# DECnet–20 User's Guide

AA–J679A–TM

**December 1982**

This manual contains user information for DECnet–20, version 3, a product that together with the TOPS–20 operating system, provides the DECSYSTEM–20 family of computers with a communications interface to DIGITAL's corporate network, DECnet. This is a new manual.

*TOPS–20 DECnet–20 Programmer's Guide and Operations Manual*, Order Number AA–5091A–TM, is to be used for DECnet–20, version 2 users.

| | |
|---|---|
| **OPERATING SYSTEM:** | TOPS–20 V5.1<br>GALAXY V4.2 |
| **SOFTWARE:** | DECnet–20 V3 |

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center. Outside the United States, orders should be directed to the nearest DIGITAL Field Sales Office or representative.

**Northeast/Mid–Atlantic Region**

Digital Equipment Corporation
PO Box CS2008
Nashua, New Hampshire 03061
Telephone:(603)884–6660

**Central Region**

Digital Equipment Corporation
Accessories and Supplies Center
1050 East Remington Road
Schaumburg, Illinois 60195
Telephone:(312)640–5612

**Western Region**

Digital Equipment Corporation
Accessories and Supplies Center
632 Caribbean Drive
Sunnyvale, California 94086
Telephone:(408)734–4915

**First printing, December 1982**

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The following are trademarks of Digital Equipment Corporation:

digital™

DEC
DECmate
DECsystem–10
DECSYSTEM–20
DECUS
DECwriter
DIBOL
MASSBUS
PDP
P/OS
Professional
Rainbow
RSTS
RSX
UNIBUS
VAX
VMS
VT
Work Processor

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.

# CONTENTS

CONTENTS (Cont.)

CONTENTS (Cont.)

FIGURES

TABLES

# PREFACE

This manual, DECnet-20 User's Guide, includes information about using and programming DECnet-20. This manual should be used by:

- The application programmer responsible for writing the programs that will be exchanging data with programs on other systems in the network. This person should be an experienced MACRO programmer with some knowledge of network applications.

- The terminal user, using the network utilities that do not require privileges. Such a user, like all timesharing users, should be familiar with the TOPS-20 Command Language.

This manual is organized into three parts as follows:

Part I, the Introduction, consists of an overview of DECnet-20, the network options and network facilities.

Part II, the Programmer's Guide, describes the programming facilities that the MACRO programmer must use to perform the following network functions:

- Establishing a network connection using network-related monitor functions.

- Transmitting and receiving both data and interrupt messages over a network link using standard TOPS-20 I/O monitor calls.

- Controlling the network using network-related monitor functions.

- Terminating a network connection using network-related and standard monitor calls.

Part III, Terminal User's Guide, describes the TOPS-20 commands available to nonprivileged users. All network-related TOPS-20 commands, including user interaction with the file transfer utility (NFT), are explained.

This part also contains information relating to network remote terminal capability (SETHOST) and provides examples of the use of this facility.

The following are suggested guidelines for using this manual:

- Application programmers should read the entire manual.
- Terminal users should read Part III.
- System managers and system programmers should read the entire manual.

DECnet-20 version 3 runs under the TOPS-20 monitor on the DECSYSTEM-2040S/2060 models. DECnet-20 for the DECSYSTEM 2020 is DECnet-20 version 2 which is described in the TOPS-20 DECnet-20 Programmer's Guide and Operations Manual, order number AA-5091B-TM.

The following documents are either referenced in this manual or may prove useful in implementing DECnet-20 facilities.

| | |
|---|---|
| TOPS-20 Monitor Calls User's Guide | AA-D859B-TM |
| TOPS-20 Monitor Calls Reference Manual and its update | AA-4166E-TM |
| TOPS-20 Commands Reference Manual and its update | AA-5115B-TM |
| TOPS-20 User's Guide and its update | AA-4179C-TM |
| TOPS-10/TOPS-20 Batch Reference Manual | AA-H374A-TK |
| DECnet-20 System Manager's and Operator's Guide | AA-J678A-TM |
| DECnet DIGITAL Network Architecture (Phase III) General Description | AA-K179A-TK |
| TOPS-20 DN200 Remote Station Guide | AA-H786B-TM |

# PART I
# INTRODUCTION

CHAPTER 1

# SYSTEM OVERVIEW

## 1.1 DECNET

DECnet is the name given to the set of software products that extend the capabilities of various DIGITAL operating systems so that these systems can be interconnected to form computer networks.

DECnet is based on sets of rules (protocols) known collectively as DIGITAL Network Architecture, or DNA. These protocols govern the transmission of data over physical lines, using error detection and retransmission to guarantee the integrity of the data, multiplexing of multiple logical messages over a single physical connection, and controlling which nodes are allowed to communicate and when they can do so.

Each operating system that supports DECnet implements some subset of the complete DNA. The subset of DNA that runs under TOPS-20 is called DECnet-20.

## 1.2 DECNET-20 OPTIONS

The capabilities of DECnet-20 include multiline support, the Network File Transfer (NFT) utility, and the ability to log into a DECSYSTEM 2040S/2060 other than the one the terminal is connected to.

One DECnet option is available:

> RJE-20. This option includes software for the DN200 remote batch entry station and provides the facility for the DN200 software to be down-line loaded, diagnosed, and operated remotely by the host system.

A DECnet site may choose to divide the use of its lines between RJE stations and other DECnet hosts.

## 1.3 DECNET-20 STRUCTURE

When DECnet-20 is running on a DECSYSTEM-2040S/2060, the network software resides in the KL10 processor with the TOPS-20 monitor and in a separate processor (the DN20) designed to handle network communications functions (the KL10 processor and DN20 are, in fact, separate nodes). This latter processor is referred to as the communications front end to distinguish it from the console front end that controls the local command terminals and unit record peripherals. The KL10 processor communicates with either front end through a DTE hardware interface. (See Figure 1-1.)

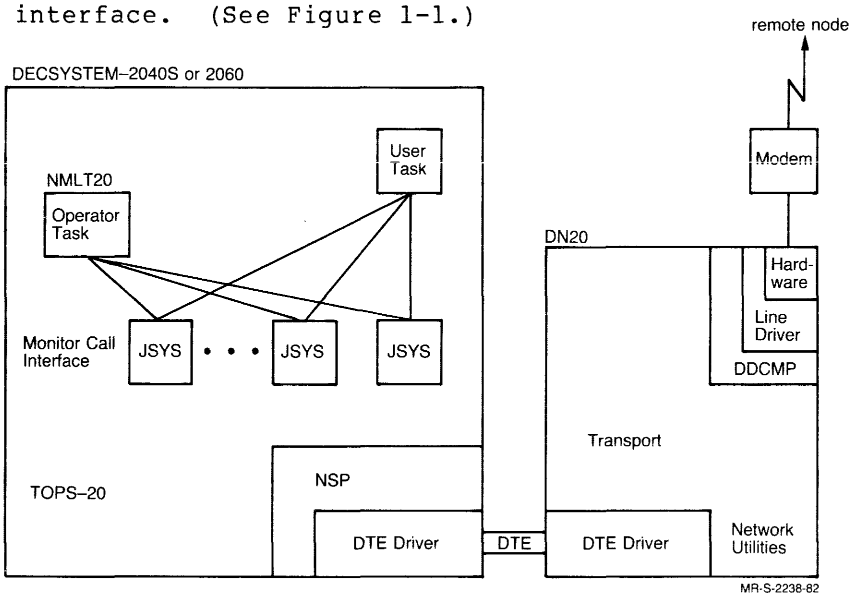


Figure 1-1 DECnet-20 on a DECSYSTEM-2040S/2060

## 1.4 DECNET-20 CAPABILITIES

DECnet-20 provides the TOPS-20 user with basic network task-to-task capabilities. That is, local system or user tasks written in MACRO-20 can exchange information with system or user tasks running in one or more nodes in a network.

The local task uses the TOPS-20 file system monitor calls to open, read and write information, and close files using a pseudo-device representing the network. These functions create a logical link, transmit data, and close the logical link.

Below this user level, and transparent to the user, the network protocols take over. Network software running in the KL10 processor and in the communications front end manages the actual transfer of data over a logical link. User data is first reformatted into network-compatible segments and then transferred. The network software also generates the appropriate control messages to open and close network connections.

## 1.5 DECNET-20 PROGRAMMER INTERFACE

A MACRO-20 program can transfer data between user storage and the network in much the same way that data is transferred between user storage and files on a peripheral device. The peripheral device, in this case, is the network; the file is a logical link. File system monitor calls, such as GTJFN, OPENF, and CLOSF, control the making and breaking of network connections. Input/output monitor calls, such as SIN, SINR, BIN, SIBE, SOUT, SOUTR, and BOUT, handle the movement of messages and data across the network. Several network-specific functions have been added to the MTOPR monitor call to provide for logical link management. These functions are described in subsequent chapters.

## 1.6 DECNET-20 TERMINAL USER'S INTERFACE

As a terminal user (without OPERATOR or WHEEL privileges), you use the TOPS-20 Command Language (EXEC) to communicate with the system. The *TOPS-20 User's Guide* and the *TOPS-20 Commands Reference Manual* describe the TOPS-20 Command Language in detail.

Part III of this manual describes the network file transfer utility (NFT) and EXEC commands that are specific to DECnet-20. NFT allows you to gain access to the files on other nodes through the Data Access Protocol (DAP) of the DNA.

# CHAPTER 2

# CONCEPTS AND FACILITIES

## 2.1 SYSTEM CONCEPTS

The user interface to DECnet-20 is based on the concept that the network is to be treated as a TOPS-20 input/output device. TOPS-20 programs written to communicate with other tasks in the network use TOPS-20 file system monitor calls to perform network functions. This concept is represented graphically in Figure 2-1. The local network software provides the user with access to the network using many of the same monitor calls that are used to access local peripherals. Figure 2-1 shows this code residing in the TOPS-20 monitor of a DECSYSTEM-2040S/2060. The communications front end is transparent to both local and remote user tasks.

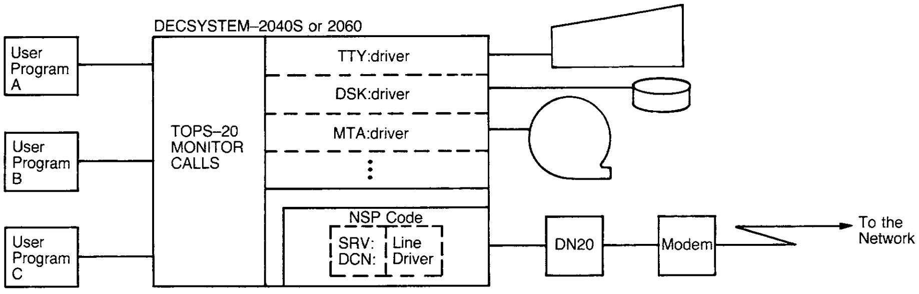


Figure 2-1 The Network as an I/O Device

In order to establish a network connection, you need one task that is willing to accept a connection (a target task) and another task that initiates the request for a connection (a source task). In the following discussion, it is easier to imagine these two tasks as existing on different nodes; however, there is no restriction that prohibits tasks on the same node from engaging in a network dialogue with one another.

To declare itself as being available for a connection by a source task, a target task identifies itself using a file specification with device type SRV:. When the SRV: device is opened, the target task has declared itself; it then waits until a connect request occurs. Any incoming connect request addressed to this target task is forwarded to it by DECnet-20. The request may be accepted or rejected. If the request is accepted, the connection is made and data can be exchanged between the two tasks via reads and writes to the file (JFN).

To initiate a dialogue with a target task, a source task identifies the target task as a file specification with device DCN:. The sour'ce task then opens the network file (note that both the source and target tasks must agree to the connection). The source and target tasks then exchange data and either the source or target task can close the link. Other network concepts that will be used in subsequent chapters are physical and logical links, network job file numbers, network task identification, and network node identification. These concepts are introduced in the following subsections.

### 2.1.1 Physical and Logical Links

Physical and logical links are the basic elements of communication on a network. Physical links connect network nodes and logical links connect network tasks.

A physical link connects two adjacent network nodes and can take one of several forms. In Figure 2-2, the physical link between nodes ABLE and ABLER or BAKERY and BAKER is the DTE interface, a hardware device between a DECSYSTEM-2040S/2060 and its communications front end. The physical link between nodes ABLER and BAKERY or between nodes BAKER and CHARLY can be a relatively permanent connection such as a leased or private telephone line or cable. It can also be a temporary connection such as a satellite link or radio circuit.

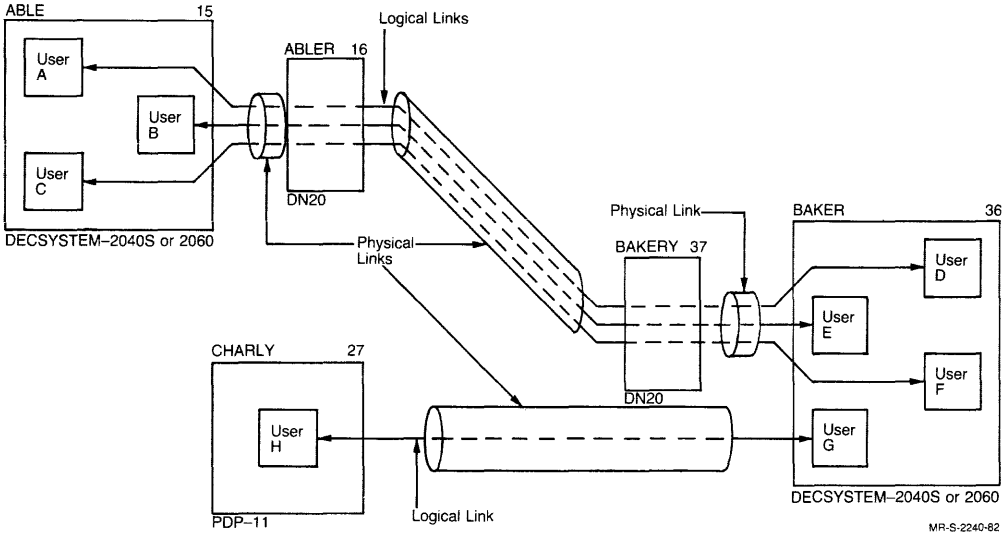


Figure 2-2 Logical and Physical Links

A physical link can support one or more logical links.

