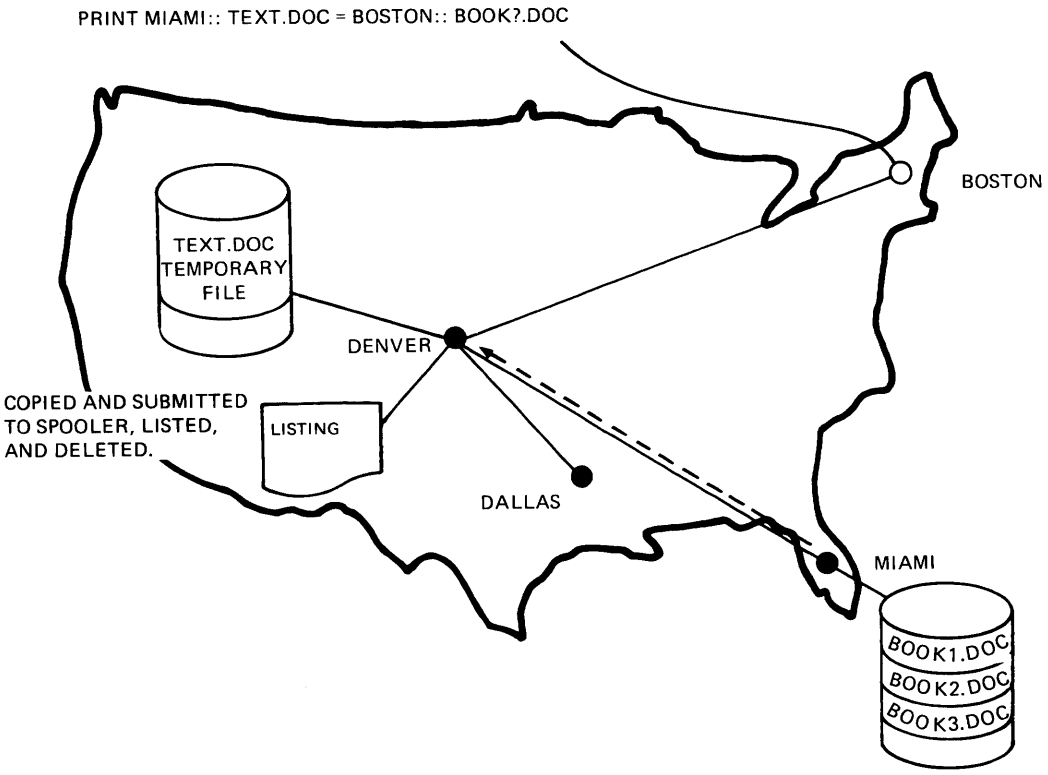


Figure 4–8 Example of the PRINT Command

4.4.9 SUBMIT Command

The SUBMIT command copies one or more files from one node to one or more temporary files at another node for execution. After execution, the command file is deleted from the destination node unless the /NODELETE switch is used. The destination node must be able to execute command files and the files must be in the proper format for execution at that node. For RSTS/E nodes, the files are submitted to BATCH, the batch processor. (The *RSTS/E System User's Guide* discusses the command file format for BATCH.)

Command Format:

SUBMIT [*dstnode*::][*outfile*][=][*srcnode*::]*file1*[,...,*filen*]

Parameter Descriptions:

dstnode

The *dstnode* is the name of the node to which the command files are to be copied, executed, and deleted. If the node name is omitted, the local node is assumed.

outfile

The *outfile* is the name of the temporary file to be created at *dstnode*. If this specification contains no wildcards, the input files will be concatentated into a single output file and submitted for execution. If wildcards are specified, multiple temporary files will be created and submitted at the destination node, with file names created according to the wildcards in the input specification.

=

The equal sign separates the destination and source specifications.

srcnode

The *scrnode* is the name of the node from which the files are to be retrieved. If omitted, the local node is assumed.

file1,...

These are files to be copied from *srcnode* and executed.

3 The equal sign is only required if *dstnode* or *outfile* is specified. If both are omitted, the input file(s) already existing at the source node are submitted to the batch processor on that node. In this circumstance, the files are not deleted after execution unless the /DELETE switch is used (see Section 4.5.6).

Example:

```
NFT> SUBMIT BOSTON::RBAT.CTL=*.CTL
```

All files at the local node with a type of .CTL are copied into a temporary file called RBAT.CTL at node BOSTON, executed, and deleted (see Figure 4–9).

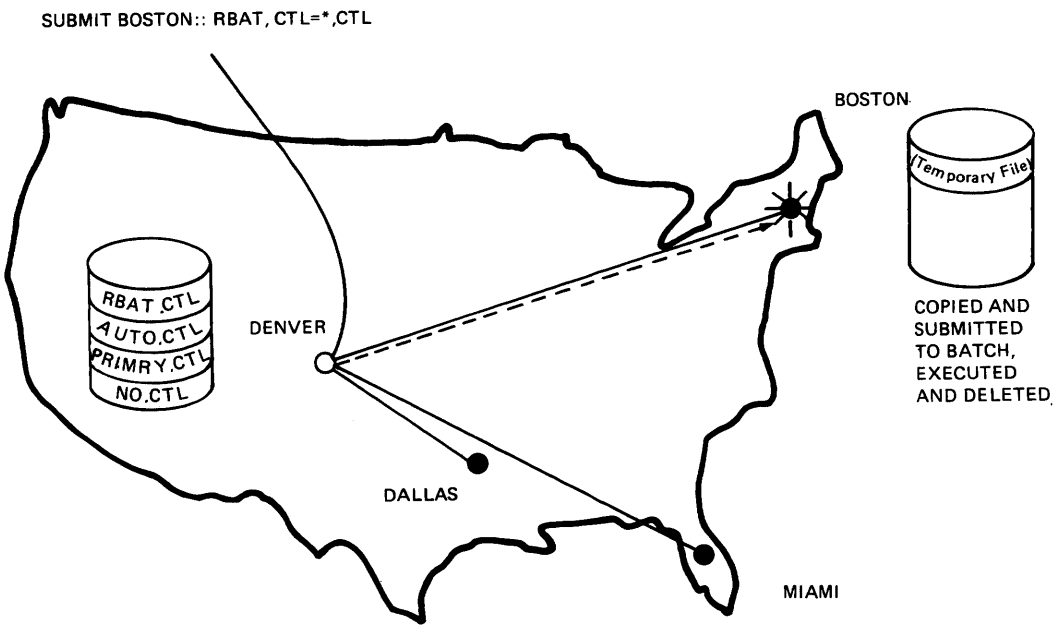


Figure 4–9 Example of the SUBMIT Command

4.4.10 TYPE Command

The TYPE command displays the input files from the source node on the user's terminal display. It is equivalent to the COPY command when everything to the left of (and including) the equal sign is omitted.

Command Format:

TYPE [*srcnode*::]*file1*[,...,*filen*]

Parameter Descriptions:

scrnode

The *srcnode* is the name of the source node from which the file(s) are to be retrieved. If omitted, the local node is assumed.

file1,...

These are the file specifications for the files stored at the source node that are to be typed on the user's console.

Example:

```
NFT> TY DALLAS::DAILY.SUM
```

In this example, the file DAILY.SUM is retrieved from the public disk on remote node DALLAS and typed out on the user's keyboard console (Figure 4–10).

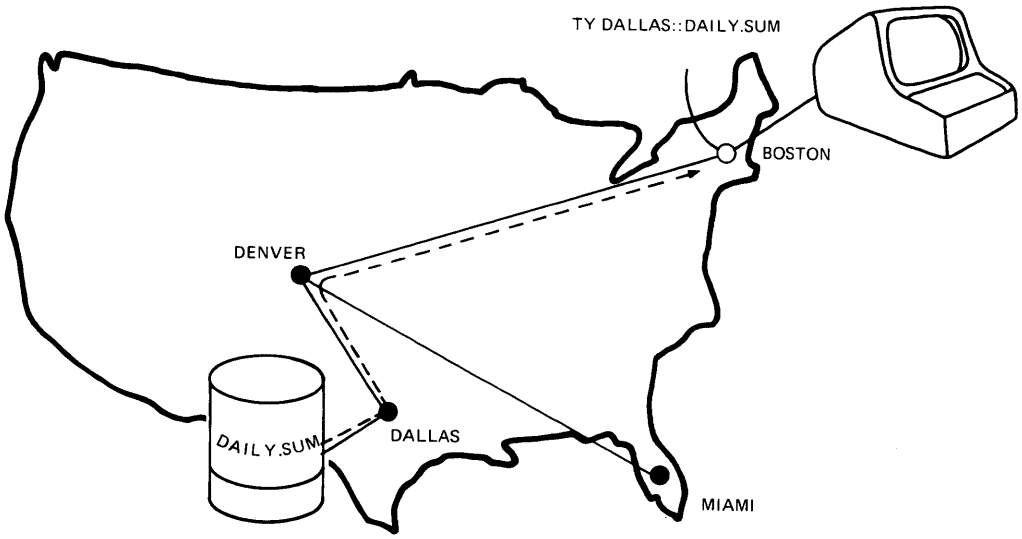


Figure 4–10 Example of the TYPE Command

4.4.11 VERSION Command

The VERSION command is used to display the version numbers of the local DECnet/E NFT program or the remote FAL program, as well as the version number of the Data Access Protocol (DAP) being used.

Command Format:

VERSION [*srcnode*::]

Parameter Descriptions:

srcnode

The *srcnode* is the name of the node at which the version number of FAL is to be examined. If this parameter is omitted, the version number of the local NFT is displayed, along with the version number of DAP.

4.5 NFT Switches

The switches listed below can be used in NFT commands. Most of these switches apply only to certain commands and some can apply only to the source or destination file specification. In general, a switch is appended to a file specification and it applies to that file only. More than one switch can be used as long as they are logically consistent.

/APPEND	Append input file(s) to output file.
/ASCII	Data being transferred is ASCII 7-bit data.
/BLOCK	Transfer image data in 512-byte blocks.
/BRIEF	Generate an abbreviated directory listing.
/CONTIGUOUS or /CTG	Allocate space contiguously.
/DELETE	Special-case delete for SUBMIT and PRINT commands.
/FULL	Generate an extended directory listing.
/IDENTIFY	Have NFT prompt for log-in information.
/IMAGE	Data being transferred in 8-bit data.
/INQUIRY	Verify input file before function execution.
/LIST	Generate a normal directory listing.
/LOG	Output file name after function execution.
/MORE or -	Continue command input on next line.
/NATIVE	Reformat file to stream ASCII.
/NOCONTIGUOUS	Allocate space noncontiguously.
/NODELETE	Do not delete temporary file.
/NOHEADING	Generate directory listing with no header.
/NOSUPERSEDE	Do not supersede an existing file.
/POS	Position magnetic tape to end of volume.
/RWC	Rewind magnetic tape on close.
/RWO	Rewind magnetic tape on open.
/SUPERSEDE	Supersede an existing file.
/TOTAL	Output directory size only.
/VARIABLE	Reformat file to variable format.

4.5.1 /APPEND – Append Input to Output

The /APPEND switch can be applied to the destination file specification in the COPY and DIRECTORY commands. When used with the COPY command, the effect is equivalent to the APPEND command; the input files are appended to the output file.

NOTE

When using this switch with the COPY command, the user should be aware of the attributes of the files to be appended and ensure that the files are compatible. Appending a variable-length record file to one with fixed-length records, for example, can have unpredictable results. An error message may or may not be output.

When used with the DIRECTORY command, applied to an existing output file, the resulting directory listing is appended to the specified file rather than overwriting it.

Example:

```
NFT> COPY TEXT.DOC/APP=DALLAS::BOOK??.DOC
NFT> DIRECTORY DIRECT.LST/AP=[7,214]*.BAS
```

4.5.2 /ASCII – Transfer 7-Bit ASCII Data

This switch is used to indicate that a transmitted file contains 7-bit ASCII data. It can be applied to the source file specification of the COPY, TYPE, SUBMIT, APPEND, or PRINT commands.

Example:

```
NFT> COPY MIAMI::MEMO.DOC=[7,214]NOTES.TXT/ASC
```

4.5.3 /BLOCK – Transfer Image Data in 512-Byte Blocks

This switch is used when transferring image (binary) files with the COPY or APPEND commands. It should be applied to the source file specification, and causes the files to be transferred in 512-byte blocks.

Note that the /BLOCK switch is generally only applicable for RSTS to RSTS file transfers since other operating systems do not fully support binary block transfers. The switch is necessary for non-ASCII files that do not have recorded attributes. On RSTS/E systems, such files include compiled BASIC–PLUS files (file types = .BAC), record-I/O or virtual array data files, and binary files such as those with file types of .OBJ or .SAV. The

switch should not be used, however, for files that have record attributes (e.g. MAC.TSK object modules). Using the /BLOCK switch results in a BLOCK image copy of the file, with the original attributes being lost. Thus, the /BLOCK switch should be used *only* with unattributed files.

Example:

```
NFT> COPY STORBN.BAC=BOSTON::COMP.BAC/BL
```

NOTE

The /BLOCK switch is incompatible with and should not be used with either the /NATIVE switch (described in Section 4.5.14) or the /VARIABLE switch (described in Section 4.5.24).

4.5.4 /BRIEF – Output a Brief DIRECTORY Listing

The /BRIEF switch can be used at the end of the DIRECTORY command to give a brief directory listing including device, directory, file name, and file type. This listing is similar to the listing generated by the RSTS/E PIP program.

Example:

```
NFT> DIR QUICK.DIR=MIAMI::[7,214]*.DOC/BR
```

4.5.5 /CONTIGUOUS or /CTG – Allocate Space Contiguously

This switch can be applied to the destination file specification in the COPY, SUBMIT, or PRINT command. It allocates contiguous space for the destination file(s). If a contiguous allocation cannot be made, the command will fail.

Example:

```
NFT> COPY DALLAS::FILE1.TSK/CO=MYFILE.TSK
```

In this example, contiguous space is allocated at node DALLAS (an RSX-11M node) for the binary file FILE1.TSK, consisting of the file MYFILE.TSK transferred from the local node.

4.5.6 /DELETE – Delete Files on Close

This switch can be applied to any file specification in any command. It is used to delete the indicated file upon successful completion of the specified function.

The primary use of this switch applies to the source file specification of the SUBMIT or PRINT commands when no destination file specification is supplied – that is, when the files to be printed or submitted to the batch processor already exist at the desired node. With this form of these two commands, the files are not automatically deleted since they are, in fact, the *source* files and not temporary destination files. Using the /DELETE switch in this case will cause the files to be deleted at the conclusion of the operation.

Example:

```
NFT> PRINT ROUGH.DRF/DELETE
```

4.5.7 /FULL – Output a Full DIRECTORY Listing

The /FULL switch can be used at the end of the DIRECTORY command to give a directory listing including device, directory, file name, file type, file size, protection code, creation date and time, as well as a listing of the symbolic attributes. This listing is similar to the full directory listing generated by the RSTS/E PIP program.

Example:

```
NFT> DIR DALSYS.DIR=DALLAS::SY:[1,2]*.*/FULL
```

4.5.8 /IDENTIFY – Prompt for Log-in Information

This switch can be applied once to both the source and destination specifications of any command. It causes NFT to prompt for log-in information. The information you give in response to this prompting is used only for the current operation. It is not kept for the rest of the NFT execution. After the execution of a command with this switch, the log-in information reverts to what was previously specified for the node, if any.

If the /IDENTIFY switch is applied to both the source and destination specifications, NFT prompts for information for the source node first. A second set of prompts is used to obtain information concerning the destination node.

Example:

```
NFT> COPY BOSTON::FILE1.TXT/ID=DALLAS::FILE3.TXT,FILE4.TXT/ID

Node:           DALLAS
User:           120,150
Password:
Account:

Node:           BOSTON
User:           30,117
Password:
Account:
```

NFT uses the log-in information to copy the files and then deletes the log-in information from its internal records. Previously specified log-in information for BOSTON and DALLAS, if any, is restored for use in subsequent commands. Note that the password is not echoed on the keyboard.

4.5.9 /IMAGE – Transfer 8-Bit Image Data

This switch is used to indicated that a transmitted file contains 8-bit image data. It can be applied to the source file specification of the COPY and APPEND commands.

Example:

```
NFT> APPEND DALLAS::TOTALS.DAT=BOSTON::[1,34]DAILY.SUM/IMAGE
```

4.5.10 /INQUIRY – Prompt for Input File Verification

The /INQUIRY switch can be applied to the source file specification the APPEND, COPY, SUBMIT, PRINT, TYPE, and DELETE commands. It is used to request that the name of each input file be displayed for verification before the file is opened and the indicated function is performed. After the name is displayed the user can respond to the prompt with either Y[ES] or N[O] to indicate whether the operation should, in fact, be performed on that particular file. If a NO response is given, the file is skipped and the next file name is displayed.

Example:

```
NFT> DELETE MIAMI::[7,214]*.DOC/INQ
```

This switch is especially useful when the destination file specification does not contain wildcards and the source files are to be concatenated into one file. In this circumstance, if the source files represented by the wildcard specification have incompatible attributes, unpredictable results can occur, depending on how the final destination file is to be used; an error message may or may not be generated. By using the /INQUIRY switch, the user can make a decision, on a file by file basis, as to whether or not a file should be included in the operation.

4.5.11 /LIST – Output a Normal DIRECTORY Listing

The /LIST switch can be used at the end of the DIRECTORY command to give a directory listing including device, directory, file name, file type, file size, protection code, and creation date and time. This listing is similar to the directory listing generated by the RSTS/E PIP program.

If no switch is appended to the DIRECTORY command, the /LIST switch is assumed.

Example:

```
NFT> DIRECT DENVER::[100,57]*.*/LIST
```

4.5.12 /LOG – Log Input File Name on Close

The /LOG switch can be applied to the source file specification in the APPEND, COPY, SUBMIT, PRINT, TYPE, and DELETE commands. It is used to request that the name of each input file be logged to the console after the indicated function is performed. This is useful when the input file specification contains wildcards.

Example:

```
NFT> COPY BOSTON::[1,230]*.*=DENVER::[3,45]*.MAC/LOG
```

4.5.13 /MORE or - – Continue Command to Next Line

This switch can be used at the end of a terminal line to indicate that the command is to be continued on the next line. NFT then prompts for continuation lines with the characters "MORE>". A CTRL/Z typed in a continuation line causes the whole command to be deleted and the "NFT>" prompt to be displayed. (Otherwise, a CTRL/Z terminates NFT and returns control to the RSTS/E monitor.) To continue a command to several lines, the switches can be used more than once.

The /MORE switch or hyphen (-) cannot be used when NFT is run as a single-line command:

NFT *command*

Example:

```
NFT>  DALLAS::DK0:[120,153]FILE.RNO=BOSTON::[30,205]DK1/MORE
MORE> :FILE.RNO,-
MORE> FILE3.RNO
NFT>
```

4.5.14 /NATIVE – Reformat File to Stream ASCII

This switch can be applied to the destination file specification in a COPY, APPEND, SUBMIT, or PRINT command when the destination node is a RSTS/E node. It may also be used in the TYPE command. It converts files stored at the source node (in either of two ASCII formats) to stream ASCII

files at the destination node. The two source formats are: (1) RMS–11 variable-length records with implied carriage control, or (2) variable-length records with imbedded carriage control characters. This is useful when transferring ASCII files from an RSX or VAX/VMS node, that does not process stream ASCII, to a RSTS/E node, that does.

Example:

```
NFT> COPY TEST.MAC/NA=DALLAS::FIL.MAC
```

The /NATIVE switch can also be applied to the source file specification in a COPY, TYPE, APPEND, SUBMIT, or PRINT command when the source node is either a TOPS20 or an RT11 node and the files are to be transferred as stream ASCII files to the destination node. The /NATIVE switch is used to inform the source node that the file is stream ASCII, so that the source node sends it as such across the network. RT11 and TOPS20 files do not have file attributes recorded as part of the file, so NFT at the source has no way of knowing, without the /NATIVE switch, that the file is stream ASCII.

Example:

```
NFT> COPY PROGA.MAC=MIAMI::PROGA.MAC/NA
```

NOTE

The /NATIVE switch is incompatible with and should not be used with the /BLOCK switch, described in Section 4.5.3.

4.5.15 /NOCONTIGUOUS – Allocate Space Noncontiguously

This switch can be applied to the destination file specification in the COPY, SUBMIT, or PRINT command. It allocates noncontiguous space for the destination file(s).

Example:

```
NFT> COPY DALLAS::FILE1.TSK/NOCO=MYFILE.TSK
```

In this example, noncontiguous space is allocated at node DALLAS (an RSX-11M node) for the binary file FILE1.TSK, consisting of the file MYFILE.TSK transferred from the local node.

4.5.16 /NODELETE – Save Temporary Files

The /NODELETE switch can be applied to the destination file specification in a PRINT or SUBMIT command to suppress deletion of the temporary file(s) created at the destination node.