A logical link connects two network tasks that have both agreed to communicate. A logical link usually shares a physical link with other logical links. A logical link can span more than one physical link as shown from node ABLE to node BAKER in Figure 2-2.

The simultaneous sharing of a physical link by multiple logical links is referred to as multiplexing. Among the functions of the network software in DECnet-20 are the mixing of outgoing logical link data from several users (multiplexing) for transmission over a physical link and the separating of incoming logical link data (demultiplexing) for distribution to individual users. These functions are transparent to the user.

In Figure 2-2, the logical links between users A and D, B and E, and C and F, are multiplexed and span three physical links: ABLE to ABLER, ABLER to BAKERY, and BAKERY to BAKER. The logical link between users G and H is the only data path on the physical link from BAKER to CHARLY and therefore no multiplexing is necessary.

### 2.1.2 Network Job File Number

The network job file number in DECnet-20 is the same as the job file number (JFN) assigned to any other TOPS-20 file specification being processed by TOPS-20 processes. In the TOPS-20 file system, a JFN is associated with a file specification and constitutes a handle on the file. In DECnet-20, where the network is treated as an input/output device, a network JFN is associated with a specification for a logical link, the network equivalent for a file.

The same monitor call (GTJFN) is used to obtain either a network or a file JFN. The information passed to the GTJFN monitor call for a network JFN is similar in format to that supplied for a file JFN. The format and usage of logical link specifications are explained in Chapter 3, Establishing a Network Connection.

### 2.1.3 Network Task Identification

A network task is any program that is engaged in, or intends to engage in, a network dialogue. Network tasks in DECnet-20 can have two distinct identities: a generic task identification and a unique task name.

The generic task identification is used to address a network task that provides a class of service to other network tasks (for example, a network utility program). Multiple copies of such a network program can be loaded, started, and identified by class of service. Other network tasks can then request this service by specifying a connection by generic identification. This guarantees a connection to one of the available copies all of which are assumed to provide the same service. The generic task identification consists of two parts: a one-byte object type (numeric) and an optional object descriptor (alphanumeric).

Object types 1 through 127 are reserved for DECnet utilities and control programs. Object types 128 through 255 are available for customer use. Object type 0 is reserved for addressing tasks by their unique task name. See Appendix B for a list of the current DECnet object types.

The use of object descriptors is dependent upon the implementation of the network software on the remote node. If the remote node is running a system other than TOPS-20, read the DECnet manuals for the system being used. In DECnet-20, you can use object descriptors with object types 128 through 255.

A unique task name is used to address a specific network task. Only one copy of such a task can be running at any one time on any one node. If a network task is identified by task name alone, it must be addressed by the special object type 0 and a descriptor that corresponds to the unique task name.

### 2.1.4 Network Node Identification

A node name must be one to six alphanumeric characters in length and one of these must be alphabetic. At each node, the system manager assigns names by which the users reference the nodes in the network. When DECnet-20 is installed, the KL10 processor and the DN20 communications front end are each considered separate network nodes. Each, therefore, must have a unique node number.

Whenever a source task requests a connection to an existing target task, the source task must give the name of the target node. The network software generates a message to the target node requesting a connection to the target task. Sending this request is the first step in establishing a logical link.

For more information on network node names and numbers, see the DECnet-20 System Manager's and Operator's Guide.

## 2.2 NETWORK FACILITIES

You, as a TOPS-20 user, can write MACRO programs to communicate with tasks in another node. When doing so, you use a subset of the TOPS-20 file system monitor calls to interface to DECnet-20. These network-related monitor calls allow you to:

- Declare a network task as willing to accept connections.
- Initiate a request for a connection to another network task.
- Accept or reject a request for a connection from another network task.
- Transmit data to and/or receive data from another network task.
- Interrogate the status of a logical link.
- Retrieve the connect attributes of a network task.
- Exchange high priority interrupt messages (up to 16 bytes in length) with other network tasks.
- Disconnect a network connection.

The network-related monitor calls and their functions are listed in Tables 2-1, 2-2, 2-3, and 2-4. Many of these calls are also used in TOPS-20 file processing and their calling sequences are described in the TOPS-20 Monitor Calls Reference Manual. Information for all the network-related calls and the calling sequences for the network functions of MTOPR appear in the next three chapters.

Table 2-1
Monitor Calls Used in DECnet-20

| Monitor Call | Network Function |
|---|---|
| GTJFN | Get a network JFN |
| OPENF | Open a network connection |
| BIN | Receive a data byte |
| *BOOT | Provide maintenance and utility functions for communications software (see Table 2-2) |
| SIN | Receive a data string |
| SINR | Receive a data record (message) |
| BOUT | Transmit a data byte |
| SOUT | Transmit a data string |
| SOUTR | Transmit a data record (message) |
| SIBE | Test for input buffer empty |
| CLOSF | Close a network connection |
| MTOPR | Perform device-dependent control functions (see Table 2-3) |
| *NODE | Set node and line characteristics (see Table 2-4) |
| *NTMAN% | Network Management interface to lower DNA levels |

* BOOT, NTMAN%, and some functions of NODE are privileged monitor calls used in DECnet-20 system programs. Detailed descriptions of these monitor calls can be found in the TOPS-20 Monitor Calls Reference Manual.

Table 2-2
BOOT Monitor Call Functions Used in DECnet-20

| Symbol | Function |
|---|---|
| .BTROM | Puts line or DTE in MOP mode and activates the front-end ROM |
| .BTLDS | Load secondary bootstrap |
| .BTLOD | Loads the DN20 or console front end |
| .BTDMP | Dump the front end |
| .BTIPR | Initiate line protocol |
| .BTTPR | Terminate line protocol |
| .BTSTS | Determine line protocol |
| .BTBEL | Wait for front-end doorbell |
| .BTRMP | Read MOP message |
| .BTCLI | Convert line-id to port number |
| .BTCPN | Convert port number to line-id |

Table 2-3
MTOPR Monitor Call Functions Used in DECnet-20

| Symbol | Function |
|---|---|
| .MOACN | Set interrupt assignments |
| .MORLS | Read link status |
| .MORHN | Read host name |
| .MORTN | Read task name |
| .MORUS | Read user identification |
| .MORPW | Read password |
| .MORAC | Read account string |
| .MORDA | Read optional data |
| .MORCN | Read object type |
| .MORIM | Read interrupt message |
| .MOSIM | Send interrupt message |
| .MOROD | Read object-descriptor |
| .MOCLZ | Reject/Close a network connection |
| .MOCC | Accept a network connection |
| .MORSS | Read segment size |
| .MOANT | Attach network terminal |
| .MOSNH | Set network host |

Table 2-4
NODE Monitor Call Functions Used in DECnet-20

| Symbol | Function |
|---|---|
| .NDSLN | Set local node name |
| .NDGLN | Get local node name |
| .NDSNM | Set local node number |
| .NDGNM | Get local node number |
| .NDSLP | Set loopback port |
| .NDCLP | Clear loopback port |
| .NDFLP | Find loopback port |
| .NDSNT | Set network topology information |
| .NDGNT | Get network topology information |
| .NDSIC | Set topology change interrupt channel |
| .NDCIC | Clear topology change interrupt channel |
| .NDGVR | Get NSP version information |
| .NDGLI | Get line information |
| .NDVFY | Verify node name |

# PART II
# PROGRAMMER'S GUIDE

# CHAPTER 3

# ESTABLISHING A NETWORK CONNECTION

To establish a network connection, you need one task that is willing to accept a connection (a target task) and another task to initiate the request for a connection (a source task). In the following discussion, it is easier to imagine these two tasks as existing on different nodes; however, there is no restriction that prohibits tasks on the same node from engaging in a network dialogue with one another.

A TOPS-20 task that wants to declare itself as a target task, available for network dialogue with other network tasks, must first obtain a Job File Number (JFN) identifying itself on device SRV:. The target task must then open the SRV: in order to have a logical link assigned to it. Whenever a connect initiate message arrives, the target task can interrogate the connect attributes of the source task and decide whether to accept or reject the connection.

A TOPS-20 task that wants to initiate a network dialogue with a declared target task must first obtain a JFN for a network connection identifying the target task on device DCN:. It must then open the DCN: to have a logical link assigned and to have a connect initiate message sent to the target task.

SRV: and DCN: are special network devices that provide logical link service to another task. The JFN constitutes a handle on the task.

A TOPS-20 task can declare itself as a target task and also act as a source task by initiating a dialogue with some other task; the two actions are not mutually exclusive.

Figure 3-1 is a general overview of the dialogue that takes place when a network connection is established.

A task at node DALLAS issues a GTJFN monitor call identifying itself as a target task named TEX. A subsequent OPENF monitor call informs the network software at node DALLAS that TEX is ready to receive connection requests from the network.

A task at node BOSTON issues a GTJFN monitor call identifying itself as a task named TONY and specifying a network connection to task TEX at node DALLAS. A subsequent OPENF monitor call causes the network software at node BOSTON to send a connect initiate message to node DALLAS.

The network software at node DALLAS knows that task TEX is accepting calls and forwards the connect initiate message. Task TEX decides whether to accept or reject the connection and returns a connect confirm or connect reject message to the source task TONY.

The following sections of this chapter describe the individual steps that are required to establish a network connection.

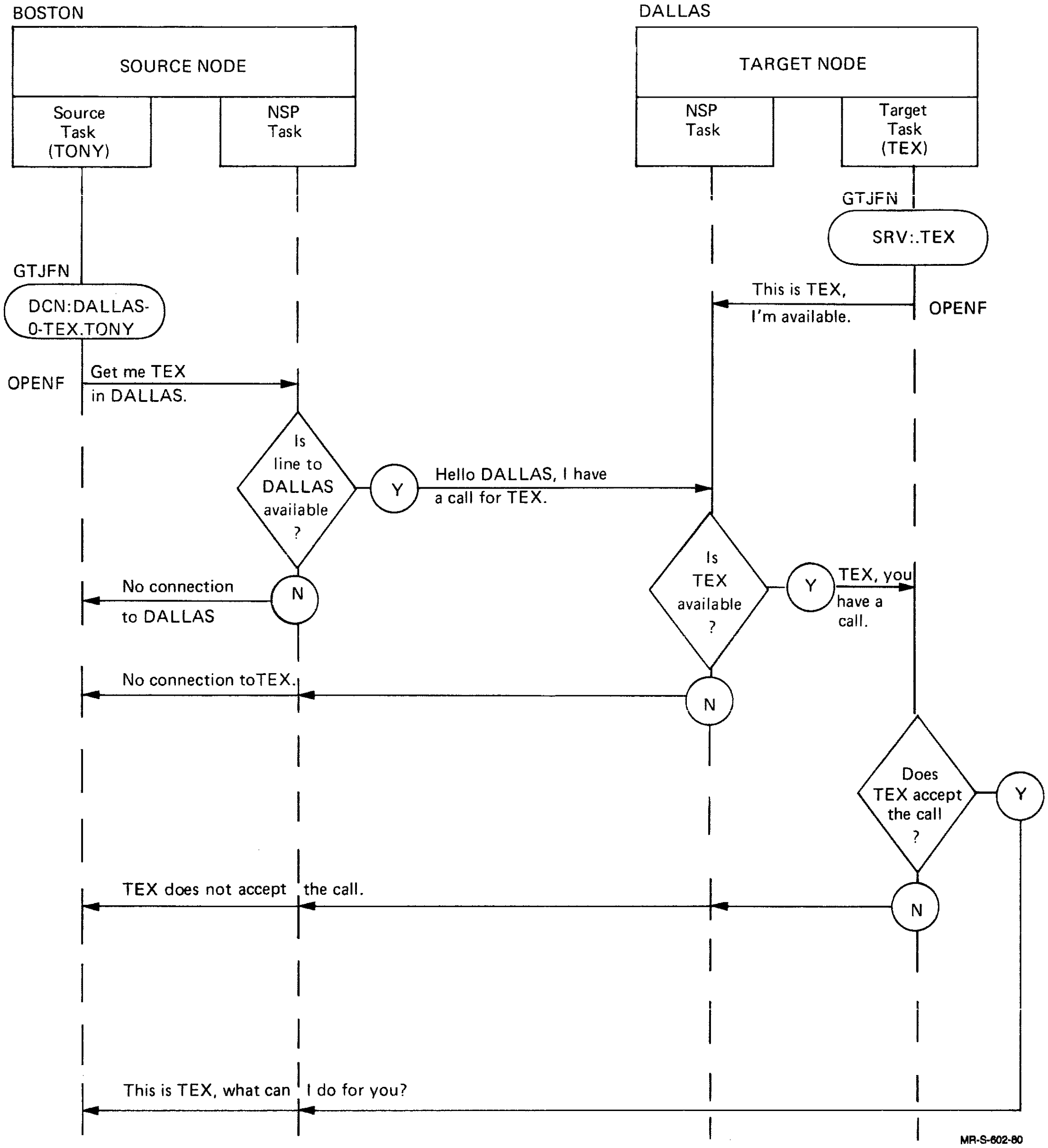


Figure 3-1 Establishing a Network Connection

## 3.1 OBTAINING A NETWORK JFN

The first step in establishing a network logical link is to obtain a Job File Number (JFN) for either the SRV: or DCN: device. Use the GTJFN monitor call in either its short or long form as described in detail in the Monitor Calls Reference Manual. The network file specification can be submitted interactively from your terminal, accessed from memory, or (in the long form) developed by a combination of both methods. Note that the connection must be opened with the OPENF call. The general format of a GTJFN file specification in TOPS-20 is:

```
dev:<directory>filename.filetype.generation;file attributes
```

When you use the GTJFN call to obtain a network JFN, the network file specification takes the following form:

dev: is replaced by one of the network pseudo-devices, SRV: or DCN:.