Example:

```
NFT> SUBMIT BOSTON::RBAT.CTL/NOD=*.CTL
NFT> PRINT TEXT.DOC/NODELETE=DALLAS::BOOK??.DOC
```

4.5.17 /NOHEADING – Output a DIRECTORY Listing with No Header

The /NOHEADING switch can be used at the end of the DIRECTORY command to give a listing without the header line across the top. The directory listing includes a complete file specification for each entry: device, project-programmer number, file name and file type. This switch is useful in creating batch command files.

Example:

```
NFT> DIRECT LIBRY.CTL=DALLAS::[2,173]*.SAV/NOHEAD
```

4.5.18 /NOSUPERSEDE – Do Not Supersede an Existing File

This switch can be applied to the destination file specification in a COPY, SUBMIT, or PRINT command. If a file already exists at the destination node with the same name as that given in the command, the command will fail.

Normally, when NFT finds that the specified output file name already exists, it prompts the user, asking whether the existing file should be superseded. The /NOSUPERSEDE switch disables this automatic feature. This is useful in batch command files that cannot handle unexpected prompting for user input.

Example:

```
NFT> COPY FILE.LST/NOSU=BOSTON::FILE.TXT
```

4.5.19 /POS – Position to End of Volume

The /POS switch can be applied to the destination file specification in a COPY command when the device is magnetic tape. It positions the file to the current end of volume before the transfer.

Example:

```
NFT> COPY MT2:HPFIL.BAC/POS=BOSTON::HPRTN.BAC
```

4.5.20 /RWC – Rewind on Close

The /RWC switch can be applied to any file name in a COPY command when the device is magnetic tape. It rewinds the tape after the file is closed (after the transfer).

Example:

```
NFT> COPY MT1:RMAX.BAS/RWC=BOSTON::RMAX.BAS
```

4.5.21 /RWO – Rewind on Open

The /RWO switch can be applied to any file name in a COPY command when the device is magnetic tape. It rewinds the tape before the file is opened (before the transfer).

Example:

```
NFT> COPY MT0:FILEX.RNO/RWO=BOSTON::TEXT.RNO
```

4.5.22 /SUPERSEDE – Supersede an Existing File

This switch can be applied to the destination file specification in a COPY, SUBMIT, or PRINT command. If a file already exists at the destination node with the same name as that given in the command, the existing file will be replaced by the new file.

Example:

```
NFT> COPY FILE.LST/SU=BOSTON::FILE.TXT
```

4.5.23 /TOTAL – Output Directory Size Only

The /TOTAL switch can be used at the end of the DIRECTORY command to output the total size of the directory. Individual directory entries are not listed.

Example:

```
NFT> DIR MIAMI::[1,2]*.*/TOTAL
```

4.5.24 /VARIABLE – Reformat File to Variable Format

This switch can be applied to the destination file specification in a COPY, APPEND, SUBMIT, or PRINT command. It converts a file from stream ASCII at the source node to RMS–11 variable-length record format with

implied carriage control. This switch is used mainly to transfer stream ASCII files from a DECnet/E or DECnet/RT node to an RSX or VAX/VMS node because RSX and VAX/VMS do not process stream ASCII files.

Example:

```
NFT> COPY DALLAS::FIL.MAC/VA=TEST.MAC
```

This example copies a MACRO file from a DECnet/E node to the RSX node in the sample network.

NOTE

The /VARIABLE switch is incompatible with and should not be used with the /BLOCK switch, described in Section 4.5.3.

4.6 NFT Error Messages

The error messages in this section are printed by NFT when you make some mistake in typing an NFT command (syntax errors) or when an error has occurred in initializing NFT. These errors are described in this section.

Other error messages may appear for conditions diagnosed according to the Data Access Protocol (DAP) or by RMS (Record Management Services, used by NFT and FAL to do local I/O). These error messages are given in Appendix D.

?NFT – Attempted continuation is illegal

You cannot continue a one-line CCL command with the /MORE or - switch.

?NFT – Break buffer overflow

There are too many quotes, commas, underscores, and equal signs in the command. The current limit on these special characters per command is 40.

?NFT – File specification too long

The file specification you typed is too long to fit in an internal buffer used by NFT.

?NFT – GMCR$ directive failure

Tell your software specialist that the RSX–11M directive used to read the CCL command line has failed. Then retype the command, starting NFT with RUN $NFT.

?NFT – Invalid command format

Can be caused by several errors:

1. Trying to type something other than a node name in a NODESPECIFICATION command.
2. Typing "::" in a NODESPECIFICATION command without any node name.
3. Using any switch other than /IDENTIFY of /INQUIRY in a DELETE or EXECUTE command.
4. Not typing an equal sign (=) in an APPEND command.
5. Typing an equal sign (=) in a DELETE, TYPE, or EXECUTE command.
6. Typing anything but COMMANDS or SWITCHES after a HELP command.
7. Not typing a node name in an EXECUTE command.

?NFT – Invalid network node name

Valid node names are 1 to 6 uppercase, alphanumeric characters and must contain at least one letter.

?NFT – Invalid node name specification

Either the node name is not in the right place or it contains non-alphanumeric characters.

?NFT – More than one input node specification

You cannot type more than one node name on the right side of an equal sign.

?NFT – More than one output node specification

You cannot type more than one node name on the left side of an equal sign.

?NFT – No node name given for /IDENTIFY

No node name was given for the /IDENTIFY switch to operate on.

?NFT – RMS initialization failure

An error occurred in initializing RMS. If the error persists, call a software specialist.

?NFT – Too many /IDENTIFY switches

More than one /IDENTIFY switch was given on one side of an equal sign.

?NFT – Too many output file specifications

More than one file name was given on the left side of an equal sign.

?NFT – Too many = signs

The command typed contains more than one equal sign.

?NFT – Unmatched quote characters

There are an odd number of quote characters in the command line. Quote characters must be paired.

?NFT – Unrecognizable command

Valid command keywords are described in Section 4.4.

?NFT – Unrecognizable switch

Valid switches are described in Section 4.5.

Chapter 5
NETCPY: Network Copy between Devices

The NETCPY utility allows you to copy all of the information on a DECtape, magtape, floppy disk, or disk from one DECnet/E node to a similar device on another DECnet/E node. One of the nodes must be the local node and both must be DECnet/E nodes. (NETCPY is a DECnet/E utility. There is no general DECnet utility to copy the contents of devices between nodes.)

To protect devices from unauthorized use, a NETCPY user must supply log-in information for the remote node specified in the command. NETCPY will prompt for the information and check its validity before allowing the transfer to proceed. You can use the TLK utility (Chapter 2) to ask the operator at the remote DECnet/E system to mount tapes or do other preliminary preparation of the physical devices.

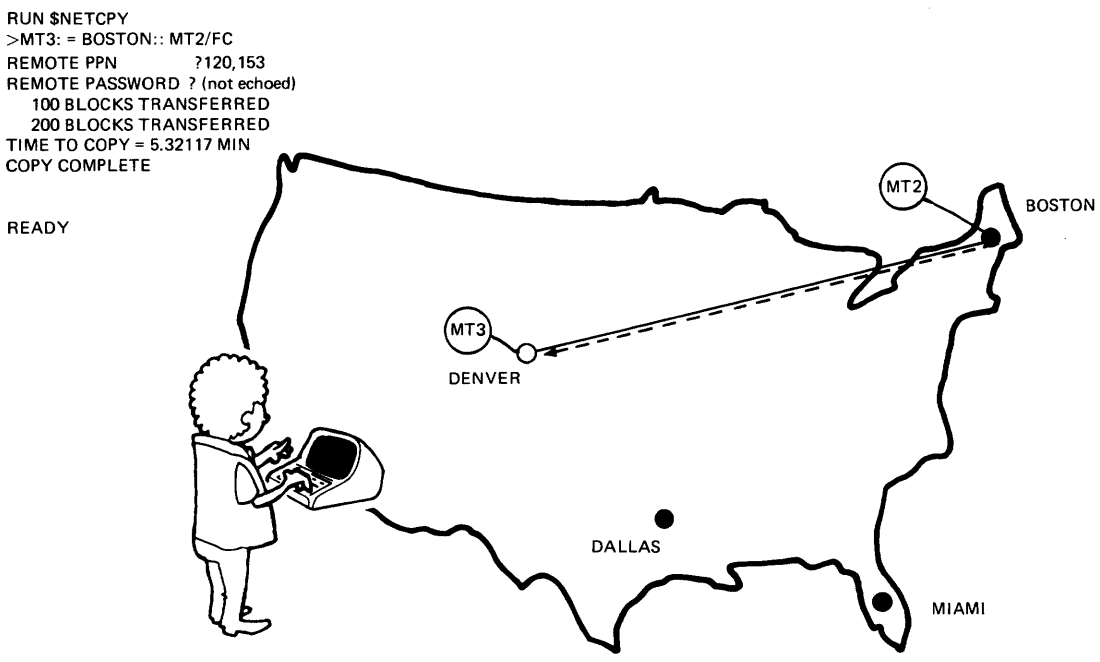


Figure 5–1 NETCPY Copies an Entire Device from One DECnet/E Node to Another DECnet/E Node

5.1 Running NETCPY

You can run NETCPY from any terminal in a DECnet/E system by typing:

```
RUN $NETCPY
```

NETCPY responds with a message identifying itself and a pound-sign prompt (#). It will then accept one command. After you type the command, NETCPY will prompt for log-in information for the remote node, process the command, and terminate. Only one copy command can be given each time NETCPY is run.

```
RUN $NETCPY
NETCPY V2.0 RSTS V7.1 DECNET/E
#command
REMOTE PPN          ? ppn
REMOTE PASSWORD     ? password

(NETCPY processes command, lists progress)

Ready
```

The *ppn* is the project-programmer number used to log in at the remote system – for example, 100,153. Likewise, the *password* is used to log in at the remote system. It is not echoed as you type it.

The *command* is described in detail in the next section.

5.2 General Form of NETCPY Command

General Form of Command:

[dstnode::]dstdevice:[dstsw] = [srcnode::]srcdevice: { /FC / /NC } *[srcsw]*

Parameter Descriptions:

dstnode

The *dstnode* is the destination node – the 1- to 6-character, uppercase, alphanumeric node name to which the contents of the device are to be copied. The name must begin with at least one letter. If this node name is omitted, the local node is assumed. Either *srcnode* or *dstnode* MUST be the local node. Both must be DECnet/E nodes.

dstdevice

The *dstdevice* is the destination device. Allowable devices include the following:

MT*n*: Magtape unit *n*
DX*n*: Floppy disk unit *n*
DK*n*: Disk unit *n*
DT*n*: DECtape unit *n*

The destination device must be the same type as the device specified by *srcdevice*. NETCPY can be used to copy larger disks as well, but the speed of most transmission lines usually makes such operations impractical for remote transfers.

dstsw

Either or both of the optional switches /PA and /DE can be used, as described in Section 5.3.

=

The equal sign separates the destination description from the source description.

srcnode

The *srcnode* is the source node – the node from which the information is to be copied. If this name is omitted, the local node is assumed. Either *dstnode* or *srcnode* MUST be the local node. Both must be DECnet/E nodes.

srcdevice

The *srcdevice* is the source device. Allowable devices include the following:

MT*n*: Magtape unit *n*
DX*n*: Floppy disk unit *n*
DK*n*: Disk unit *n*
DT*n*: DECtape unit *n*

The source device must be the same type as the device specified by *dstdevice*.

/FC

The /FC switch indicates that a fast copy is to be done.

/NC

The /NC switch indicates no copy is to be done. It is used with the /VE switch for a verify-only run.

srcsw

Any or all of the optional switches /VERIFY, /BLOCK, /PARITY, and /DENSITY can be used, as described in Section 5.3. Switches can be abbreviated to the first two characters of the keyword.

Example:

```
#BOSTON::MT1:=MT2:/FC
```

This command will cause the magtape device unit 2 at the local node to be copied to magtape device 1 at node BOSTON. The fast copy (/FC) switch is used to cause the copy.

5.3 NETCPY Option Switches

There are seven option switches that can be used with NETCPY. These are listed below and described in the following subsections.

/BLOCK	Specify Block size for magtape.
/DENSITY	Set the density for magtape.
/FC	Request a device copy.
/HELP	Obtain instructional information.
/NC	Request no device copy.
/PARITY	Set the parity for magtape.
/VERIFY	Verify that the data is copied correctly.

5.3.1 /BLOCK:*b* – Block Size Switch

The /BLOCK switch, specifying block size, will allocate buffers for magtapes written with nonstandard record sizes. The default block size is 2048-byte blocks, which will handle most tapes and allows fast disk copies. Block (*b*) sizes must be in multiples of 512 bytes. The *srcdevice* is buffered at the specified block size and the *dstdevice* is written (and/or verified) at the specified block size.

Example:

```
#MTO:=BOSTON::MT1:/BL:4096/FC
```

This command will cause magtape unit 1 at node BOSTON to be copied to magtape unit 0 at the local node in 4096-byte blocks, rather than the default of 2048-byte blocks.

5.3.2 /DENSITY:*d* – Set Density Switch

The /DENSITY switch can be used when copying magtapes to set the density for either the source or destination device. The *d* parameter can be either 800 or 1600 bits per inch (BPI).

Example:

```
#MTO:/DENSITY:1600/PARITY:EVEN=BOSTON::MT1:/DENSITY:800/FC
```

NETCPY reads a 9-track magtape on unit 1 at node BOSTON at 800 BPI and writes in 9-track format on magtape unit 0 at the local node at 1600 BPI with even parity (see Section 5.3.6).

5.3.3 /FC – Fast Copy Switch

The /FC switch is used to request a device copy. Either /FC or /NC must be used after the *srcdevice* specification. If neither is used, NETCPY takes no action and prints an error message.

NETCPY lists the progress of the copy in 100-block units for disk and in 100-record units for tape. When the copy is complete, NETCPY lists the time it took and the total number of blocks or records transferred.

Example:

```
RUN $NETCPY
NETCPY V2.0 RSTS V7.1 DECNET/E
#BOSTON::DT0:=DT2:/FC
REMOTE PPN       ? 1,222
REMOTE PASSWORD ?

 100 BLOCKS TRANSFERRED
 200 BLOCKS TRANSFERRED
 300 BLOCKS TRANSFERRED
 400 BLOCKS TRANSFERRED
 500 BLOCKS TRANSFERRED
TIME TO COPY = 5.11111 MIN
COPY COMPLETE
BLOCKS TRANSFERRED = 578

Ready
```

5.3.4 /HELP – Help Switch

You can type /HE in response to the pound-sign prompt and get a help display at the terminal. After the help information is displayed, another pound-sign prompt is given and you can then type a command.

Example:

```
RUN $NETCPY
NETCPY V2.0  RSTS V7.1 DECNET/E
#/HE
   .
   .
   .
(Help Display)
   .
   .
   .
#(command)
```

5.3.5 /NC – No Copy Switch

The /NC switch is used to request no copy. This switch is always used with the /VE (verify) switch for a verify-only run. Either /NC or /FC must be used after the *srcdevice* specification. The /NC switch has no effect by itself.

5.3.6 /PARITY:*p* – Set Parity Switch

The /PARITY switch can be used when copying magtapes to set the parity for either the source or destination device. The *p* parameter can be either "ODD" or "EVEN". The default is whatever was set by the system manager for that device at the node referenced.

An example of the use of the /PARITY switch is given in Section 5.3.2.

5.3.7 /VERIFY – Verify Switch

This switch can be used with either the /FC or /NC switch to verify that the information on a device unit has been copied exactly. Verification is performed block by block and takes as much time as the copy operation. When used with /FC, the information is first copied from the source device to the destination device, and then sent back to be verified. When used with /NC, the information is simply verified. As the verification is performed, NETCPY lists the progress.

Example:

```
RUN $NETCPY
NETCPY V2.0 RSTS V7.1 DECNET/E
#BOSTON::DT0:=DT2:/NC/VE
REMOTE PPN       ? 1,222
REMOTE PASSWORD ?
BEGINNING VERIFICATION PASS
 100 BLOCKS VERIFIED
 200 BLOCKS VERIFIED
 300 BLOCKS VERIFIED
 400 BLOCKS VERIFIED
VERIFICATION COMPLETE    0  BAD BLOCKS
OUT OF 492 BLOCKS VERIFIED

Ready
```

If the information has not been copied correctly, NETCPY prints the decimal number of blocks or records in which inconsistencies appear. The numbers printed correspond to magtape record numbers and to disk and DECtape blocks of 512 bytes.

Example:

```
#DX1:=BOSTON::DX2/FC/VE
REMOTE PPN       ? 102,50
REMOTE PASSWORD ?
 100     BLOCKS TRANSFERRED
 200     BLOCKS TRANSFERRED
 300     BLOCKS TRANSFERRED
 400     BLOCKS TRANSFERRED
TIME TO COPY =  7.56667 MIN
COPY COMPLETE
BLOCKS TRANSFERRED #492
```

```
BEGINNING VERIFICATION PASS
THE FOLLOWING BLOCKS ARE BAD
17
31
89
 100    BLOCKS VERIFIED
 200    BLOCKS VERIFIED
 300    BLOCKS VERIFIED
 400    BLOCKS VERIFIED

VERIFICATION COMPLETE 3 BAD BLOCKS
OUT OF 492 BLOCKS VERIFIED


Ready
```

5.4 NETCPY Error Messages

Cannot specify both /FC and /NC

Either fast copy (/FC) or no copy (/NC) option must be specified, but not both.

Error in specifying density

The density selected is not one of the valid values 200, 556, 800, DUMP, or 1600.

Error in specifying parity

The parity selected must be either ODD or EVEN.

/FC or /NC must be specified

Either fast copy (/FC) or no copy (/NC) must be specified.

Illegal block size

The block size selected with the /BL option must be a multiple of 512.

Invalid node specification

Either the nodes specified did not include the local node, or NETCPY did not recognize the node name typed.

Must have same type device

Both the source device and the destination device must be magtape (MT*n*:), DECtape (DT*n*:), floppy disk (DX*n*:), or disk (DK*n*:).

(NSP command) – ABORT

NETCPY sets up a logical link to the remote node to effect the transfer requested. If any network error occurs during execution of NETCPY, the program is aborted and this message is printed. The NSP command printed is the one NETCPY was trying to execute when the error occurred. For example, if NETCPY was trying to receive a message from the remote node, the message RECEIVE NETWORK ERROR – PROGRAM ABORT is printed.

Type /HE for help

NETCPY does not recognize the command typed. Use the /HE switch for help.

Chapter 6
NETOFF: Network Shutdown

The NETOFF utility lets you shut down your network in an orderly fashion without shutting down the entire RSTS/E operating system. It causes a gradual shutdown rather than an immediate disconnect.

NETOFF performs a similar operation to the NCP SET EXECUTOR STATE SHUT command (see *DECnet/E System Manager's Guide*). Both permit existing logical links to complete before ending network operations, but only NETOFF warns users that the network will close down in some determined period of time. This gives network users an opportunity to respond to the warning message by finishing up network processing or by requesting that network operations be extended.

To invoke NETOFF, you type:

```
RUN $NETOFF
```

NETOFF responds by asking how many minutes it should wait before new network links are prohibited (that is, until it effects a SET EXECUTOR STATE SHUT command) and how many minutes it should wait until actual network shutdown.

After this, the following sequence of operations occurs:

1. NETOFF broadcasts a message to all system terminal users:

   THE NETWORK IS SHUTTING DOWN IN [variable] minutes

2. NETOFF then suspends execution for the appropriate number of minutes, as determined in the dialog. When it resumes execution, it sends the SET EXECUTOR STATE SHUT message to NSP. If no logical links are open, NSP terminates network operations and the NETOFF program completes. Otherwise, NSP prohibits the forming of any new logical links, but allows normal operations to continue on existing links.

3. NETOFF again suspends execution, this time for the remainder of the period before final network shutdown. When it resumes execution, it sends the SET EXECUTOR STATE OFF message to NSP.

4. At this point, NSP aborts any existing logical links and sends a Network Abort message to any local programs with links open. NSP then shuts down Transport, thus ending network operations.

5. Finally, NETOFF sends a STOP command to the Event Logger program and displays the following message on the terminal:

   ```
   SHUTDOWN COMPLETE
   ```

You can interrupt this sequence at any point by typing CTRL/C. NETOFF resumes execution and sends the SET EXECUTOR STATE ON command to NSP, aborting the shutdown procedure and resuming normal network operations.