<directory> is unused.

filename is replaced by objectid-descriptor for an SRV: file and by hostname-objectid-descriptor for a DCN: file.

file type is replaced by a task name uniquely identifying the task issuing the GTJFN.

generation is unused.

file attributes are replaced by network attributes.

The individual fields in the network file specification are described in detail in the following subsections.

### 3.1.1 Specifying a Target Task

Use the following format of the network file specification to obtain a JFN identifying yourself as a target task:

```
SRV:objectid-descriptor.taskname
```

where:

SRV: is the logical device name for a target task.

objectid is part of an optional generic identification for a target task. If included, it must be a nonzero object type expressed as a decimal number or an object name (see Appendix B). The numbers 1 through 127 are reserved for DECnet system tasks and require enabled WHEEL or OPERATOR privileges. Numbers 1 through 127 should not be assigned to user tasks unless the task provides the service and uses the protocol implied by the object type (see Appendix B). Numbers 128 through 255 are available to all tasks. If objectid is not specified, the target task must be addressed by its unique task name.

| | |
|---|---|
| - (hyphen) | is a subfield separator that is required only if the descriptor is specified. |
| descriptor | is an optional modifier to be associated with the objectid. If specified, it must be 1 to 16 characters in length and contain only alphanumerics, hyphens, dollar signs, or underscores. If objectid is not specified, the descriptor must also be omitted. If descriptor is specified, it must also appear in the specification used by the source task to address this task (see Section 3.1.2). |

NOTE

Some DECnet implementations do not allow a descriptor to be associated with a nonzero object type. When communicating with a non-DECnet-20 node, read the applicable documentation to determine any restrictions on the generic task identification.

| | |
|---|---|
| . (period) | is a separator character and is required only if task name is specified. |
| taskname | is the unique task name by which a task is to be addressed independent of its generic identification. If taskname is specified, it must be 1 to 16 characters in length and contain only alphanumerics, hyphens, dollar signs, or underscores. If taskname is not specified, the monitor will assign one. (To subsequently determine the monitor-assigned task name, use the read task name function described in Section 3.4.3.) |

The maximum lengths of the variable fields in the SRV: file specification as imposed by TOPS-20 are:

| | |
|---|---|
| objectid | see object type and name in Appendix B |
| descriptor | 16 characters |
| taskname | 16 characters |

The above maximums may be reduced by any size limitations imposed by a DECnet product running under a different operating system on a remote node.

### 3.1.2 Specifying a Network Connection

Use the following format of the network file specification to obtain a JFN identifying a target task that you wish to connect to:

```
DCN:hostname-objectid-descriptor.taskname;A1;A2...
```

where:

DCN: is the logical device name for a network connection.

hostname is the node name of the node on which the target task is running. If this field is omitted, the target task is assumed to be running on the local node.

- (hyphen) is a subfield separator and is required.

objectid is the identification of the target task. It is an object name or a numeric object type when addressing a target task by its generic identification (see Appendix B). The special object type 0 (or corresponding object name TASK) is used to address a target task by its unique task name. The objectid, when specified as a numeric object type, must be entered in decimal. This subfield is required.

- (hyphen) is a subfield separator that is required only if the descriptor is specified.

descriptor is an optional modifier to be associated with the objectid. If objectid is TASK or 0, this field must be the unique task name of the target task. If objectid identifies some other object type, this field must be the descriptor specified by the target task.

NOTE

> Some DECnet implementations do not allow a descriptor to be associated with a nonzero object type. When you wish to communicate with a non-DECnet-20 node, read the applicable documentation to determine any restrictions on the generic task identification.

. (period) is a separator character and is required only if taskname is specified.

taskname is the unique taskname of the source task initiating the network connection. If taskname is specified, it must be 1 to 16 characters in length and contain only alphanumerics, hyphens, dollar signs, or underscores. If taskname is not specified, the monitor will assign one. (To subsequently determine the monitor-assigned task name, use the read task name function described in Section 3.4.3.)

;A1;A2... are a collection of attributes of the source task that are included in the connect initiate message sent to the target task. These attributes can be used by the target task to validate a network connection or to perform any other handshaking functions agreed to by both tasks. The allowable attributes are:

;USERID:userid where userid consists of 1 to 39 contiguous alphanumeric ASCII characters (including the hyphen, dollar sign, and underscore) identifying the source task.

Example:

```
;USERID:ALIBABA
```

NOTE

Special characters in a file specification must be ^v'ed. This allows the acceptance of PPNs for the USERID.

;PASSWORD:password where password consists of 1 to 39 contiguous alphanumeric ASCII characters (including the hyphen, dollar sign, and underscore) required by the target task to validate the connection.

Example:

```
;PASSWORD:SESAME
```

The password can also be specified in binary to allow non-ASCII characters. The keyword for this type of entry is BPASSWORD.

;BPASSWORD:password where password, in this context, consists of 1 to 8 octal triplets representing the required password. Each triplet represents an 8-bit byte.

Example:

```
BPASSWORD:123056002
```

;CHARGE:acctno where acctno consists of 1 to 39 contiguous alphanumeric ASCII characters (including the hyphen, dollar sign, and underscore) representing the source task's account identification.

Example:

```
;CHARGE:ACCT-13C
```

;DATA:userdata — where userdata consists of 1 to 16 contiguous alphanumeric ASCII characters (including the hyphen, dollar sign, and underscore) representing user data.

Example:

```
;DATA:THIS-IS-A-TEST
```

The user data can also be specified in binary to allow non-ASCII characters. The keyword for this type of entry is BDATA.

;BDATA:userdata — where userdata, in this context, consists of 1 to 13 octal triplets representing user data. Each triplet represents an 8-bit byte.

Example:

```
;BDATA:231337001
```

The attributes of a source task can be retrieved by a target task via functions of the MTOPR monitor call (see Section 3.4).

The maximum lengths of the variable fields in the DCN: file specification as imposed by TOPS-20 are:

| | |
|---|---|
| hostname | 6 characters |
| objectid | see object type and name in Appendix B |
| descriptor | 16 characters |
| hostname-objectid-descriptor | 39 characters including the hyphens |
| taskname | 16 characters |
| ;A1;A2... | see the description of the individual attribute |

The above maximums may be reduced by any size limitations imposed by the DECnet product running on the remote host system.

### 3.1.3 Examples of Network File Specifications

The following examples show various ways that a target task can declare itself and the corresponding ways that a source task must use to address the target task. These examples assume two DECnet-20 nodes with host node names of BOSTON and DALLAS.

**Example 1**

A task at node BOSTON wants to declare itself as the unique target task SAM. It does so with the specification:

```
SRV:.SAM
```

In order to request a connection to SAM at node BOSTON, a task TEX at node DALLAS would specify:

```
DCN:BOSTON-TASK-SAM.TEX
```

A task COD at node BOSTON requesting a connection to SAM at node BOSTON can omit the node name in the specification because the target node is the local node. It need only specify:

```
DCN:-TASK-SAM.COD
```

**Example 2**

A task at node BOSTON wants to declare itself as a generic service task, object type 128. It does so with the specification:

```
SRV:128
```

Task TEX at node DALLAS can connect to the above task with the specification:

```
DCN:BOSTON-128.TEX
```

Assume that the BOSTON task had included a descriptor in its specification such as:

```
SRV:128-PART1
```

The DALLAS task would then have to modify its specification to:

```
DCN:BOSTON-128-PART1.TEX
```

**Example 3**

Several tasks at node BOSTON, running the same utility program, want to declare themselves both generically as object type 129 and uniquely by task name. The respective specifications used to declare three such tasks are:

```
SRV:129.TOM
SRV:129.DON
SRV:129.TONY
```

A task TEX at node DALLAS, wanting to use the utility but not caring which copy of the program completes the connection, can specify:

```
DCN:BOSTON-129.TEX
```

If, for some reason, TEX had to connect to the particular task TONY, the specification must be submitted as:

```
DCN:BOSTON-TASK-TONY.TEX
```

Example 4

A task at node BOSTON declares itself as the target task XDATA with the specification:

```
SRV:.XDATA
```

Assume that the task XDATA restricts connections to those remote tasks that have a valid userid, password, and charge account. A specification from task TEX at node DALLAS to connect to XDATA would then have to include the above attributes, for example:

```
DCN:BOSTON-TASK-XDATA.TEX;USERID:RITTER;PASSWORD:SESAME
 ;CHARGE:ACCT-XYZ
```

The target task can then confirm the connect requirements by retrieving the network attributes using the read logical link data functions of the MTOPR monitor call. These functions are described in Section 3.4.

## 3.2 OPENING A NETWORK JFN

Having obtained a JFN for a network file specification, the network task must then open the file with the OPENF monitor call. The events that occur when a network file is opened depend on whether the file represents a target task or a source task.

### 3.2.1 Opening a Target Task JFN

An OPENF monitor call for a JFN that represents a target task implies that the task is ready to accept connect initiate messages from other tasks in the network. The network software performs the following functions:

- Constructs a link data base for this connection.
- Places the target task on a list of available connections.

Subsequently, when a connect initiate message is received from a source task, the network software:

- Searches the list of available connections for a matching generic or unique task identification.
- Notifies the appropriate target task via a connect interrupt that it has a connect request pending and modifies the link status appropriately.

The target task can then access the logical link data (see Section 3.4) to determine whether to accept or reject the connection.

### 3.2.2 Opening a Source Task JFN

An OPENF monitor call for a JFN that represents a source task implies a request for a connection to a target task. The network software:

- Constructs a link data base for this connection.
- Generates a connect initiate message and forwards it to the host node specified by the host name in the DCN: file specification.
- Processes the resulting connect confirm or connect reject message, and notifies the source task of the acceptance or rejection by a connect interrupt.

NOTE

> The successful completion of the OPENF monitor call for a network connection does not ensure that a network connection has been completed. To ensure that the remote node has accepted the connect request, read the link status with the .MORLS function of the MTOPR JSYS before transferring data over the link.

### 3.2.3 Limit on Open Links

DECnet-20 software sets a user quota of four open links per job. These can be any combination of SRV: and DCN: types. However, a task running with enabled WHEEL or OPERATOR privileges is not bound by this quota and may open as many links as the system will allow.

The system quota of open links varies according to the amount of monitor free space available at the time. Free space, in turn, is dependent upon the current demands of other processes. Whenever a request to open a link cannot be completed because of insufficient free space, an appropriate error code is returned.

## 3.3 USING NETWORK INTERRUPTS

Whenever a MACRO task uses a SIN or SINR monitor call to input a data string from the network, the task will stop running (block) if no data is available. In situations where a network task is supporting multiple links, blocking for each SIN or SINR call severely impacts the speed of data transmission. Asynchronously notifying the task of the arrival of network data reduces idle time and increases the overall throughput.

## ESTABLISHING A NETWORK CONNECTION

DECnet-20 has an interface to the MTOPR monitor call to allow a network task to enable software interrupt channels for any combination of the following types of network events:

- Connect event pending (connect initiate, connect confirm)
- Interrupt message available
- Data message or disconnect received

The MTOPR calling sequence to enable for network interrupts places the following arguments in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MOACN (function code)

AC3: Control information specifying the changes in the interrupt assignments for this link. This control information is placed in three 9-bit fields that are defined as follows:

| Field | Symbol | Used to signal |
|---|---|---|
| B0-B8 | MO%CDN | Connect event pending |
| B9-B17 | MO%INA | Interrupt message available |
| B18-B26 | MO%DAV | Data available |

The content of each of these fields must be one of the following:

| Value | Meaning |
|---|---|
| nnn | Enable the channel specified by nnn |
| .MOCIA | Clear the interrupt |
| .MONCI | Do not change the previous setting |

Valid user-assignable channels are defined in the Monitor Calls Manual.

### 3.3.1 Example

The following program segment illustrates one method of enabling interrupt channels for the three types of network events:

```
;  SET UP THE INTERRUPT CHANNELS AND GO INTO WAIT STATE

ENACHN: MOVEI   T1,.FHSLF                       ;IDENTIFY CURRENT PROCESS
        MOVE    T2,[LEVTAB,,CHNTAB]             ;SPECIFY TABLE ADDRESSES
        SIR                                     ;DEFINE PSI SYSTEM TABLES
        MOVX    T2,7B2                          ;SET BITS FOR CHAN 0,1,2
        AIC                                     ;ACTIVATE CHANNELS 0,1,2
        EIR                                     ;ENABLE THE SYSTEM
        MOVE    T1,NETJFN                       ;GET NETWORK JFN
        MOVEI   T2,.MOACN                       ;SET UP FUNCTION
        MOVX    T3,<FLD(0,MO%CDN)+FLD(1,MO%DAV)+FLD(2,MO%INA)>
        MTOPR                                   ;ISSUE THE CALL
PAUSE:  WAIT                                    ;WAIT FOR INTERRUPT
        .
        .
        .
LEVTAB: PC                                      ;LEVEL 1 PC ADDRESS
        0
        0
CHNTAB: 1,,HELLO                                ;CONNECT INTERRUPT
        1,,HAVDAT                               ;DATA AVAILABLE INTERRUPT
        1,,INTRPT                               ;INTERRUPT MESSAGE INTERRUPT
        REPEAT  ^D33,<EXP 0>                    ;UNUSED INTERRUPT CHANNELS
PC:     BLOCK 1                                 ;LEVEL 1 PC
```

NOTE

MOVX and FLD are macros defined in MACSYM.

## 3.4 RETRIEVING INFORMATION FROM THE LINK DATA BASE

Associated with each open logical link is a link data base. This data base contains information such as link status, link control data, allowable segment sizes, and data governing the transmission and receipt of data and interrupt messages.