Appendix A
Object Type Codes

Code	Type of Process
000	General Task, User Process
001	File Access (FAL/DAP Version 1)
002	Unit Record Services (URD)
003	Application Terminal Services (ATS)
004	Command Terminal Services (CTS)
005	RSX–11M Task Control, Version 1
006	Operator Services Interface
007	Node Resource Manager
008	IBM 3270-BSC Gateway
009	IBM 2780-BSC Gateway
010	IBM 3790-SDLC Gateway
011	TPS Application
012	RT–11 DIBOL Application
013	TOPS–20 Terminal Handler
014	TOPS–20 Remote Spooler
015	RSX–11M Task Control, Version 2
016	TLK Utility
017	File Access (FAL/DAP Version 4 and later)
018	RSX–11S Remote Task Loader
019	Network Management Listener (NICE Process)
020	RSTS/E Media Transfer Program (NETCPY)
021	Reserved for DECnet use
022	Mail Listener
023	Host Terminal Handler (NPKDVR)
024	Concentrator Terminal Handler
025	Loopback Mirror
026	Event Receiver
027	VAX/VMS Personal Message Utility
028	File Transfer Spooler (FTS)
029-062	Reserved for DECnet use
063	DECnet test tool (DTR)
064-127	Reserved for DECnet use
128-255	Reserved for customer extensions

Appendix B
NSP Reason Codes for Connect Reject and Link Abort

Code	Reason
000	No error – user-initiated reject or abort
001	Resource allocation failure
002	Destination node does not exist
003	Node shutting down
004	Destination program does not exist
005	Invalid destination name or source field
006	Destination program's queue full
007	Unspecified error condition
008	Third party aborted logical link
009	User-initiated link abort
010-020	Reserved
021	Invalid destination address in Connect Initiate Message
022	Invalid desination address in Connect Confirm Message
023	Source address zero in Connect Initiate or Connect Confirm Message
024	Flow control violation – invalid request count in Link Service Message
025-031	Reserved
032	Too many connects to node
033	Too many connects to destination program
034	Access not permitted
035	Logical link services mismatch
036	Invalid accounting information
037	Segment size too small
038	User aborted, timed out, or canceled link
039	No path to node
040	Flow control failure – data received when request count zero
041	No current link (cannot recognize destination address)
042	Confirmation from remote system of Disconnect Message
043	Image data field too long

Appendix C
NETACT: Automatic Log-in Utility for NFT

The NETACT utility lets you define log-in entries for remote nodes so that you do not have to define them each time you run NFT. NETACT creates a file in your account (NETACT.DAT) that NFT uses to provide access control information whenever you reference a new remote node name in an NFT command. If the access control information specified in the file is subsequently rejected by the remote node, NFT will delete the entry from its internal data structures. It does *not*, however, delete the entry from NETACT.DAT.

The NFT command NODESPECIFICATION and the /IDENTIFY switch takes precedence over the NETACT.DAT file in an NFT run.

C.1 How to Run NETACT

You can run NETACT from any terminal in a DECnet/E system by typing:

```
RUN $NETACT
```

NETACT displays a message identifying itself and a prompt: NETACT>. It will then accept commands until you use the EXIT command to terminate NETACT. You can also type a CTRL/Z at any point during execution of NETACT.

```
RUN $NETACT
NETACT V2.0 RSTS V7.1 The ARK
NETACT>command
        .
        .
        .
NETACT>^Z

Ready
```

C.2 NETACT Commands

Six commands providing five functions are available with NETACT. These commands are listed below and described in the following subsections. Each command can be abbreviated to the first letter of the keyword.

DEFINE	Lets you define a log-in sequence for a remote node.
EXIT	Terminates NETACT.
HELP or ?	Displays a line of helpful information at your terminal.
LIST	Displays the current definitions.
PURGE	Eliminates a log-in sequence for a remote node.

C.2.1 DEFINE Command

The DEFINE command displays a series of questions. You respond to these questions to define a log-in sequence for a node.

Example:

```
NETACT>DEFINE
Node name                    ? DENVER
User:              <>        ? 2,155
Password:          <>        ? MELBA
Account:           <>        ? 3N4
```

NETACT stores the entries in the file accessed by NFT. Thereafter, when the user types an NFT command referring to node DENVER, the project-programmer number (2,155), the password (MELBA), and account (3N4) will be used to gain access to node DENVER.

Note that although the password is displayed as you type it, NETACT uses a simple two-way encryption to write it to the file NETACT.DAT. A malicious user cannot determine your passwords by displaying the file NETACT.DAT from your account.

To redefine entries for a node, use the DEFINE command again.

Example:

```
NETACT>DEFINE
Node name                          ? DENVER
User:              <2,155>         ? 2,154
Password:          <MELBA>         ?
Account:           <3N4>           ?
```

When you type a node name for which NETACT already has a definition, the current entries are displayed in angle brackets. You can change an entry by typing new values, as in the previous example. Depress the RETURN key to leave the entry unchanged. (Thus, the password MELBA is retained in the new entry created by the preceding example.) Depress the ESCAPE key to delete the entry entirely. (The account 3N4 is simply deleted in the preceding example; no new account is entered.)

C.2.2 EXIT Command

You can terminate NETACT by typing EXIT in response to the prompt. You can also terminate NETACT at any point by typing a CTRL/Z. If this is done while NETACT is waiting for response to a node name, user, password, or account prompt, the last command typed is not processed.

Example:

```
NETACT>EXIT

Ready
```

or

```
NETACT>DEFINE
Node name                       ? DALLAS
User:           <2,154>         ? 2,117
Password:       <MELBA>         ? ^Z

Ready
```

In the last example, the entire DEFINE command for node DALLAS is ignored; the original entries are retained.

C.2.3 HELP Command

The HELP command displays information. It has two forms.

```
NETACT>HELP
```

or

```
NETACT>?
```

C.2.4 LIST Command

The LIST command displays current definitions.

Example:

```
NETACT>LIST

Node            BOSTON
User            1,117
Password        AGHAST
Account

Node            DENVER
User            2,154
Password        MELBA
Account
```

C.2.5 PURGE Command

The PURGE command eliminates a current entry.

Example:

```
NETACT>PURGE
Node name                   ? DALLAS
Node DALLAS purged from file.
```

C.3 NETACT Error Messages

Field too long

You have entered too many characters. The following limits apply:

Node name is limited to 6 characters.
User is limited to 16 characters.
Password is limited to 8 characters.
Account is limited to 16 characters.

Illegal command – *command* Type "Help" for help

The command you just typed is not a valid NETACT command. Typing "HELP" will display the commands acceptable to NETACT.

Invalid node name

The nodename field cannot be null.

Node *nodename* not defined.

You are trying to purge a node definition that does not exist. Check the node name you typed for misspelling. Issue a LIST command to see current definitions.

Appendix D
Error Messages Generated by RMS and DAP

This appendix contains non-NFT error messages that can appear while using the Network File Transfer (NFT) program. These messages can appear for conditions diagnosed by the Record Management Services (RMS), used by NFT and FAL to do local input and output, or for violations of the Data Access Protocol (DAP), used to do remote file transfers across the network.

D.1 NFT Error Messages from RMS–11

NFT prints the following error message in response to detected RMS–11 errors:

?NFT – RMS ERROR = *nnnnnn*

The errors that can occur when you use NFT should be fairly understandable, such as error 176440, "File not found." However, all possible RMS errors are listed here, and some descriptions will be obscure to all but those familiar with RMS. If you get one of these errors repeatedly, call a software specialist and report the error.

NOTE

The following list reflects errors generated by RMS Version 1.8 as described in the *RMS–11 MACRO–11 Reference Manual.*

Octal Value (*nnnnnn*)	Description
177760	Operation aborted: Stack save area exhausted or in-core data structures corrupted.
177740	Files–11 ACP could not access the file.
177720	File activity precludes action (e.g., attempting to close a file with outstanding asynchronous record operation).
177700	Bad area identification number (AID) field in allocation XAB (i.e., out of sequence).
177660	Invalid value in alignment boundary type (ALN) field of allocation XAB.

(continued on next page)

Octal Value (*nnnnnn*)	Description
177640	Value in allocation quantity (ALQ) field in FAB (or allocation XAB) exceeds maximum or, during an explicit $EXTEND operation, equals zero.
177620	Records in a field on ANSI-labeled magnetic tape are variable length but not in ANSI-D format.
177600	Invalid value in allocation options (AOP) field in allocation XAB.
177560	Invalid operation at AST level: Attempting to issue a synchronous operation from an asynchronous record operation completion routine.
177540	Read error on file header attributes.
177530	Invalid file ID.
177520	Write error on file header attributes.
177500	Bucket size (BKS) field in FAB contains value exceeding maximum.
177460	Bucket size (BKZ) field in allocation XAB contains value exceeding maximum.
177440	Block length (BLN) field in a FAB, RAB, or XAB is incorrect.
177430	Beginning of file detected on $SPACE operation to magnetic tape file.
177420	Private buffer pool address not a double word boundary.
177400	Private buffer pool size not a multiple of 4.
177360	Internal error detected in RMS–11. No recovery possible; contact a software specialist.
177340	Cannot connect RAB (i.e., only one record access stream permitted for sequential files).
177320	$UPDATE attempting to change a key field that does not have the change attribute.
177300	Index file bucket check-byte mismatch. The bucket has been corrupted. Recovery can be attempted by: 1. Moving disk pack to another device and trying the process again. 2. Recreating file using either RMSIFL or RMSONV utility. 3. Restoring file from last backup.

(continued on next page)

Octal Value (*nnnnnn*)	Description
177260	$CLOSE function failed.
177240	Invalid OOD field in XAB or XAB type is invalid for the organization or operation.
177220	Files–11 ACP could not create file.
177200	No current record: Operation not immediately preceded by a successful $GET or $FIND.
177160	Files–11 ACP deaccess error during $CLOSE.
177140	Invalid area number in DAN field of key definition XAB.
177120	Record accessed by RFA access mode has been deleted.
177100	1. Syntax error in device name. 2. No such device. 3. Inappropriate device for operation (e.g., attempting to create an indexed file on magnetic tape).
177070	Files–11 ACP could not write bucket. RMS–11 deferred the I/O operation until it needed the I/O buffer for another bucket because the user program specified deferred writes.
177060	Syntax error in directory name.
177040	Dynamic memory exhausted: Insufficient space in central space pool or private buffer pool.
177020	Directory not found.
177000	Device not ready.
176770	Device positioning error.
176760	DTP field invalid (STV = @XAB).
176740	Duplicate key detected, "duplicates allowed" attribute not set for one or more key fields.
176720	Fiels–11 ACP enter function failed.
176700	Environment error: Operation or file organization not specified in ORG$ macro.
176640	Expanded string area in NAM block too short.
176630	File expiration date not reached.
177620	File extend failure.
176600	Not a valid FAB: BID field does not contain FB$BID.

(continued on next page)

Octal Value (*nnnnnn*)	Description
176560	1. Record operation attempted was not declared in FAC field of FAB at open time. 2. Invalid contents in FAC field. 3. FB$PUT not present in FAC for $CREATE operation.
176530	Invalid file ID.
176520	Invalid combination of values in FLG field of key definition XAB (e.g., "no duplicates" and "changeable keys").
176500	File locked by another user: Cannot access the file because sharing specification cannot be met.
176460	Files–11 ACP $FIND function failed.
176440	File not found during $OPEN.
176420	Syntax error in file name.
176400	Invalid file options.
176370	System error during FNA/DNA string parse (STV = system error code).
176360	Device full: Cannot create or extend file.
176340	Invalid area number in IAN field of key definition XAB.
176320	Index not initialized. This code can only occur in the STV field when STS contains ER$RNF.
176300	Invalid IFI field in FAB.
176260	Maximum number (254) of key definition or allocation XABs exceeded or multiple summary, protection, or date XABs present during operation.
176240	$INIT or $INITIF macro call never issued.
176220	Invalid operation. Examples include: 1. Attempting a $TRUNCATE operation to a nonsequential file. 2. Attempting an $ERASE or $EXTEND operation to a magnetic tape file. 3. Issuing a block mode operation (e.g., $READ or $WRITE) to a stream not connected for block operations. 4. Issuing a record operation (e.g., $GET, $PUT) to a stream connected for block mode operations.

(continued on next page)

Octal Value (*nnnnnn*)	Description
176200	Invalid record encountered in sequential file: Invalid count field.
176160	Invalid internal stream identifier (ISI) field in RAB (field may have been altered by user) or $CONNECT never issued for stream.
176140	Key buffer address (KBF) field equals 0.
176120	Record identifier (i.e., the 4-byte location addressed by KBF) for random operation to relative file is 0 or negative.
176100	Invalid key of reference (KRF) in RAB: 1. During random $GET or $FIND operation, or 2. During $CONNECT or $REWIND. In this case, ER$KRF is returned for the first record operation following the $CONNECT or $REWIND.
176060	Key size equals zero or too large (indexed file) or not equal to 4 (relative file).
176040	Invalid area number in LAN field of key definition XAB.
176020	Magnetic tape is not labeled in accordance with ANSI standards.
176000	Logical channel busy.
175760	Invalid value in logical channel number (LCH) field of FAB.
175750	Attempt to extend an area containing an unused extent.
175740	Invalid value in LOC field of allocation XAB.
175720	In-core data structures (e.g., I/O buffers) corrupted. This code can only occur in the STV field when STS contains ER$ABO.
175700	Files–11 ACP could not mark file for deletion.
175660	1. Maximum record number field contains a negative value during $CREATE of relative file. 2. Record identifier (pointed to by KBF) for random operation to relative file exceeds maximum record number specified when file created.
175640	Maximum record size (MRS) field contains 0 during $CREATE operation and: 1. Record format is fixed, or 2. File organization is relative.

(continued on next page)

Octal Value (*nnnnnn*)	Description
175620	Odd address in Name Block address (NAM) field in FAB on $OPEN, $CREATE, or $ERASE.
175600	Not at end-of-file: Attempting a $PUT operation to a sequential file when stream is not positioned to EOF.
175560	Cannot allocate internal index descriptor: Insufficient room in space pool while attempting to open an indexed file.
175540	No primary key definition XAB present during $CREATE of indexed file.
175520	$OPEN function failed.
175500	XABs in chain not incorrect order. 1. Allocation or key definition XABs not in ascending (or densely ascending) order. 2. XAB of another type intervenes in key definition or allocation XAB sub-chain.
175460	Invalid value in file organization (ORG) field of FAB.
175440	Error in file's prologue: File is corrupted; recovery can be attempted by: 1. Moving disk pack to another device. 2. Recreating file using either the RMSIFL or RMSONV utility. 3. Restoring file from latest backup.
175420	Key position (POS) field in key definition XAB contains a value exceeding maximum record size.
175400	File header contains bad date and time information (retrieved by RMS–11 because a date and time XAB is present during an $OPEN or $DISPLAY operation). File may be corrupted.
175360	Privilege violation: Access to the file denied by the operating system.
175340	Not a valid RAB: BID field does not contain RB$BID.
175320	1. Invalid values in record access mode (RAC) field of RAB. 2. Illogical value in RAC field (e.g., RB$KEY with a sequential file).
175300	1. Invalid values in record attributes (RAT) field of FAB during $CREATE. 2. Illogical combination of attributes (e.g., FB$CR and FB$FTN) in RAC field during $CREATE.

(continued on next page)

Octal Value (*nnnnnn*)	Description
175260	Record address (RBF) field in RAB contains an odd address (block mode access only).
175240	Files–11 ACP error: 1. In record processing – read failure on file block. 2. In block I/O – VBN = 0.
175220	Record already exists: During a $PUT operation in random mode to a relative file, an existing record found in the target record position.
175200	Invalid RFA in RFA field of RAB during RFA access.
175160	1. Invalid record format in RFM field of FAB during $CREATE. 2. Specified record format is invalid for file organization.
175140	Target bucket locked by another task or another stream in the same program.
175120	Files–11 ACP $REMOVE function failed.
175100	Record identified by KBF/KSZ fields of RAB for random $GET or $FIND operation does not exist in relative or indexed file (for indexed files only, STV may contain ER$IDX). Record may never have been written or may have been deleted.
175060	$FREE operation issued but no bucket was locked by stream.
175040	Record options (ROP) field contains invalid values or illogical combination of values.
175020	Error while reading prologue.
175000	Invalid RRV record encountered in indexed file. File may be corrupted.
174760	Record stream active: In asynchronous environment, attempting to issue a record operation to a stream that has a request outstanding.
174740	Record size specified in RSZ of RAB during $PUT or $UPDATE is invalid: 1. RSZ equals zero. 2. RSZ exceeds maximum record size (MRS) specified when file created. 3. RSZ not equal to size of Current Record for $UPDATE operation to a sequential file on disk.

(continued on next page)

Octal Value (*nnnnnn*)	Description
	4. RSZ does not equal MRS (for fixed format records). 5. RSZ not large enough to contain Primary Key of indexed file.
174720	Record too big for user's buffer: RMS–11 could not move entire record retrieved by $GET operation to user work area (UBF/USZ). Note that this error does not destroy the current context of the stream. Rather, the stream's context is updated as if the operation had been completely successful and as much of the record as possible is moved to user buffer.
174710	RRV update error on insert.
174700	During $PUT operation, key of record to be written is not equal to or greater than key of previous record (and RAC field contains RB$SEQ).
174660	Illogical value in SHR field of FAB (e.g., FB$WRI specified for sequential file).
174640	Invalid SIZ field in key definition XAB during $CREATE (e.g., specified size exceeds maximum record size).
174620	During asynchronous record operation, RMS–11 has found that the stack is too big to be saved. This code can only occur in the STV field when STS contains ER$ABO.
174600	System directive error.
174560	Index tree error: Indexed file is corrupted.
174540	Syntax error in file type (e.g., more than 3 characters specified.
174520	Invalid address in user buffer (UBF) field of RAB: 1. UBF contains 0, or 2. UBF not word aligned (for block mode access only).
174500	Invalid user buffer size (USZ) field in RAB (i.e., USZ contains 0).
174440	Invalid VOL field in allocation XAB (i.e., VOL does not contain 0).
174430	Wildcard encountered during FNA/DNA string parse.
174420	File write error.
174410	Device is write locked.

(continued on next page)

Octal Value (*nnnnnn*)	Description
174400	Error while writing prologue.
174360	XAB field in FAB (or NXT field in XAB) contains an odd address.
174340	Extraneous field detected during FNA/DNA string parse.

D.2 NFT Error Messages from DAP

NFT prints the following error message in response to an error in NFT/FAL communication, violating the Data Access Protocol:

?NFT – DAP ERROR = *aannnn*

The *aa* value gives the macro or functional group reason for the error. The *nnnn* value gives the micro, or specific, reason for the error.

Possible values for the macro (*aa*) field are listed below.

NOTE

This list reflects the errors generated as the result of a violation of DAP Version 5.6, as described in the *DECnet DIGITAL Network Architecture*, "Data Access Protocol Functional Specification," released in October, 1980.

D.2.1 Macro (*aa*) Field Values

0	Pending	Operation in progress
1	Successful	Returns information that indicates success.
2	Unsupported	This implemation of DAP does not support the specified request.
3		Reserved
4	File Open	Errors that occur before a file is successfully opened.
5	Transfer Error	Errors that occur after opening a file and before closing that file.
6	Transfer Warning	For operations on open files, indicates the operation completed, but not with complete success.
7	Access Termination	Errors associated with terminating access to a file.

(continued on next page)

10	Format	Error in parsing a message. Format is not correct.
11	Invalid	Field of message is invalid. For example, bits that are meant to be mutually exclusive are set, an undefined bit is set, a field value is out of range, or an invalid string is in a field.
12	Sync	DAP message received out of synchronization.
13-15		Reserved.
16-17		User-defined STATUS message MACCODE field.

D.2.2 Micro (*nnnn*) Field Values

Micro (*nnnn*) values for use with macro (*aa*) values of 2, 10, and 11 octal are listed below. These refer to the macro Unsupported, Format, and Invalid categories.

NOTE

Micro (nnnn) Format: Bits 6-11 specify the DAP message type number. Bits 0-5 specify the DAP message field number.

Miscellaneous errors:

00	00	Unspecified DAP message error (catch all)
00	10	DAP message type field (TYPE) error

CONFIGURATION message errors by field:

01	00	Unknown field
01	10	DAP message flags field (FLAGS)
01	11	Data stream identification field (STREAMID)
01	12	Length field (LENGTH)
01	13	Length extension field (LEN256)
01	14	Bit count field (BITCNT)
01	20	Buffer size field (BUFSIZ)
01	21	Operating system type field (OSTYPE)
01	22	File system type field (FILESYS)
01	23	DAP version number field (VERNUM)
01	24	ECO version number field (ECONUM)
01	25	USER protocol version number field (USRNUM)
01	26	DEC software release number field (SOFTVER)
01	27	User software release number field (USRSOFT)
01	30	System capabilities field (SYSCAP)

ATTRIBUTES message errors by field:

02	00	Unknown field
02	10	DAP message flags field (FLAGS)
02	11	Data stream identification field (STREAMID)
02	12	Length field (LENGTH)
02	13	Length extension field (LEN256)
02	14	Bit count field (BITCNT)
02	15	System specific field (SYSPEC)
02	20	Attributes menu field (ATTMENU)
02	21	Data type field (DATATYPE)
02	22	File organization field (ORG)
02	23	Record format field (RFM)
02	24	Record attributes field (RAT)
02	25	Block size field (BLS)
02	26	Maximum record size field (MRS)
02	27	Allocation quantity field (ALQ)
02	30	Bucket size field (BKS)
02	31	Fixed control area size field (FSZ)
02	32	Maximum record number field (MRN)
02	33	Run-time system field (RUNSYS)
02	34	Default extension quantity field (DEQ)
02	35	File options field (FOP)
02	36	Byte size field (BSZ)
02	37	Device characteristics field (DEV)
02	40	Spooling device characteristics field (SDC)
02	41	Longest record length field (LRL)
02	42	Highest virtual block allocated field (HBK)
02	43	End of file block field (EBK)
02	44	First free byte field (FFB)
02	45	Starting LBN for contiguous file (SBN)