Whenever a target task is notified of a pending connect request from a source task, the target task's data base will contain the connect attributes submitted by the source task. These attributes, as well as other link data, can be retrieved by a target task using the MTOPR monitor call. These functions retrieve the source's host name, task name, user identification, password, user account number, and optional user data. Using this data, the target task can decide whether to accept or reject the connection.

### 3.4.1 Reading the Link Status

The read link status function of MTOPR returns a 36-bit word of information regarding the status of the logical link.

Arguments must be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORLS (function code)

The following information is returned in AC3:

AC3: Flag bits in the left half and a disconnect code in the right half

The flag bits are:

| Symbol | Bit | Meaning |
|---|---|---|
| MO%CON | B0 | Link is connected |
| MO%SRV | B1 | Link is a server |
| MO%WFC | B2 | Link is waiting for an incoming connect |
| MO%WCC | B3 | Link is waiting for a connect to complete |
| MO%EOM | B4 | Link has the end of, or entire, message to be read |
| MO%ABT | B5 | Link has been aborted |
| MO%SYN | B6 | Link has been disconnected normally |
| MO%INT | B7 | Link has an interrupt message available |
| MO%LWC | B8 | Link has been previously connected |

The various disconnect codes are listed in Appendix A. If a disconnect code does not apply to the current status of the link, the value of the right half of AC3 will be zero.

**Example**

Assume that a source task obtains a JFN for a connection to a target task and opens the JFN. A successful return from the OPENF call does not necessarily mean that a connect confirm message from the target node or task has been received. To ensure that a connection really exists, you can use a coding sequence such as the following, which checks the link status every five seconds for one minute to determine whether the link was established:

```
        MOVEI   T2,.MORLS               ;SET UP FUNCTION
        SETZ    T4,                     ;INITIALIZE COUNTER
  CHK1: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
        TXNE    T3,MO%CON               ;NOW CONNECTED?
        JRST    CNCTED                  ;YES
        TXNE    T3,MO%WCC               ;NO, WAITING FOR CC?
        JRST    [CAIL T4,MXTRY          ;MXTRY=12
                JRST ABORT              ;GO RELEASE THE NETWORK JFN
                MOVEI T1, *D5000        ;YES WAIT 5
                DISMS                   ;SECONDS
                AOJA T4, CHK1]          ;CHECK LINK STATUS AGAIN
        TXNN    T3,MO%LWC               ;NO, EVER CONNECTED
        JRST    REJECT                  ;NO, CI REJECTED
        JRST    ABORT                   ;GO RELEASE THE NETWORK JFN
CNCTED:                                 ;OK, CONTINUE
```

### 3.4.2 Reading the Host Name

The read host name function of MTOPR returns the ASCII name of the node at the other end of the logical link.

The following arguments must be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORHN (function code)

AC3: A byte pointer to the location where the node name is to be stored

The monitor call returns an updated pointer in AC3 and the node name stored as specified.

**Example**

A target task may wish to give special connection privileges to tasks running on a particular remote node. The following program segment will retrieve the name of the node submitting the current connect request and store it at location NODNAM:

```
GTHNAM: MOVE    T1,NETJFN                  ;GET NETWORK JFN
        MOVEI   T2,.MORHN                  ;SET UP FUNCTION
        HRROI   T3,NODNAM                  ;POINTER TO NODE NAME
        MTOPR                              ;ISSUE THE CALL
         ERJMP  JSYSXX                     ;JSYS ERROR
         .
         .
         .
NODNAM: BLOCK   2                          ;REMOTE NODE NAME
```

### 3.4.3 Reading the Task Name

The read task name function of MTOPR returns the unique task name that is associated with your end of the logical link. If you had defaulted the task name in the network file specification, the call returns the monitor-supplied task name. In DECnet-20, the default or monitor-supplied task name is actually a unique number.

The following arguments must be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORTN (function code)

AC3: A byte pointer to the location where the task name is to be stored

The monitor call returns an updated pointer in AC3 and the task name stored as specified.

**Example**

Target tasks, especially those that perform utility functions, often default their task names because they do not expect to be addressed other than generically. If a connected source task wishes to initiate a second connection to your particular target task, the connected task requires your unique task name. You can first retrieve the monitor assigned name using a program segment as follows:

```
GTDTNM: MOVE    T1,NETJFN                 ;GET NETWORK JFN
        MOVEI   T2,.MORTN                 ;SET UP FUNCTION
        HRROI   T3,TSKNAM                 ;POINTER TO TASK NAME
        MTOPR                             ;ISSUE THE CALL
         ERJMP  JSYSXX                    ;JSYS ERROR
         .
         .
         .
TSKNAM: BLOCK   4                         ;LOCAL TASK NAME
```

Once retrieved, the task name can be sent to the connected source task via a data transfer or interrupt message (see Sections 4.1.1 and 4.2.1).

### 3.4.4 Reading the User Name

The read user name function of MTOPR is valid only for target tasks. It returns the source task's ASCII user identification supplied in the connect initiate message.

The following arguments must be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORUS (function code)

AC3: A byte pointer to the location where the user identification is to be stored

The monitor call returns with an updated pointer in AC3 and the user identification stored as specified. If no user identification was supplied by the source task, AC3 continues to point to the beginning of the string and a null is returned as the only character.

**Example**

A target task may include code to reject connect initiate requests from all but a select group of userids. The following program segment retrieves the userid of the current connect request:

```
GTUSID: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MORUS               ;SET UP FUNCTION
        MOVE    T3,UIDPTR               ;POINTER TO USERID
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
        CAMN    T3,UIDPTR               ;CHECK IF ANY USERID
         JRST   REJECT                  ;NO - REJECT
         .
         .
         .
        check if userid is valid
         .
         .
         .
USERID: BLOCK   4                       ;SOURCE USERID
UIDPTR: POINT   7,USERID                ;USERID POINTER
```

### 3.4.5 Reading the Password

The read password function of MTOPR is valid only for target tasks. It returns the source task's password supplied in the connect initiate message.

The following arguments must be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORPW (function code)

AC3: A byte pointer to the location where the password is to be stored. Passwords may be binary; therefore, the byte pointer should accommodate 8-bit bytes unless you know that the password is ASCII.

The monitor call returns with an updated pointer in AC3 and the source task's password stored as specified. AC4 contains the number of bytes in the string; a zero value indicates that no password was supplied by the source task.

**Example**

In addition to screening a source task's userid, a target task may require the submission of a password before confirming a connect request. Whereas a userid is usually permanent, a password can be changed periodically to ensure security. The following program segment retrieves the password submitted by a source task in its connect request. The two tasks have agreed to use ASCII passwords:

```
GTPSWD: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MORPW               ;SET UP FUNCTION
        MOVE    T3,PWDPTR               ;POINTER TO PASSWORD
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
        JUMPE   T4,REJECT               ;REJECT IF NO PASSWORD
         .
         .
        check for valid password
         .
         .
         .
PWDPTR: POINT   7,PASSWD                ;PASSWORD POINTER
PASSWD: BLOCK   2                       ;SOURCE PASSWORD
```

### 3.4.6 Reading the Account String

The read account string function of MTOPR is valid only for target tasks. It returns the ASCII account string supplied by the source task in the connect initiate message.

The following arguments must be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORAC (function code)

AC3: A byte pointer to the location where the account string is to be stored

The monitor call returns with an updated pointer in AC3 and the source task's account string stored as specified. If no account string was supplied by the source task, AC3 continues to point to the beginning of the string and a null is returned as the only character.

**Example**

A target task that includes a cost distribution of its services may set up a chart of accounts and require each connecting task to supply an account identification. With this information the target task can control access, set budgets, check overruns, and provide billing data. The following program segment retrieves the account string supplied in a connect initiate message:

```
GTACCT: MOVE    T1,NETJFN                  ;GET NETWORK JFN
        MOVEI   T2,.MORAC                  ;SET UP FUNCTION
        MOVE    T3,ACCPTR                  ;POINTER TO ACCT NO.
        MTOPR                              ;ISSUE THE CALL
         ERJMP  JSYSXX                     ;JSYS ERROR
        CAMN    T3,ACCPTR                  ;CHECK IF ANY ACCT NO.
         JRST   REJECT                     ;NO - REJECT
         .
         .
        process the account number
         .
         .
         .
ACCTNO: BLOCK   4                          ;ACCOUNT STRING
ACCPTR: POINT   7,ACCTNO                   ;ACCT NO. POINTER
```

### 3.4.7 Reading the Optional Data

The read optional data function of MTOPR returns the optional data supplied in any of the connect or disconnect messages.

The following arguments must be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORDA (function code)

AC3: A byte pointer to the location where the optional user data is to be stored. This field. may be binary; therefore, the byte pointer should accommodate 8-bit bytes unless you know that the data is ASCII.

The monitor call returns with an updated pointer in AC3 and the optional data stored as specified. AC4 contains the number of bytes in the data string; a zero value indicates that no optional data was supplied.

#### Example

The user level protocol, agreed to by two corresponding tasks, may state that optional user data will always accompany a connect reject message. The following program segment will retrieve optional user data in binary:

```
GTDATA: MOVE    T1,NETJFN                ;GET NETWORK JFN
        MOVEI   T2,.MORDA                ;SET UP FUNCTION
        MOVE    T3,DATPTR                ;POINTER TO USER DATA
        MTOPR                            ;ISSUE THE CALL
         ERJMP  JSYSXX                   ;JSYS ERROR
        JUMPE   T4,NODATA                ;BRANCH IF NO DATA
          .
          .
          .
DATPTR: POINT   8,USRDAT                 ;USER DATA POINTER
USRDAT: BLOCK   4                        ;USER DATA
```

### 3.4.8 Reading the Object Type

The read object type function of MTOPR is valid only for target tasks. It returns the object type that was used by the source task to address this connection. The result indicates whether the local task was addressed by its generic object type or its unique network task name.

The following arguments must be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORCN (function code)

The monitor call returns with the object type in AC3. A zero object type indicates that the target task was addressed by its unique network task name; a nonzero value indicates that it was addressed by its generic object type.

**Example**

Assume for example that the services of a target task depend on whether a source task connects to it by generic object type or by unique task name. The following program segment retrieves the object type used in the connect initiate message:

```
GTOBJT: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MORCN               ;SET UP FUNCTION
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
        JUMPE   T3,TSKCON               ;TEST TYPE OF CONNECT
OBJCON:  .                              ;CONNECTED BY OBJECT TYPE
         .
         .
TSKCON:  .                              ;CONNECTED BY TASK NAME
         .
         .
```

### 3.4.9 Reading the Object-Descriptor

The read object-descriptor function of MTOPR is valid only for target tasks. It returns the unique identification of the source task. This identification is in the format of object-descriptor and the contents depend on the DECnet implementation on the remote host. In addition, if the source task is running on a system that provides for group and user codes, this information is also returned. If the source task is on a DECnet-20 host, the data returned to the target task is TASK-taskname.

The following arguments must be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MOROD (function code)

AC3: A byte pointer to the location where the object-descriptor of the source task is to be stored

The monitor call returns with an updated pointer in AC3 and the object-descriptor stored as specified. In addition, if the source host system uses group and user codes, AC4 contains the following:

AC4: The group code in the left half and the user code in the right half

If the source host system does not provide for group or user codes, or if none was provided in the connect initiate message, AC4 contains zeros.

**Example**

A target task can retrieve the unique identification of the source task with the following program segment:

```
GTOBJD: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MOROD               ;SET UP FUNCTION
        HRROI   T3,OBJDES               ;POINTER TO OBJ. DESC.
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JSYSXX                  ;JSYS ERROR
        JUMPN   T4,[HRRZM T4,USRCOD     ;SAVE USER CODE
                HLRZM T4,GRPCOD         ;SAVE GROUP CODE
                 JRST .+1]              ;RETURN
          .
          .
          .
OBJDES: BLOCK   5                       ;OBJECT-DESCRIPTOR
GRPCOD: BLOCK   1                       ;GROUP CODE
USRCOD: BLOCK   1                       ;USER CODE
```

### 3.4.10 Reading the Segment Size

The read segment size function of MTOPR returns the maximum segment size that can be used over this link. The local task can use this value to determine the optimum size of data records being transmitted over the link.

The following arguments must be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: .MORSS (function code)

The maximum segment size, in bytes, is returned in AC3. If the link has not been established, the monitor call takes the error return.

#### Example

A task can retrieve this value with a program segment such as the following:

```
GTSGSZ: MOVE    T1,NETJFN               ;GET NETWORK JFN
        MOVEI   T2,.MORSS               ;SET UP FUNCTION
        MTOPR                           ;ISSUE THE CALL
         ERJMP  JYSXX                   ;JSYS ERROR
        MOVEM   T3,SEGSIZ               ;STORE SEGMENT SIZE
          .
          .
          .
SEGSIZ: BLOCK   1                       ;MAX SEGMENT SIZE
```

## 3.5 ACCEPTING OR REJECTING A CONNECTION

When the target task has decided to accept or reject the connection, it must inform the network software with a monitor call or by transmitting data. Two MTOPR monitor call functions are provided: .MOCC to accept a connection and return data, and .MOCLZ to reject the connection and return data along with a specific reject reason. If no data or specific reject reason is to be returned, the target task can accept or reject the connection implicitly, without using either of the two MTOPR functions. This is explained in the following subsections.

### 3.5.1 Accepting the Connection

Connections can be accepted either explicitly or implicitly.

You can accept a connection explicitly by sending a connect confirm message to the source task with the .MOCC function of MTOPR. This method allows you to include up to 16 bytes of optional data in the connect confirm message.

The following arguments must be placed in the specified accumulators:

| | |
|---|---|
| AC1: | The JFN of the logical link |
| AC2: | .MOCC (function code) |
| AC3: | A byte pointer to any data to be returned |
| AC4: | The count of bytes in the data string. A zero indicates no data. The maximum amount of data is 16 bytes. |

You can accept a connection implicitly by performing one of the following:

- Issuing an output monitor call such as BOUT, SOUT, or SOUTR to the network JFN
- Issuing an input monitor call such as BIN, SIN, or SINR to the network JFN
- Placing yourself in an input or output wait state

Performing one of these operations does not allow you to send any optional data.

### 3.5.2 Rejecting the Connection

Connections can be rejected either explicitly or implicitly.

You can reject a connection explicitly by sending a connect reject message to the source task with the .MOCLZ function of MTOPR. This method allows you to include a reject code as well as up to 16 bytes of optional data in the connect reject message.

The following arguments must be placed in the specified accumulators:

AC1: The JFN of the logical link

AC2: A reject code in the left half and .MOCLZ in the right half

AC3: A byte pointer to any data to be returned

AC4: The count of bytes in the data string. A zero indicates no data. The maximum amount of data is 16 bytes.

The reject code in AC2 is a 2-byte, NSP-defined decimal number indicating the reason that a target task is rejecting a connection. A list of these codes, applicable to both user and system tasks, appears in Appendix A.

You can reject a connection implicitly by closing the JFN of the logical link before accepting the connection either explicitly or implicitly. To close the JFN, use the CLOSF monitor call. When the CLOSF call is used, the source task will see a reject code of 38 (user aborted). You must reopen the JFN to receive subsequent connect initiate messages.

### 3.5.3 Examples

The following program segment will send a connect confirm message to the source task that requested the connection:

```
ACCEPT: MOVE    T1,NETJFN                  ;GET THE NETWORK JFN
        MOVEI   T2,.MOCC                   ;CODE TO ACCEPT
        SETZ    T4,                        ;FLAG NO OPT. DATA
        MTOPR                              ;ISSUE THE CALL
         ERJMP  JSYSXX                     ;JSYS ERROR
```

To include up to 16 bytes of ASCII user data with the connect confirm message, modify the above segment as follows:

```
ACCEPT: MOVE    T1,NETJFN                  ;GET THE NETWORK JFN
        MOVEI   T2,.MOCC                   ;CODE TO ACCEPT
        MOVE    T3,[POINT 7,MSGC]          ;POINTER TO USER DATA
        MOVEI   T4,^D16                    ;USER DATA BYTE COUNT
        MTOPR                              ;ISSUE THE CALL
         ERJMP  JSYSXX                     ;JSYS ERROR
          .
          .
          .
MSGC:   ASCIZ   /OPEN UNTIL 10 PM/
```

The following program segments will send a connect reject message to the source task that requested the connection. The reason for the rejection is coded as 34 (access not permitted). With no user data, the instruction sequence is:

```
REJECT: MOVE    T1,NETJFN                 ;GET THE NETWORK JFN
        MOVEI   T2,.MOCLZ                 ;CODE TO REJECT
        HRLI    T2,.DCX34                 ;ADD REJECT CODE 34
        SETZ    T4,                       ;FLAG NO USER DATA
        MTOPR                             ;ISSUE THE CALL
         ERJMP  JSYSXX                    ;JSYS ERROR
```

Because reject code 34 is somewhat general, you may want to include ASCII user data to clarify the rejection. You can modify the above sequence as follows:

```
REJECT: MOVE    T1,NETJFN                 ;GET THE NETWORK JFN
        MOVEI   T2,.MOCLZ                 ;CODE TO REJECT
        HRLI    T2,.DCX34                 ;ADD REJECT CODE 34
        MOVEI   T3,[POINT 7,XPWD]         ;POINTER TO REJECT MSG
        MOVE    T4,^D14                   ;REJECT MSG BYTE COUNT
        MTOPR                             ;ISSUE THE CALL
         ERJMP  JSYSXX                    ;JSYS ERROR
          .
          .
          .
XPWD:   ASCIZ   /WRONG PASSWORD/
```

For longer program segments that include sequences similar to those in this chapter, see Figures 5-1 and 5-2.

# CHAPTER 4

# TRANSFERRING INFORMATION OVER THE NETWORK

Once a network connection has been established, the task at either end of the logical link can send information to the task at the other end. DECnet-20 provides for two types of information exchange:

- Data transfers
- Interrupt messages

Data transfers are primarily used by network tasks to move large blocks of data. Interrupt messages are used by network tasks to exchange small amounts (16 bytes or less) of high priority data that are not sequentially related to the main data flow.

Data transfers and interrupt messages are discussed in the remainder of this chapter.

## 4.1 TRANSFERRING DATA

Data transfers over a logical link involve the segmentation and restructuring of data at both the logical and physical link levels. The network software accepts data from the user program, segments it to conform to the maximum segment size allowable on that logical link, precedes each segment with a header, and passes these segments to the physical link management layer. This layer segments the data to conform to the maximum segment size allowable on the physical link and precedes each segment with a header to form a packet. These packets are then sent over the physical line to the destination node. At the destination node the reverse procedure takes place: headers are stripped and segments re-assembled.

Data transfers on a logical link can take one of two forms: logical messages or continuous byte streams. The logical message format provides for the transmission of information in discrete logical units called records, or messages. Data transmitted in this format can be retrieved by the receiving task on a message-by-message basis.

The continuous byte stream format does not have any end-of-message indicators. Data transmitted in this format is presented to the receiving task as a continuous stream of data. The receiving task must reconstruct the original messages via some prearranged protocol agreed to by both user tasks.

The logical message format allows for simpler user retrieval routines at the expense of not taking full advantage of the monitor's buffering capabilities. The continuous byte stream format permits more efficient use of resources but requires the user to write routines to reconstruct the original messages.

### 4.1.1 Sending Data

You can send data to another task with the SOUT, BOUT, or SOUTR monitor call. In general, use SOUTR to send data in the logical message format and SOUT and BOUT to send data as a continuous byte stream.

The exclusive use of SOUTR usually implies that both tasks have agreed to exchange information in the form of messages with some stated maximum length. Each use of the SOUTR monitor call results in the transmission of a logical message terminated with an end-of-message indicator. A DECnet-20 receiving task can then retrieve the message with the SINR (read record) monitor call (see Section 4.1.2).

The exclusive use of SOUT or BOUT usually implies that both tasks have agreed to exchange information in the form of a continuous byte stream. You can repeatedly fill your data buffer and execute the SOUT monitor call. Each SOUT transmits a buffer's worth of data to the destination node where it is presented to the receiving task. A DECnet-20 target task can then use the SIN and BIN (read data) monitor calls (see Section 4.1.2) to retrieve as many bytes at a time as it can handle in its data buffer. Normally, you would include a count byte at the beginning of each logical data group to provide the receiving task with a means to reconstruct the logical data.

You can also intermix the use of SOUT and SOUTR when you send data to a receiving task. For example, assume that the program you write has a data buffer limited to 300 bytes. The receiving task has a 1000 byte buffer and requires that data be sent to it in logical message format. You can send an 800 byte logical message by sending two SOUTs for 300 bytes each followed by a SOUTR for 200 bytes. The receiving task can then retrieve the entire message by using a SINR for 800 bytes or more.

### 4.1.2 Receiving Data

You can retrieve data from the network using the SINR, SIN, or BIN monitor calls. In general, use SINR to retrieve logical messages and SIN to retrieve data from a continuous byte stream.

The exclusive use of SINR usually implies that tasks have agreed to exchange information in the form of messages with some stated maximum length. Each SINR monitor call results in the retrieval of one logical message. The sending task must have sent the message using the SOUTR monitor call (see Section 4.1.1).

The exclusive use of SIN usually implies that tasks have agreed to exchange information in the form of a continuous byte stream. You can retrieve as many bytes at a time as your data buffer can handle. If your data buffer is 300 bytes in length, you can execute a SIN for 300 bytes and fill the buffer with data. The manner in which you then reconstruct the original logical messages depends on the user-level protocol agreed to by both tasks. For example, if each message is preceded by a count byte, you can use the BIN monitor call to read the first byte and obtain the number of bytes in the message and then issue a SIN for that number of bytes to retrieve the message. A subsequent BIN would read the number of bytes in the next message.

You can intermix the use of SIN and SINR when you wish to retrieve data from the network. However, a few precautions are in order where multiple messages are concerned. A SIN monitor call does not recognize an end-of-message (EOM) indicator if one is present. Therefore, if you issue a SIN for 300 bytes and the link buffer contained the last 200 bytes of a logical message (sent with a SOUTR), you will retrieve those 200 bytes plus the first 100 bytes of the next message. The SIBE monitor call is useful here because it will return the number of bytes of the current message that are available in the link buffer (merely checking for an EOM is not sufficient). You can then issue a SIN for that number of bytes and not encroach on the succeeding message.

As the coding sample below illustrates, it is important to test the condition of the input buffer (using the SIBE JSYS) before attempting to read data. An interrupt can occur on the data channel when data is available, but can also occur because of a disconnect event. An interrupt on a data channel indicates one of four states.

1. Data available, disconnect event (disconnect initiate)

2. Data available, no disconnect event ("normal" case)

3. No data, disconnect event (disconnect initiate)

4. No data, no disconnect event (spurious interrupt)

All four states must be handled in the user code, with the aid of the SIBE JSYS and .MORLS function of the MTOPR JSYS.

```
PAUSE:  WAIT                    ;WAIT FOR NETWORK INTERRUPT
        .
        .
        .
;
; READ INCOMING DATA
;
; THIS ROUTINE IS EXECUTED WHEN AN INTERRUPT OCCURS
; ON THE PSI CHANNEL INDICATING THAT DATA MAY BE
; AVAILABLE
;
; NOTE THAT THE SIBE MONITOR CALL IS USED TO CHECK THAT
; DATA ACTUALLY EXISTS FOR THE USER TASK.  THIS IS A
; RECOMMENDED PROCEDURE TO FOLLOW TO GUARANTEE THAT
; DATA IS AVAILABLE FOR THE PROCESS.
```

```
;
GOTSOM: MOVE T1,NETJFN          ;GET NETWORK JFN
        SIBE                    ;INPUT BUFFER EMPTY?
         JRST READ              ;NO, GO TO READ ROUTINE
        MOVEI T2,.MORLS         ;YES, CHECK LINK STATUS
        MTOPR                   ;ISSUE THE CALL
         ERJMP NOGOOD           ;UNLIKELY ERROR HERE
        TXNN T3,MO%CON          ;STILL CONNECTED?
         JRST NOCNCT            ;NO, CONNECTION IS GONE
        DEBRK                   ;YES, HAVE READ ALL AVAILABLE
;                               ;DATA. WAIT FOR MORE.
; THIS ROUTINE IS EXECUTED WHEN THE INPUT BUFFER FOR
; THE LOGICAL LINK CONTAINS DATA. THE NUMBER OF BYTES
; IN THE INPUT BUFFER WAS RETURNED IN AC2 BY THE SIBE
; MONITOR CALL.
;
READ:   MOVNI T3,0(T2)          ;GET NEG OF COUNT
        MOVE T4,T3              ;SAVE COUNT FOR ECHO
        MOVE T2,INPTR           ;WHERE TO PUT DATA
        SIN                     ;GET THE DATA
;
; NOW SEND IT BACK DOWN THE NETWORK LINE
;
        MOVE T2,INPTR           ;POINT TO DATA JUST READ
        MOVE T3,T4              ;RECOVERY BYTE COUNT
        SOUTR                   ;WRITE A MESSAGE TO LINK
        JRST GOTSOM             ;PROCESS MORE INPUT
        .
        .
```

Should the SIN or SOUT fail, you should read the link status before closing the link.

### 4.1.3 Summary of Procedures - Source and Target Tasks

The following summarizes procedures for target and source tasks:

**Target Task**

STEP 1: Issue a GTJFN on a SRV: file specification.

STEP 2: Issue an OPENF to become available on the network.

STEP 3: Set up the software interrupt system with a SIR, AIC, EIR, and MTOPR (function .MOACN), in that order. You must execute the MTOPR JSYS last, or interrupts may be lost.

STEP 4: Issue the WAIT JSYS to wait for an interrupt.

STEP 5: Process interrupts on the connect channel.

STEP 6: Process interrupts on the data channel when they occur. Check with SIBE to see if input is waiting and read all of the input.

Always do another SIBE before the DEBRK since data arriving while you are processing an interrupt will not generate another interrupt.

If no input is waiting, check to see if the link is still connected (.MORLS function of MTOPR, check for MO%CON). If the link is connected, you have received a spurious interrupt and should:

Go to STEP 6

If the link is not connected, you have received a DISCONNECT INITIATE

Go to STEP 7

STEP 7: The link is disconnected. Issue a CLOSF on the JFN, and either halt the program or return to STEP 1.

**Source Task**

STEP 1: Issue a GTJFN on a DCN: file specification.

STEP 2: Issue an OPENF to establish the link. The OPENF sends a connect initiate; success of the OPENF does not mean the link is established.

STEP 3: Use the .MORLS function of MTOPR to read link status and then wait for connect confirm or reject.

STEP 4: Transfer data with SOUT or SOUTR until done.

STEP 5: Use CLOSF to break the link, setting CZ%ABT and performing clean-up routines if necessary.

4.1.3.1 **Special SINR/SOUTR Considerations** - If you have enabled a channel for data interrupts, a SINR or SOUTR will block forever if it is interrupted and a disconnect initiate has arrived. You cannot simply dismiss the interrupt with DEBRK, because the SINR or SOUTR will hang rather than fail.

You must force the SINR or SOUTR to complete by (1) setting the user mode bit (bit 5) of the return PC word, and/or (2) modifying the PC word's return address. You can then safely dismiss the interrupt.

A user program can determine whether it was interrupted out of user code or monitor code by examining bit 5 of the PC word. Bit 5 "on" indicates user code; bit 5 "off" indicates monitor code. (The address contained in the PC word is always in user code, however.)