ACCESS message errors by field:

03	00	Unknown field
03	10	DAP message flags field (FLAGS)
03	11	Data stream identification field (STREAMID)
03	12	Length field (LENGTH)
03	13	Length extension field (LEN256)
03	14	Bit count field (BITCNT)
03	15	System specific field (SYSPEC)
03	20	Access function field (ACCFUNC)
03	21	Access options field (ACCOPT)
03	22	File specification field (FILESPEC)
03	23	File access field (FAC)
03	24	File sharing field (SHR)

03	25	Display attributes request field (DISPLAY)
03	26	File password field (PASSWORD)

CONTROL message errors by field:

04	00	Unknown field
04	10	DAP message flags field (FLAGS)
04	11	Data stream identification field (STREAMID)
04	12	Length field (LENGTH)
04	13	Length extension field (LEN256)
04	14	Bit count field (BITCNT)
04	15	System specific field (SYSPEC)
04	20	Control function field (CTLFUNC)
04	21	Control menu field (CTLMENU)
04	22	Record access field (RAC)
04	23	Key field (KEY)
04	24	Key of reference field (KRF)
04	25	Record options field (ROP)
04	26	Hash code field (HSH)
04	27	Display attributes request field (DISPLAY)

CONTINUE TRANSFER message errors by field:

05	00	Unknown field
05	10	DAP message flags field (FLAGS)
05	11	Data stream identification field (STREAMID)
05	12	Length field (LENGTH)
05	13	Length extension field (LEN256)
05	14	Bit count field (BITCNT)
05	15	System specific field (SYSPEC)
05	20	Continue transfer function (CONFUNC)

ACKNOWLEDGE message errors by field:

06	00	Unknown field
06	10	DAP message flags field (FLAGS)
06	11	Data stream identification field (STREAMID)
06	12	Length field (LENGTH)
06	13	Length extension field (LEN256)
06	14	Bit count field (BITCNT)
06	15	System specific field (SYSPEC)

ACCESS COMPLETE message errors by field:

07	00	Unknown field
07	10	DAP message flags field (FLAGS)
07	11	Data stream identification field (STREAMID)
07	12	Length field (LENGTH)
07	13	Length extension field (LEN256)

07	14	Bit count field (BITCNT)
07	15	System specific field (SYSPEC)
07	20	Access complete function field (CMPFUNC)
07	21	File options field (FOP)
07	22	Checksum field (CHECK)

DATA message errors by field:

10	00	Unknown field
10	10	DAP message flags field (FLAGS)
10	11	Data stream identification field (STREAMID)
10	12	Length field (LENGTH)
10	13	Length extension field (LEN256)
10	14	Bit count field (BITCNT)
10	15	System specific field (SYSPEC)
10	20	Record number field (RECNUM)
10	21	File data field (FILEDATA)

STATUS message errors by field:

11	00	Unknown field
11	10	DAP message flags field (FLAGS)
11	11	Data stream identification field (STREAMID)
11	12	Length field (LENGTH)
11	13	Length extension field (LEN256)
11	14	Bit count field (BITCNT)
11	15	System specific field (SYSPEC)
11	20	Macro status code field (MACCODE)
11	21	Micro status code field (MICCODE)
11	22	Record file address field (RFA)
11	23	Record number field (RECNUM)
11	24	Secondary status field (STV)

KEY DEFINITION message errors by field:

12	00	Unknown field
12	10	DAP message flags field (FLAGS)
12	11	Data stream identification field (STREAMID)
12	12	Length field (LENGTH)
12	13	Length extension field (LEN256)
12	14	Bit count field (BITCNT)
12	15	System specific field (SYSPEC)
12	20	Key definition menu field (KEYMENU)
12	21	Key option flags field (FLG)
12	22	Data bucket fill quantity field (DFL)
12	23	Index bucket fill quantity field (IFL)
12	24	Key segment repeat count field (SEGCNT)

12	25	Key segment position field (POS)
12	26	Key segment size field (SIZ)
12	27	Key of reference field (REF)
12	30	Key name field (KNM)
12	31	Null key character field (NUL)
12	32	Index area number field (IAN)
12	33	Lowest level area number field (LAN)
12	34	Data level area number field (DAN)
12	35	Key data type field (DTP)
12	36	Root VBN for this key field (RVB)
12	37	Hash algorithm value field (HAL)
12	40	First data bucket VBN field (DVB)
12	41	Data bucket size field (DBS)
12	42	Index bucket size field (IBS)
12	43	Level of root bucket field (LVL)
12	44	Total key size field (TKS)
12	45	Minimum record size field (MRL)

ALLOCATION message errors by field:

13	00	Unknown field
13	10	DAP message flags field (FLAGS)
13	11	Data stream identification field (STREAMID)
13	12	Length field (LENGTH)
13	13	Length extension field (LEN256)
13	14	Bit count field (BITCNT)
13	15	System specific field (SYSPEC)
13	20	Allocation menu field (ALLMENU)
13	21	Relative volume number field (VOL)
13	22	Alignment options field (ALN)
13	23	Allocation options field (AOP)
13	24	Starting location field (LOC)
13	25	Related file identification field (RFI)
13	26	Allocation quantity field (ALQ)
13	27	Area identification field (AID)
13	30	Bucket size field (BKZ)
13	31	Default extension quantity field (DEQ)

SUMMARY message errors by field:

14	00	Unknown field.
14	10	DAP message flags field (FLAGS)
14	11	Data stream identification field (STREAMID)
14	12	Length field (LENGTH)
14	13	Length extension field (LEN256)
14	14	Bit count field (BITCNT)
14	15	System specific field (SYSPEC)
14	20	Summary menu field (SUMENU)
14	21	Number of keys field (NOK)

14	22	Number of areas field (NOA)
14	23	Number of record descriptors field (NOR)
14	24	Prologue version number (PVN)

DATE AND TIME message errors by field:

15	00	Unknown field
15	10	DAP message flags field (FLAGS)
15	11	Data stream identification field (STREAMID)
15	12	Length field (LENGTH)
15	13	Length extension field (LEN256)
15	14	Bit count field (BITCNT)
15	15	System specific field (SYSPEC)
15	20	Date and time menu field (DATMENU)
15	21	Creation date and time field (CDT)
15	22	Last update date and time field (RDT)
15	23	Deletion date and time field (EDT)
15	24	Revision number field (RVN)

PROTECTION message errors by field:

16	00	Unknown field
16	10	DAP message flags field (FLAGS)
16	11	Data stream identification field (STREAMID)
16	12	Length field (LENGTH)
16	13	Length extension field (LEN256)
16	14	Bit count field (BITCNT)
16	15	System specific field (SYSPEC)
16	20	Protection menu field (PROTMENU)
16	21	File owner field (OWNER)
16	22	System protection field (PROTSYS)
16	23	Owner protection field (PROTOWN)
16	24	Group protection field (PROTGRP)
16	25	World protection field (PROTWLD)

NAME message errors by field:

17	00	Unknown field
17	10	DAP message flages field (FLAGS)
17	11	Data stream identification field (STREAMID)
17	12	Length field (LENGTH)
17	13	Length extension field (LEN256)
17	14	Bit count field (BITCNT)
17	15	System specific field (SYSPEC)
17	20	Name type field (NAMETYPE)
17	21	Name field (NAMESPEC)

ACCESS CONTROL LIST message errors by field: (Reserved for future use)

20	00	Unknown field
20	10	DAP message flags field (FLAGS)
20	11	Data stream indentification field (STREAMID)
20	12	Length field (LENGTH)
20	13	Length extension field (LEN256)
20	14	Bit count field (BITCNT)
20	15	System specific field (SYSPEC)
20	20	Access control list repeat count field (ACLCNT)
20	21	Access control list entry field (ACL)

Micro (*nnnn*) values for use with macro (*aa*) values of 0, 1, 4, 5, 6, and 7 octal, are listed below.

NOTE

Micro (nnnn) Format: Bits 0-11 contains error code number. Symbolic status codes, where supplied, refer to the corresponding RMS status codes. They are included here for ease of reference only – they have no meaning for DAP.

0000		Unspecified error
0001	ER$ABC	Operation aborted (STV = ER$STK/MAP)
0002	ER$ACC	F11-ACP could not access file (STV = sys err code)
0003	ER$ACT	File activity precludes operation
0004	ER$AID	Bad area ID
0005	ER$ALN	Alignment options error
0006	ER$ALQ	Allocation quantity too large or equal to 0
0007	ER$ANI	Not ANSI-D format
0010	ER$AOP	Allocation options error
0011	ER$AST	Invalid (i.e. synch) operation at AST level
0012	ER$ATR	Attribute read error
0013	ER$ATW	Attribute write error
0014	ER$BKS	Bucket size too large
0015	ER$BKZ	Bucket size too large
0016	ER$BLN	BLN length error
0017	ER$BOF	Beginning of file detected
0020	ER$BPA	Private pool address not multiple of 4
0021	ER$BPS	Private pool size not multiple of 4
0022	ER$BUG	Internal RMS error condition detected
0023	ER$CCR	Cannot connect RAB
0024	ER$CHG	$UPDATE changed a key without having attribute of XB$CHG set
0025	ER$CHK	Bucket format check-byte failure
0026	ER$CLS	RSTS/E close function failed
0027	ER$COD	Invalid or unsupported COD field