If you modify the PC word, you should also set a flag to indicate that an abnormal branch has occurred. It may also be useful to check T3 for the count of bytes remaining after the SOUTR.

## 4.2 TRANSFERRING INTERRUPT MESSAGES

Although data transfers over a logical link are guaranteed to be received at the other end in the same order in which they were sent, it is occasionally necessary to bypass the normal flow of data and to send data that is to be delivered immediately. Events such as errors or status changes in one task or the other are situations that justify bypassing the normal data flow.

The logical link management level, NSP, allows for the transmission and reception of short high-priority messages called interrupt messages. An interrupt message is sent and accounted for independently of any buffered data messages and its delivery is guaranteed to be prompt. Interrupt messages are limited to 16 bytes in length and therefore are not very useful for exchanging data. They are most effectively used as event indicators and usually require the subsequent exchange of data by the two processes owning the logical link. In this respect, they closely resemble software interrupts. Consequently, DECnet-20 provides the network task with a monitor call function to enable an interrupt channel for the receipt of an interrupt message. (See Section 3.3.)

### 4.2.1 Sending Interrupt Messages

A network task communicating over a logical link can initiate an interrupt message at any time. Whether DECnet-20 will send the message over the link depends on conditions at the other end. If the task at the other end has not acknowledged a previous interrupt message, the sending task is notified by an error message. If the other task is not enabled for interrupt messages, the link will not accept the message.

To send an interrupt message, use the .MOSIM function of the MTOPR monitor call.

The calling sequence expects the following:

AC1: The JFN of the logical link

AC2: .MOSIM (function code)

AC3: A byte pointer to the message

AC4: The count of bytes in the interrupt message. The maximum message length is 16 bytes.

**Example**

The following program segment can be used to send an interrupt message to another task:

```
SNDMSG: MOVE    T1,NETJFN                    ;GET NETWORK JFN
        MOVEI   T2,.MOSIM                    ;SET UP FUNCTION
        HRROI   T3,MSG                       ;POINTER TO MESSAGE
        MOVEI   T4,^D14                      ;BYTE COUNT
        MTOPR                                ;ISSUE THE CALL
         ERJMP  JSYSXX                       ;JSYS ERROR
         .
         .
         .
MSG:    ASCIZ   /CLOSING AT 6PM/
```

### 4.2.2 Receiving Interrupt Messages

If the protocol used by network tasks includes interrupt messages, each task should provide a way to be notified asynchronously when messages arrive. The .MOACN function of the MTOPR monitor call allows a network task to enable specific channels for software interrupts (see Section 3.3). One of these interrupts signals the arrival of an interrupt message.

When a remote task sends your task an interrupt message, the appropriate channel presents a software interrupt to your task. To read the interrupt message, use the .MORIM function of the MTOPR monitor call.

The calling sequence expects the following:

AC1: The JFN of the logical link

AC2: .MORIM (function code)

AC3: A byte pointer to the receiving buffer. The maximum message length is 16 bytes.

The call returns with an updated pointer in AC3, the message stored in the buffer, and the count of bytes received in AC4.

Because interrupt messages are used to signal important asynchronous events, it is recommended that a task receiving an interrupt message read the interrupt message promptly. Furthermore, DECnet-20 does not acknowledge an interrupt message until the task reads it. This means that each network task is limited to one outstanding interrupt message and until that message is read, no others will be accepted.

**Example**

The following program segment retrieves an interrupt message:

```
INTRPT: MOVE    T1,NETJFN                       ;GET NETWORK JFN
        MOVEI   T2,.MORIM                       ;SET UP FUNCTION
        MOVE    T3,[POINT 8,MSGBUF]             ;POINTER TO MESSAGE
        MTOPR                                   ;ISSUE THE CALL
         ERJMP  JSYSXX                          ;JSYS ERROR
         .
         .
         .
MSGBUF: BLOCK   4                                ;MESSAGE BUFFER
```

For longer program segments that include sequences similar to those in this chapter, see Figures 5-1 and 5-2.

# CHAPTER 5

# CLOSING A NETWORK CONNECTION

Either of the two connected tasks can close a network connection. A connection can be closed normally, thereby preserving the integrity of any data in transit; or, a connection can be aborted without regard to any undelivered data.

## 5.1 CLOSING A CONNECTION NORMALLY

A normal close is usually accomplished with the CLOSF monitor call specifying a network JFN in AC1. The CZ%ABT bit in AC1 must be off. All buffered data that is in transit at the time is delivered (unless the remote task executes an abort before the CLOSF has completed). The network JFN is then closed.

An MTOPR call with function code .MOCLZ also disconnects the logical link and completes the delivery of all buffered data; however, it does not close the JFN. This method of closing a link is only used if it is necessary to send user data (up to 16 bytes) to the remote task. In order to send user data and also close the JFN, the .MOCLZ function must be followed by a CLOSF call.

The calling sequence for the MTOPR call is:

AC1: The JFN of the logical link

AC2: 0 in the left half and .MOCLZ in the right half

AC3: A byte pointer to the user data. If the byte size is over 8, bytes are truncated to eight bits.

AC4: The count of bytes in the user data. The maximum is 16 bytes.

The network does not have explicit protocol for a normal close. That is, no one specific network control message is available to both disconnect a logical link and also automatically have all data correctly delivered. When you use the MTOPR or CLOSF monitor call to close a network connection, you are actually turning over control of the link to the local NSP. It is the job of the local NSP to ensure that all outstanding data packets have been properly acknowledged before sending the disconnect message to the remote NSP.

The remote NSP, in turn, notifies the remote task according to the protocol in effect at the remote node.

If the remote node is a DECnet-20 node, the NSP task receiving the disconnect message sets the MO%SYN bit in the remote task's link status to reflect that the link has been closed normally. If the remote task is not in the process of reading data, it is issued a data interrupt. If the remote task issues a SIBE call, it will be informed that no bytes are available. If the remote task attempts to read data, it will receive an end-of-file indication. In any case, reading the link status with the .MORLS function of MTOPR will indicate that the MO%SYN bit has been set.

### Example

To close a logical link after delivering all the data currently in transit, use a program segment such as the following:

```
CLOSE: MOVE     T1,NETJFN                        ;GET NETWORK JFN
       CLOSF                                     ;ISSUE THE CALL
        ERJMP   JSYSXX                           ;JSYS ERROR
         .
         .
         .
```

To close a logical link as above and also include ASCII user data for the target task, a program segment such as the following can be used:

```
CLOSED: MOVE     T1,NETJFN                       ;GET NETWORK JFN
        MOVEI    T2,.MOCLZ                       ;SET UP FUNCTION
        MOVE     T3,[POINT 7,MSG]                ;POINTER TO MESSAGE
        MOVEI    T4,^D14                         ;BYTE COUNT
        MTOPR                                    ;ISSUE THE CALL
         ERJMP   JSYSXX                          ;JSYS ERROR
        CLOSF                                    ;CLOSE THE JFN
         ERJMP   JSYSXX                          ;JSYS ERROR
          .
          .
          .
MSG:    ASCIZ    /BE BACK AT 6PM/                ;USER DATA
```

## 5.2 ABORTING A CONNECTION

You can abort a logical link with the CLOSF monitor call by specifying both the CZ%ABT bit and the network JFN in AC1. All buffered data in transit is discarded and the network JFN is closed. This operation can result in the loss of data and should only be used in a fatal error condition.

The .MOCLZ function of MTOPR, used to normally close a logical link in Section 5.1, can be used to abort a logical link if you insert a nonzero code in the left half of AC2. This method of aborting a link should only be used if it is necessary to send the remote task a specific reason code for the abort, up to 16 bytes of user data, or both. The .MOCLZ function with the abort option discards all buffered data in transit and closes the link; however, it does not close the network JFN. To close the JFN, the MTOPR call must be followed by a CLOSF call with the CZ%ABT bit set in AC1.

The calling sequence for the MTOPR call is:

AC1: The JFN of the logical link

AC2: A reason code, nn, in the left half and .MOCLZ in the right half

AC3: A byte pointer to the user data

AC4: The count of bytes in the user data. The maximum is 16 bytes

The reason code (nn) in the left half of AC2 is one. of the nonzero codes listed in Appendix A.

With either the CLOSF or MTOPR monitor call, the local NSP sends a disconnect message to the remote NSP which, in turn, notifies the remote task according to some established protocol.

If the remote node is a DECnet-20 node, the NSP task receiving the disconnect message sets the MO%ABT bit in the remote task's link status to reflect that the link has been aborted. If the remote task is not in the process of reading data, it is issued a data interrupt. Any attempts to read data will result in an I/O error. Reading the link status with the .MORLS function of MTOPR will indicate that the MO%ABT bit has been set and the right half of AC3 will contain a disconnect code if one was given.

### Example

To abort a logical link immediately without completing the delivery of any data in transit, use a program segment such as the following:

```
ABORT: MOVE     T1,NETJFN                     ;GET NETWORK JFN
       TLO      T1,(CZ%ABT)                   ;SET ABORT BIT
       CLOSF                                  ;ISSUE THE CALL
        ERJMP   JSYSXX                        ;JSYS ERROR
        .
        .
        .
```

To abort a logical link as above and also include a specific reason code and user data, use a program segment such as the following:

```
ABORTD: MOVE    T1,NETJFN                     ;GET NETWORK JFN
        MOVEI   T2,.MOCLZ                     ;SET UP FUNCTION
        HRLI    T2,.DCX9                      ;CODE FOR USER ABORT
        MOVE    T3,[POINT 7,MSGX]             ;POINTER TO MESSAGE
        MOVEI   T4,^D16                       ;BYTE COUNT
        MTOPR                                 ;ISSUE THE CALL
         ERJMP  JSYSXX                        ;JSYS ERROR
        TLO     T1,(CZ%ABT)                   ;SET ABORT BIT
        CLOSF                                 ;CLOSE THE JFN
         ERJMP  JSYSXX                        ;JSYS ERROR
         .
         .
         .
MSGX:   ASCIZ   /RESTART XMISSION/            ;USER DATA
```

If the target task has the MO%ABT bit set in the link status word, the target task must use CZ%ABT or the CLOSF will fail. An example of a program segment using CZ%ABT follows:

```
MOVE    T1,NETJFN
CLOSF
 ERJMP  [ MOVE  T1,NETJFN                        ;If fail, then
          TLO   T1,(CZ%ABT)                      ;use CZ%ABT
          CLOSF
            JRST JSYSXX
          JRST  .+1]
```

## 5.3 SOURCE AND TARGET TASK CODING EXAMPLES

Figures 5-1 and 5-2 are examples of coding for source and target programs.

```
; Get JFN for Network Connection
;
        MOVX    T1,GJ%SHT
        HRROI   T2,[ASCIZ/DCN:NODEA-TASK-TARGET.SOURCE/]
        GTJFN
         ERJMP  NOGOOD          ;Failed, Probably out of resources
        MOVEM   T1,OURJFN       ;Successful, save our JFN
;
;OPENF to create the Logical Link
;
        MOVX    T2,<FLD(^D7,OF%BSZ)+OF%WR+OF%RD> ;Open for read and
                                                 ;write
        OPENF
         ERJMP  NOGOOD          ;Failed
;
; Wait for network connect to succeed or fail
;
CHKST:  MOVX    T1,^D1000       ;Wait before checking status
        DISMS
        MOVE    T1,OURJFN       ;Check line status
        MOVX    T2,.MORLS
        MTOPR
         ERJMP  NOGOOD
        TXNE    T3,MO%CON       ;Connected?
         JRST   HELLO           ;Yes, proceed to HELLO
        TXNE    T3,MO%WCC       ;No, are we waiting still?
         JRST   CHKST           ;Yes, delay some more
; If we get here, target process or network rejected cannot attempt
        JRST    NOGOOD          ;We lose
;
; Send data to Target task
;
HELLO:  MOVE    T1,OURJFN
        HRROI   T2,[ASCIZ/Hello Target!
/]
        SETZM   T3
        SOUTR
         ERJMP  NOGOOD          ;Network went away
;
; CLOSF to disconnect logical link
;
        MOVE    T1,OURJFN
        CLOSF
         ERJMP  [ MOVE  T1,OURJFN       ;Failed, use CZ%ABT
                  TXO   T1,CZ%ABT       ; to close link
                  CLOSF
                   JFCL                 ;Don't Care if it fails
                  JRST  .+1]
        HALTF                           ;Stop source task
;
;Include what you wish to do on failure of logical link
;
NOGOOD  .
        .
OURJFN: BLOCK   1
```

Figure 5-1  Example of Source Task Coding

```
;
; Get JFN for Network Connection
;
START:  MOVX    T1,GJ%SHT
        HRROI   T2,[ASCIZ/SRV:.TARGET/]
        GTJFN
         ERJMP  NOGOOD          ;Failed, Probably out of resources
        MOVEM   T1,OURJFN       ;Successful, save our JFN
;
; Start setting up interrupt system for network JFN
;
        MOVX    T1,.FHSLF       ;Set up interrupt system first
        MOVE    T2,[LEVTAB,,CHNTAB]
        SIR                     ;Set interrupt system tables
        MOVX    T2,3B1          ;Enable channels 0 and 1
        AIC                     ;Activate interrupt channels
        EIR                     ;Enable for interrupts
;
; OPENF to make us available to the network
;
        MOVX    T2,<FLD(^D7,OF%BSZ)+OF%WR+OF%WR+OF%RD>  ;Open for read and
                                                        ;write
        OPENF
         ERJMP  NOGOOD          ;Failed
;
; Finish setting up interrupt system for network JFN
;
        MOVE    T1,OURJFN       ;Set up Connect and Data Interrupts
        MOVEI   T2,.MOACN
        MOVX    T3,<FLD(0,MO%CDN)+FLD(1,MO%DAV)+FLD(.MONCI,MO%INA)>
        MTOPR
;
PAUSE:  WAIT                    ;Wait for Interrupts
;
LEVTAB: PC                      ;Level 1 PC address
        0
        0
CHNTAB: 1,,HELLO                ;On connect go to HELLO
        1,,READIT               ;On data interrupt try to read it
        REPEAT  ^D34,<0>        ;Zero fill rest of table

PC:     BLOCK   1               ;Level 1 PC save location
;
; Process interrupt on Connect channel
;
HELLO:  MOVE    T1,OURJFN       ;Always accept the connection
        MOVX    T2,.MOCC
        SETZB   T3,T4           ;No optional data
        MTOPR
         ERJMP  NOGOOD          ;Something blew up
        DEBRK                   ;Done, wait some more
```

Figure 5-2  Example of Target Task Coding

```
;
; Process interrupt on Data channel
;
READIT: MOVE    T1,OURJFN       ;Any data?
        SIBE
         JRST   READ            ;Yes, Process it
        MOVX    T2,.MORLS       ;No, See if link still connected
        MTOPR
         ERJMP  NOGOOD          ;An error here is not likely, but...
        TXNN    T3,MO%CON       ;Link Still Connected?
         JRST   DISCON          ;No, process link down
        DEBRK                   ;Yes, then wait for another interrupt
;
READ:   HRROI   T2,BUFFER       ;Put data into buffer
        MOVNI   T3,^D1000       ;-Size of buffer
        SINR                    ;Read the data
         ERJMP  NOGOOD          ;Shouldn't happen
        SETZM   T1              ;Store a zero byte
        IDPB    T1,T2
        HRROI   T1,BUFFER
        PSOUT                   ;Output the message
        JRST    READIT          ;Process any more input
;
; Process disconnect on link
;
DISCON: MOVE    T1,OURJFN       ;Close our JFN
        CLOSF
         ERJMP  [ MOVE  T1,OURJFN       ;Try CLOSF with CZ%ABT
                  TXO   T1,CZ%ABT
                  CLOSF
                   JFCL         ;Don't care if it fails
                  JRST .+1]
        JRST    START           ;Start over
;
; Include what you wish to do on failure
;
NOGOOD: .
        .
        .
;
BUFFER: BLOCK   ^D1000          ;Buffer save location
OURJFN: BLOCK   1               ;OURJFN save location
;
```

Figure 5-2 (Cont.)  Example of Target Task Coding

# PART III
# TERMINAL USER'S GUIDE

# CHAPTER 6

# TOPS-20 DECnet-20 EXEC COMMANDS

As a nonprivileged terminal user, you communicate with the system by using the TOPS-20 Command Language (EXEC). This chapter assumes that you are familiar with the most frequently used TOPS-20 commands (both for timesharing and batch). TOPS-20 commands that relate directly to DECnet functions are described in this chapter.

NOTE

> If you have had little experience with the TOPS-20 Command Language, refer to the list of suggested documents in the Preface of this manual. As an absolute minimum, you should read the following manuals before continuing with this and the following chapter:
>
> <u>TOPS-20 User's Guide</u><br>
> <u>TOPS-20 Commands Reference Manual</u><br>
> <u>TOPS-10/TOPS-20 Batch Reference Manual</u>

You should know the name of your local node and the names of all remote nodes with which you will communicate. If remote nodes require a user name, password, or account, you will need to know the specific way in which this information must be formatted. Your installation should have the basic user's manuals for all systems accessible to you via DECnet. If you need help, see your system manager or operator.

The TOPS-20 operating system in conjunction with DECnet software allows you to do the following:

- List accessible DECnet nodes using the TOPS-20 INFORMATION DECNET command.

- List the destination for your queued output using the TOPS-20 INFORMATION JOB-STATUS command.

- Direct queued output to an accessible DN200 or IBM-type remote station using the TOPS-20 SET LOCATION command or the TOPS-20 /DESTINATION-NODE: node switch for queue-class commands.

To delete files from an accessible DECnet node and to transfer files to or from an accessible DECnet node, use the Network File Transfer (NFT) program. (The NFT program is described in Chapter 7, Network File Transfer.)

## 6.1 INFORMATION COMMAND

The INFORMATION command has two options that give DECnet information: INFORMATION DECNET and INFORMATION JOB-STATUS.

### 6.1.1 Information DECnet

INFORMATION DECNET lists the accessible DECnet nodes.

The format of the INFORMATION DECNET command is as follows:

```
@INFORMATION (ABOUT) DECNET NODES
```

Example:

```
 ESC
  ↓
@infORMATION (ABOUT) decnet RET
 Local DECNET node: KL2137
 Accessible DECNET nodes are:  D2102A  DN200  KL1031  KL2102
```

(Note that NODES is assumed as the default if omitted.)

### 6.1.2 Information Job-Status

INFORMATION JOB-STATUS lists the destination for your queued output if you have used the SET LOCATION command to specify a DN200 or IBM-type remote station as that destination.

The format of the INFORMATION JOB-STATUS command is as follows:

```
@INFORMATION (ABOUT) JOB-STATUS
```

Example:

```
 ESC
  ↓
@INFORMATION (ABOUT) JOB-STATUS RET
 Job 41, User SKOGLUND, MISC:<SKOGLUND>, Account 341, TTY225
 Located at DN200
@
```

## 6.2 SET LOCATION COMMAND

The SET LOCATION command instructs the TOPS-20 operating system to regard the specified DN200 or IBM-type remote station as the destination for your queued output. (When you log in, the destination for your queued output is your local host.) Note that print requests for a DN200 or IBM-type remote station must conform to the capabilities of that remote station.

The format of the SET LOCATION command is:

```
@SET LOCATION (TO) node::
```

where:

node:: The name of the DN200 or IBM-type remote station that becomes the destination for your queued output. If no node name is entered, the node name defaults to the name of your local host.

Example:

```
     @set location dn200::(RET)
     @print test.txt(RET)
     [Job TEST Queued, Request-ID 550, Limit 27]
     @information (ABOUT) output-requests
     Printer Queue:
     Job Name     Req    Limit              User
     --------     ----   -----   ------------------------
      TEST1        87      5     HORAN         On Unit:0/Dest:DN200
       Started at 16:24:43, printed 0 of 5 pages
```

The INFO OUTPUT command above illustrates the effect of the SET LOCATION command.

If you give the INFORMATION JOB-STATUS command immediately following the SET LOCATION command, you can check to be sure your logical location has changed before you continue:

```
               (ESC)
                 ↓
     @set locATION (TO) dn200::(RET)
     @i j(RET)
      Job 62, User CIRINO, Account 341, TTY106
     Located at DN200
```

Remember that the request remains in the queue until it is honored. If it appears that the request is being ignored, use the INFORMATION DECNET command to see if the DN200 is still available. If it is available, repeat the INFORMATION DECNET command later; if it is not available, check with your operations staff if the job is critical. (The DN200 may require manual loading.)

Printed on the DN200 printer are the contents of the file TEST.TXT:

```
     This is a test file!
```

You can also use the SET LOCATION command to direct requests to a DN200 or IBM-type remote station. If the operator is not at the terminal of the remote station, the response to your PLEASE request will be delayed. The example below shows the input and output at both the local site and the remote station:

Typed on the user's terminal at the local site is:

```
     @set location dn200::

       (ESC)                 (ESC)
         ↓                     ↓
     @infORMATION (ABOUT) joB-STATUS (RET)

      Job 33, User CIRINO, Account 341, TTY106
     Located at DN200
     @please turn printer on(RET)

     [PLSOPN Operator at DN200 has been notified at 15:21:09]
```

Output to the console at the remote site is:

```
     KL2102::OPR>
     KL2102::
     15:21:09     <8>    --Message from Timesharing User--
                        JOB 33 CIRINO at Terminal 106
                        PLEASE turn printer on
```

Input by the remote operator is:

```
     KL2102::OPR> RESPOND 8 PRINTER IS ON(RET)
     KL2102::OPR>
```

The answer received at the host site is:

```
     [Operator Response Received at 15:21:58]
     PRINTER IS ON
```

## 6.3 /DESTINATION-NODE SWITCH

The /DESTINATION-NODE switch is used with the PRINT command to direct output to the specified DN200 or IBM-type remote station. When this switch is used with the SUBMIT command, the log file is directed to the specified remote station.

The format for the /DESTINATION-NODE switch is as follows:

```
     /DESTINATION-NODE:node::
```

where:

node::     The name of the remote station to which output is directed.

Example:

```
     @ PRINT FOO.BAR/DESTINATION-NODE:DN200::(RET)
```

## 6.4 ADDITIONAL FEATURES AVAILABLE TO NONPRIVILEGED USERS

The Network File Transfer Program described in chapter 7 can be run by a nonprivileged user.

Chapter 8 describes the SETHOST program that uses DECnet-20's task-to-task communications capabilities. This program allows a privileged or nonprivileged user at a terminal to log in to a remote host on the same network as the user's local host.

All users may use the SPEAR program to type or print network error and event logging reports. See the SPEAR manual, order number AA-J833A-TK.

CHAPTER 7

# NETWORK FILE TRANSFER

## 7.1 OVERVIEW

The Network File Transfer utility allows you to access or delete files residing on DECnet hosts that provide network file access capabilities. NFT is a task-to-task utility consisting of an active task NFT (DCN:) and a server task FAL (SRV:). By using NFT, you can delete files from a remote host and transfer sequential files between your local host and a remote host; FAL checks for requests made by NFT. The NFT and FAL programs communicate using the Data Access Protocol (DAP).

All network file transfers must be direct requests between the local host and one remote host. Files can be transferred from your local host to a remote host or from a remote host to your local host.

The files deleted or transferred using NFT can be text, program, data, control, or any other sequential files. Some file formats cannot be transferred between TOPS-20 systems and non-TOPS-20 systems. See Section 7.2.3, which discusses the NFT COPY command, for more information on this subject.

NFT does not include network file spooling. Unless you are using the wildcard feature, you can make only one file transfer request at a time and that request must be for only one file to be transferred. (See Section 7.2.1.)

### 7.1.1 Specifying File Access Information

Each file deletion or transfer request must include a valid user identification, account, and password for the system to be accessed. The FAL at the remote host is responsible for verifying your access to the requested file and subsequently honoring or rejecting your request. The requirements of the remote node determine the values you specify in access information switches or in response to prompts for access information. This security measure is necessary to protect a node's files from unauthorized access or accidental deletion. You must enter either the particular access parameter required by the remote node, or a carriage return if the remote node does not require a parameter or has an established default value.

NFT prompts you for access information (user, account, password) when you type the first NFT command that requires such information. If the access is successful, all subsequent file requests to the node addressed will use the access information that you provided in response to the prompt. If you supply access information by using the SET DEFAULTS command as the first NFT command, or if you have set defaults for the node in an NFT.INIT file in your logged-in directory, you will not be prompted for that access information. Whether you supply access information in response to a prompt from NFT, by the SET DEFAULTS command typed to your terminal, or by SET DEFAULTS commands in an NFT.INIT file, the values will be remembered. The NFT.INIT file is read each time you run NFT.

Access information entered in response to a prompt or with a SET DEFAULTS command remains effective until changed with another SET DEFAULTS command. Access information switches are used to override default values already established. Switch values are effective only for the command in which entered.

## 7.2 NFT COMMANDS

You call the NFT program by typing NFT or R NFT in response to the TOPS-20 operating system prompt. After you type NFT and press RETURN, the NFT program prints the prompt NFT>. The list of valid NFT commands follows:

COPY
DELETE
DIRECTORY
EXIT
HELP
INFORMATION
PRINT
SET
SUBMIT
TAKE
TYPE

The file specifications for remote files must have the format required by the operating system at the remote host. The operating systems and corresponding formats include the following:

| Operating system | File Specifications Format |
|---|---|
| TOPS-20 | device:<directory>filnam.type.gen |
| VMS | device:[username]filnam.ext;gen |
| RT, RSTS | device:[UIC]filnam.ext |
| RSX, IAS | device:[UIC]filnam.ext;gen |

### 7.2.1 SET DEFAULTS Command

The SET DEFAULTS command establishes the default access information to be used with all subsequent NFT commands for the specified node. The values established with the SET DEFAULTS command remain in effect until you exit from NFT or change the values with another SET DEFAULTS command. The SET DEFAULTS command does not prompt for omitted information. However, NFT does prompt for required omitted access information switches at the time you type the first command that requires a switch not previously set with a SET DEFAULTS command.

Access information values may be changed for a specific command by including the desired access information switch. After the command containing the switch has been executed, the access information values revert to the previously established default values.

The format of the SET DEFAULTS command is as follows:

```
NFT>SET DEFAULTS (FOR NODE) node::/switches
```

where:

node:: is the node name to which the default values are assigned.

/switches are any combination of access information switches (see Table 7-1) and, in addition, the switch /OSTYPE:operating-system. Valid values for the operating-system argument are TOPS-10, TOPS-20, RSX, RSTS, VMS, or IAS.

Examples:

```
    @NFT(RET)
               (ESC)
                 ↓
     NFT>SET DEFAULTS (FOR NODE) ALPHA::/USER:JONES/OSTYPE:VMS(RET)

               (ESC)
                 ↓
     NFT>SET DEFAULTS (FOR NODE) DELTA::/USER:CLEMENS/PASSWORD:TOPS20(RET)

               (ESC)
                 ↓
     NFT>SET DEFAULTS (FOR NODE) GAMMA::/USER:REILLY/ACC:UETP(RET)
     NFT>
```

You can place SET DEFAULTS commands in an initialization file that will be read when you run NFT. The initialization file must be in your logged-in directory and must be called NFT.INIT. Access information established in the NFT.INIT file may be changed by typing a SET DEFAULTS command, or may be overridden by an access information switch. The SET DEFAULTS command you type at the terminal will be effective for the current NFT session unless you change or override it. The access information switch is effective only for the command in which it is specified.