0030	ER$CRE	F11-ACP could not create file (STV = sys err code)
0031	ER$CUR	No current record (operation not preceded by $GET/ $FIND)
0032	ER$DAC	F11-ACP deaccess error during close
0033	ER$DAN	Date area number invalid
0034	ER$DEL	RFA-Accessed record was deleted
0035	ER$DEV	Bad device or inappropriate device type
0036	ER$DIR	Error in directory name
0037	ER$DME	Dynamic memory exhausted
0040	ER$DNF	Directory not found
0041	ER$DNR	Device not ready
0042	ER$DPE	Device has positioning error
0043	ER$DTP	DTP field invalid
0044	ER$DUP	Duplicate key detected, XB$DUP not set
0045	ER$ENT	RSX-F11ACP enter function failed
0046	ER$ENV	Operation not selected in ORG$ macro
0047	ER$EOF	End of file
0050	ER$ESS	Expanded string area too short
0051	ER$EXP	File expiration date not yet reached
0052	ER$EXT	File extend failure
0053	ER$FAB	Not a valid FAB (BID field not = FB$BID)
0054	ER$FAC	Invalid FAC for REC-OP: 0, or FB$PUT not set for $CREATE
0055	ER$FEX	File already exists
0056	ER$FID	Invalid file ID
0057	ER$FLG	Invalid flag-bits combination
0060	ER$FLK	File is locked by other user
0061	ER$FND	RSX-F11ACP find function failed
0062	ER$FNF	File not found
0063	ER$FNM	Error in file name
0064	ER$FOP	Invalid file options
0065	ER$FUL	Device/file full
0066	ER$IAN	Index area number invalid
0067	ER$IFI	Invalid IFI value or unopened file
0070	ER$IMX	Maximum NUM (254) areas/key XABS exceeded
0071	ER$INI	$INIT macro never issued
0072	ER$IOP	Operation unknown or invalid for file organization
0073	ER$IRC	Invalid record encountered (sequential files only)
0074	ER$ISI	Invalid ISI value on unconnected RAB
0075	ER$KBF	Bad key buffer address (KBF = 0)
0076	ER$KEY	Invalid key field (KEY = 0 or negative)
0077	ER$KRF	Invalid key-of-reference ($GET/$FIND)
0100	ER$KSZ	Key size too large (IDX/NOT = :(REL))
0101	ER$LAN	Lowest level index area number invalid
0102	ER$LBL	Not ANSI labeled tape
0103	ER$LBY	Logical channel busy
0104	ER$LCH	Logical channel number too large

0105	ER$LEX	Logical extend error, prior extend still valid
0106	ER$LOC	LOC field invalid
0107	ER$MAP	Buffer mapping error
0110	ER$MKD	F11ACP could not mark file for deletion
0111	ER$MRN	MRN value = negative or relative key > MRN
0112	ER$MRS	MRS value = 0 for fixed length records or 0 for relative files
0113	ER$NAM	NAM block address invalid (NAM = 0 or not accessible)
0114	ER$NEF	Not positioned to EOF (sequential files only)
0115	ER$NID	Cannot allocate internal index descriptor
0116	ER$NPK	No primary key defined for indexed file
0117	ER$OPN	RSTS/E open function failed
0120	ER$ORD	XABs not in correct order
0121	ER$ORG	Invalid file organization value
0122	ER$PLG	Error in file's prologue (reconstruct file)
0123	ER$POS	POS field invalid (POS>MRS, STV = XAB indicator)
0124	ER$PRM	Bad file date field retrieved
0125	ER$PRV	Privilege violation (OS denies access)
0126	ER$RAB	Not a valid RAB (BID field not = RB$BID)
0127	ER$RAC	Invalid RAC value
0130	ER$RAT	Invalid record attributes
0131	ER$RBF	Invalid record buffer address (odd, or not word-aligned if BLK-IO)
0132	ER$RER	File read error (STV = sys err code)
0133	ER$REX	Record already exists
0134	ER$RFA	Bad RFA value (RFA = 0)
0135	ER$RFM	Invalid record format
0136	ER$RLK	Tanget bucket locked by another stream
0137	ER$RMV	RSX-F11ACP remove function failed
0140	ER$RNF	Record not found
0141	ER$RNL	Record not locked
0142	ER$ROP	Invalid record options
0143	ER$RPL	Error while reading prologue
0144	ER$RRV	Invalid RRV record encountered
0145	ER$RSA	RAB stream currently active
0146	ER$RSZ	Bad record size (RSZ>MRS, or not=MRS if fixed length records)
0147	ER$RTB	Record too big for user's buffer
0150	ER$SEQ	Primary key out of sequence (RAC = RB$SEQ for $PUT)
0151	ER$SHR	SHR field invalid for file (cannot share sequential files)
0152	ER$SIZ	SIZ field invalid
0153	ER$STK	Stack too big for save area
0154	ER$SYS	System directive error
0155	ER$TRE	Index tree error
0156	ER$TYP	Error in file type; extension on FNS too big

0157	ER$UBF	Invalid user buffer address (0, odd, or if BLK-IO not word-aligned)
0160	ER$USZ	Invalid user buffer size (USZ = 0)
0161	ER$VER	Error in version number
0162	ER$VOL	Invalid volume number
0163	ER$WER	File write error (STV = sys err code)
0164	ER$WLK	Device is write locked
0165	ER$WPL	Error while writing prologue
0166	ER$XAB	Not a valid XAB (@XAB = odd, STV = XAB indicator)
0167	BUGDDI	Default directory invalid
0170	CAA	Cannot access argument list
0171	CCF	Cannot close file
0172	CDA	Cannot deliver AST
0173	CHN	Channel assignment failure (STV = sys err code)
0174	CNTRLO	Terminal output ignored due to Control-O
0175	CNTRLY	Terminal input aborted due to Control-Y
0176	DNA	Default file name string address error
0177	DVI	Invalid device ID field
0200	ESA	Expanded string address error
0201	FNA	File name string address error
0202	FSZ	FSZ field invalid
0203	IAL	Invalid argument list
0204	KFF	Known file found
0205	LNE	Logical name error
0206	NOD	Node name error
0207	NORMAL	Operation successful
0210	CK_DUP	Record inserted had duplicate key
0211	CK_IDX	Index update error occurred; record inserted
0212	CK_RLK	Record locked buy read anyway
0213	CK_RRV	Record inserted in primary, buy may not be accessible by secondary keys or RFA
0214	CREATE	File was created, but not opened
0215	PBF	Bad prompt buffer address
0216	PNDING	Asynchronous operation pending completion
0217	QUO	Quoted string error
0220	RHB	Record header buffer invalid
0221	RLF	Invalid related file
0222	RSS	Invalid resultant string size
0223	RST	Invalid resultant string address
0224	SQO	Operation not sequential
0225	SUC	Operation successful
0226	SPRSED	Created file superseded existing version
0227	SYN	File name syntax error
0230	TMO	Time-out period expired
0231	ER$BLK	FB$BLK record attribute not supported
0232	ER$BSZ	Bad byte size

0233	ER$CDR	Canot disconnect RAB
0234	ER$CGJ	Cannot get JFN for file
0235	ER$COF	Cannot open file
0236	ER$JFN	Bad JFN value
0237	ER$PEF	Cannot position to end of file
0240	ER$TRU	Cannot truncate file
0241	ER$UDF	File is currently in an undefined state; access is denied
0242	ER$XCL	File must be opened for exclusive access
0243		Directory full
0244		Handler not in system
0245		Fatal hardware error
0246		Attempt to write beyond EOF
0247		Hardware option not present
0250		Device not attached
0251		Device already attached
0252		Device not attachable
0253		Sharable resource in use
0254		Invalid overlay request
0255		Block check or CRC error
0256		Caller's nodes exhausted
0257		Index file full
0260		File header full
0261		Accessed for write
0262		File header checksum failure
0263		Attribute control list error
0264		File already accessed on LUN
0265		Bad tape format
0266		Invalid operation on file descriptor block
0267		2 different devices specified on a rename
0270		New file name specified in rename already in use
0271		Cannot rename old system
0272		File already open
0273		Parity error on device
0274		End of volume detected
0275		Data over-run
0276		Bad block on device
0277		End of tape detected
0300		No buffer space for file
0301		File exceeds allocated space; no blocks
0302		Specified task not installed
0303		Unlock error
0304		No file accessed on LUN
0305		Send/receive failure
0306	SPL	Spool or submit command file failure
0307	NMF	No more files
0310	CRC	DAP file transfer checksum error
0311		Quota exceeded

0312	BUGDAP	Internal network error condition detected
0313	CNTRLC	Terminal input aborted due to Control-C
0314	DFL	Data bucket fill size > bucket size in XAB
0315	ESL	Invalid expanded string length
0316	IBF	Invalid bucket format
0317	IBK	Bucket size of LAN not = IAN in XAB
0320	IDX	Index not initialized
0321	IFA	Invalid file attributes (corrupt file header)
0322	IFL	Index bucket fill size > bucket size in XAB
0323	KNM	Key name buffer not readable or writeable in XAB
0324	KSI	Index bucket will not hold two keys for key of reference
0325	MBC	Multi-buffer count invalid (negative value)
0326	NET	Network operation failed at remote node
0327	CK_ALK	Record is already locked
0330	CK_DEL	Deleted record successfully accessed
0331	CK_LIM	Retrieved record exceeds specified key value
0332	CK_NOP	Key XAB not filled in
0333	CK_RNF	Nonexistent record successfully accessed
0334	PLV	Unsupported prologue version
0335	REF	Invalid key-of-reference in XAB
0336	RSL	Invalid resultant string length
0337	RVU	Error updating RRVs; some paths to data may be lost
0340	SEG	Data types other than string limited to one segment in XAB
0341		Reserved
0342	SUP	Operation not supported over network
0343	WBE	Error on write behind
0344	WLD	Invalid wildcard operation
0345	WSF	Working set full (cannot lock buffers in working set)
0346		Directory listing; error in reading volume-set name, directory name, or file name
0347		Directory listing; error in reading file attributes
0350		Directory listing; protection violation in trying to read the volume-set, directory or file name
0351		Directory listing; protection violation in trying to read file attributes
0352		Directory listing; file attributes do not exist
0353		Directory listing; unable to recover directory list after Continue Transfer (Skip)
0354	SNE	Sharing not enabled
0355	SPE	Sharing page count exceeded
0356	UPI	UPI bit not set when sharing with BRO set
0357	ACS	Error in access control string (poor man's route through error)
0360	TNS	Terminator not seen

0361	BES	Bad escape sequence
0362	PES	Partial escape sequence
0363	WCC	Invalid wildcard context value
0364	IDR	Invalid directory rename operation
0365	STR	User structure (FAB/RAB) became invalid during operation
0366	FTM	Network file transfer mode precludes operation
6000 to 7777		User defined errors

Micro (*nnnn*) values for use with macro (*aa*) value of 12 octal are listed below. This refers to the macro Synchronization category.

NOTE

Micro (nnnn) Format: Bits 0-11 contains message type number.

0000	Unknown
0001	Configuration
0002	Attributes
0003	Access
0004	Control
0005	Continue transfer
0006	Acknowledge
0007	Access complete
0010	Data
0011	Status
0012	Key definition attributes
0013	Allocation attributes extension
0014	Summary attributes extension
0015	Date and time attributes extension
0016	Protection attributes extension
0017	Name
0020	Access control list extended attributes

INDEX

A

B

C

D

E

O

P

Q

DECnet/E
Guide to User Utilities
AA-H504B-TC

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or
Country

— — — — —Do Not Tear - Fold Here and Tape —

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE DOCUMENTATION
1925 ANDOVER STREET TW/E07
TEWKSBURY, MASSACHUSETTS 01876

— — — — Do Not Tear - Fold Here and Tape — — — — — — — — — — — — — — — — — — —

Cut Along Dotted Line