Following is an example of an NFT.INIT file currently in use. The INFORMATION DECNET command included at the end provides a convenient way to check the currently available nodes whenever you run NFT.

```
SET DEFAULTS  KL2102::/USER:PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
SET DEFAULTS  KL1031::/USER:PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
SET DEFAULTS  SYS880::/USER:GUEST/ACCOUNT:FOO/OSTYPE:RSX11/PASS:DUMB
SET DEFAULTS  KL4097::/USER:REILLY.PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
SET DEFAULTS  KL2137::/USER:REILLY/ACC:UETP
INFORMATION DECNET
```

You may clear the default access information for a node by typing the SET DEFAULTS command without specifying any access information. To clear the default access information for a node other than the node where your logged-in directory is located, the node name must be included. If you omit node name, the system assumes the node to be the node where your logged-in directory is located.

Two examples are shown below. In Example 1 there is no NFT.INIT file in the logged-in directory. In Example 2 there is an NFT.INIT file with default access information for four nodes as shown. The NFT.INIT file also includes the INFORMATION DECNET command. Including this command in the NFT.INIT file saves you from having to type the command at the beginning of each NFT session. Note in the first example that the default access information is available for the node where your logged-in directory is located even though you have not given a SET DEFAULTS command. NFT establishes these parameters each time the NFT or R NFT command is given.

When you exit from NFT, all default access information as cleared or set in the NFT run from which you have exited is lost. If you run NFT again and type the INFORMATION DEFAULTS command before any SET DEFAULTS commands are given the response will always be either the default access information for the node where your logged-in directory is located (no NFT.INIT file) or defaults given by the SET DEFAULTS commands in the NFT.INIT file.

Example 1.

```
@NFT(RET)

    (ESC)               (ESC)
      ↓                   ↓
NFT>inFORMATION (ABOUT) deFAULTS(RET)
Node KL2102::/USER:CIRINO/ACCOUNT:341/OSTYPE:TOPS20

        (ESC)
          ↓
NFT>set deFAULTS (FOR NODE) (RET)

    (ESC)               (ESC)
      ↓                   ↓
NFT>inFORMATION (ABOUT) deFAULTS(RET)
Node KL2102::/OSTYPE:TOPS20
NFT>
```

Example 2.

```
     @NFT RET
      Accessible DECNET nodes are:  DN20A   DN200   KL1031  KL2102  KS4097

          ESC                ESC
           ↓                  ↓
     NFT>inFORMATION (ABOUT) deFAULTS RET
     Node KL2102::/USER:PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
     Node KL1031::/USER:P:TAYLOR/ACCOUNT:341/OSTYPE:TOPS20
     Node KL4097::/USER:REILLY.PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
     Node SYS880::/USER:GUEST/ACCOUNT:FOO/OSTYPE:RSX

              ESC
               ↓
     NFT>set deFAULTS (FOR NODE) RET

          ESC                ESC
           ↓                  ↓
     NFT>inFORMATION (ABOUT) deFAULTS RET
     Node KL2102::/OSTYPE:TOPS20
     Node KL1031::/USER:PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
     Node KL4097:/USER:REILLY.PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
     Node SYS880::/USER:GUEST/ACCOUNT:FOO/OSTYPE:RSX

              ESC
               ↓
     NFT>set deFAULTS (FOR NODE) KL1031:: RET

          ESC                ESC
           ↓                  ↓
     NFT>inFORMATION (ABOUT) deFAULTS RET
     Node KL2102::/OSTYPE:TOPS20
     Node KL1031::/OSTYPE:TOPS20
     Node KL4097::/USER:REILLY.PTAYLOR/ACCOUNT:341/OSTYPE:TOPS20
     Node SYS880::/USER:GUEST/ACCOUNT:FOO/OSTYPE:RSX
     NFT>
```

### 7.2.2 INFORMATION Command

The INFORMATION command has two options: DEFAULTS and DECNET.

The INFORMATION DEFAULTS command displays the current settings of the default switches for a specified node.

The format of the INFORMATION command is as follows:

```
     NFT>INFORMATION (ABOUT) DEFAULTS
```

NFT displays the information about defaults in the following format:

```
     node::/USER:userid/ACCOUNT:account-string/OSTYPE:ostype
```

where:

| | |
|---|---|
| node:: | is the name of the node for which the defaults listed have been set. |
| userid | is the default user identification at the specified node. |
| account-string | is the default account at the specified node. |

ostype — is the operating system at the specified node.

Example:

```
@NFT(RET)

                 (ESC)
                   ↓
NFT>INFORMATION (ABOUT) DEFAULTS(RET)

Node KL2102::/USER:SKOGLUND/ACCOUNT:341/OSTYPE:TOPS20
NFT>
```

The INFORMATION DECNET command displays the list of accessible DECnet nodes.

The format of the INFORMATION DECNET command is as follows:

```
NFT>INFORMATION (ABOUT) DECNET
```

Example:

```
@NFT(RET)
                 (ESC)
                   ↓
NFT>INFORMATION (ABOUT) DECNET(RET)
 Accessible DECNET nodes are:  DN20A   KL1031  KL2102
NFT>
```

### 7.2.3 COPY Command

The COPY command transfers files from the local node to a remote node or transfers files from a remote node to the local node. Note that only the remote node should be specified.

The format of the COPY command is as follows:

**Local-to-Remote**

```
NFT>COPY (FROM) filespecl/switches (TO) REMOTE::filespec2/switches
```

**Remote-to-Local**

```
NFT>COPY (FROM) REMOTE::filespecl/switches (TO) filespec2/switches
```

where:

REMOTE:: — is the node name of the remote host from/to which the file is transferred.

filespecl — is the file specification of the file to be transferred. The specification must be in the format required by the operating system at REMOTE::.

filespec2 — is the file specification to be given to the transferred file. The specification must be in the format required by the operating system at REMOTE::.

/switches — are one or more of the switches defined in Table 7-1, as required.

Table 7-1
COPY Command Switches

Access Information Switches (valid also with SET DEFAULTS, DIRECTORY, DELETE, SUBMIT, and TYPE commands)

/USER:userid

Sets the user identification associated with the node specified. Provides the access control information only; it does not provide the directory name.

/ACCOUNT:account

Sets the account associated with the user identification at the node specified. Account must be an ASCII string of 1 to 16 characters.

/PASSWORD:password

Sets the password associated with the user identification at the node specified. Password must be an ASCII string of 1 to 8 characters.

Processing Mode Switches

/ASCII

Sets the file processing mode to ASCII.

/IMAGE

Sets the file processing mode to IMAGE. IMAGE indicates that the file is sent or received exactly as stored on disk.

/MACY11

Sets the file processing mode to MACY11. This switch is required to transfer PDP-11 object code. The MACY11 file format is produced by a TOPS-10/20 PDP-11 cross-assembler.

Record Length Switches (used only in combination with one of the processing mode switches)

/FIXED:nn

Defines a file as consisting of fixed length records of nn bytes.

/VARIABLE:nn

Defines a file as consisting of variable length records of maximum size of nn bytes.

DECnet-20 Version 2.1 ASCII-mode file transfers require neither processing-mode nor record-length switches. To understand which switches you should use when executing any other type of network file transfer, you should understand the file systems of the two machines involved in the transfer.

The TOPS-20 file system stores data in units of 512 36-bit words, called pages. Descriptive information about the file is stored in a special "header" page called the File Descriptor Block (FDB). Record formats and attributes are not stored in the FDB. Only the programs which access the file know whether the record format is undefined, stream, fixed, variable, VFC, etc. Only accessing programs know whether the items in a record are characters (SIXBIT, ASCII, EBCDIC, etc.) or fixed or floating numbers.

The following information is, however, kept in the FDB; BYTE SIZE and LENGTH IN BYTES. This lack of knowledge about the file's format makes heterogeneous non-ASCII file transfer somewhat complex. You have noticed that there are file switches for TOPS-20 files such as /VAR and /MACY11. These formats are not native to TOPS-20, nor are they produced or read by any TOPS-20 utility. Following is a description of each of these file formats. These descriptions should allow you to design your data transfer techniques to take full advantage of the file transfer capabilities of DECnet.

1. NO FILE FORMAT SWITCHES ON EITHER FILE.

   If the file transfer is TOPS-20 to TOPS-20, the FDB and the entire file are copied in page size records. All FDB information is retained, and files with holes retain the holes. This is the most efficient homogeneous file transfer format; the files are read and written with PMAPs.

   If the file transfer is not 20 to 20 and the file's byte size is 7 or 36 the data mode defaults to ASCII, otherwise it defaults to /IMAGE.

2. TOPS-20 STREAM ASCII FILE FORMAT (/ASCII)

   Stream ASCII files contain a continuous stream of 7 bit ASCII characters. Logical records are delimited by any of the following characters: ESC, Z, DC1, DC2, DC3, DC4, DLE, FF, VT, LF. The line numbers in line numbered files are ignored by NFT/FAL. Nulls are stripped by NFT/FAL. Both /ASCII/FIX:n and /ASCII/VARIABLE:n are processed as /ASCII except that records longer than n characters are split into two records.

3. TOPS-20 Image File Format (/IMAGE)

   Image format files are considered to be streams of bytes. The bytes are all of the same size from 1 to 36 bits. There is no concept of records or record lengths.

4. TOPS-20 MACY11 File Format (/MACY11)

   The MACY11 file format is the format of object files produced by the MACY11 and DNMAC cross assemblers. An object file produced by MACY11 can be copied to an RSX or VMS system, task built, and run successfully.

A MACY11 file is a 36-bit byte file containing variable length records of the following format. Four bytes are stored in each 36-bit word:

[<2 ZERO BITS><BYTE 2><BYTE 1><2 ZERO BITS><BYTE 4><BYTE 3>].

Each record looks like this:

| | |
|---|---|
| Byte 0 <1> | sync byte |
| Byte 1 <0> | null follows sync |
| Byte 2 <cnt> | low order of (length of "Data" in bytes)+4=[n] |
| Byte 3 <cnt> | high order of (length of "Data" in bytes)+4=[n] |
| Byte 4 <data> | |
| Byte n | (last byte of "Data") |
| Byte n+1 | checksum byte (two's complement add with carry ignored); checksum includes all bytes in record including header |

6 Nulls followed by next record (The nulls are ignored)

5. TOPS-20 Variable Length Record File Format (/VARIABLE:n or /IMAGE/VARIABLE:n)

   A TOPS-20 variable length file suitable for transfer to or from a VMS or RSX type file system consists of a sequence of variable length 8-bit byte records. The first two bytes of each record contain the byte count of the data in the remainder of the record (Low order byte first, high order byte second). Four bytes are stored in each 36-bit word as follows:

   [<BYTE 1><BYTE 2><BYTE 3><BYTE4><4 ZERO BYTES>]

6. TOPS-20 MACY11 Variable Length Record File Format (/MACY11/VAR:n)

   A TOPS-20 MACY11 variable length file consists of a sequence of variable length 8-bit byte records where each record starts on a word or half word boundary and the first two bytes of each record contain the count of the data in the remainder of the record. The count is stored low order byte first, high order byte second. Four bytes are stored in each 36-bit word as follows:

   [<2 ZERO BITS><BYTE 2><BYTE 1><2 ZERO BITS><BYTE 4><BYTE 3>]

7. TOPS-20 MACY11 Fixed Length Record File Format (/MACY11/FIX:n)

   A TOPS-20 MACY11 fixed length record file consists of a sequence of 8-bit bytes stored in 36-bit words where the length of each record is arbitrary (remember that TOPS-20 does not store the record length anywhere). This is the format of task files (.TSK) produced by TKB20 (the fixed record size must be 512). A task file produced by TKB20 can be copied by NFT to an RSX system and run provided that PIP is used on the RSX system to make the copied task file contiguous.

   If the last record is only a record fragment, then different target systems may act differently. Refer to the discussion for each target system. Four bytes are stored in each 36-bit word as follows:

   [<2 ZERO BITS><BYTE 2><BYTE 1><2 ZERO BITS><BYTE 4><BYTE 3>]

8. TOPS-20 Fixed Length Record File Format (/FIX:n or /IMAGE/FIX:n)

   A TOPS-20 fixed length file suitable for transfer to or from a VMS or RSX type file system consists of a sequence of 8-bit bytes. Since TOPS-20 does not store the record size in the FDB, it can be considered to be any length. If the last record is only a record fragment, then different target systems may act differently. Please refer to the section for each target system. Four bytes are stored in each 36-bit word as follows:

   [<BYTE 0><BYTE 1><BYTE 2><BYTE 3><4 ZERO BITS>]

## NFT file specification defaults

The following table shows, for each field in a file specification, whether wildcards can be used, whether it can be defaulted, and if it can, what the default is.

LOCAL FILE SPECIFICATION (after logical name defaulting)

| | | |
|---|---|---|
| NODE:: | Default is the local node (Local node cannot be specified explicitly) | No wildcards allowed |
| DEVICE: | Default is DSK: ** | No wildcards allowed |
| <DIRECTORY> | Default is PS: | Wildcards allowed * |
| FILE-NAME. | Must be supplied in source file specification, will default to the name of the source file if omitted from destination file specification | Wildcards allowed |
| .FILE-TYPE | If not supplied in source file specification it is null; will default to the type of the source file if omitted from the destination file specification | Wildcards allowed |
| .VERSION | Default is most recent version for existing file, or next version for new file | Wildcards allowed |

REMOTE FILE SPECIFICATION

| | | |
|---|---|---|
| NODE:: | Default is local node | No wildcards allowed |
| DEVICE: | No default is provided, the remote node performs the defaulting; for TOPS-20 remotes the default is PS: *** | No wildcards allowed |

| | | |
|---|---|---|
| <DIRECTORY> | No default is provided, the remote node performs the defaulting; for TOPS-20 remotes the default is the argument of the /USER: switch supplied with the node | Wildcards allowed * |
| FILE-NAME. | Must be supplied in source file specification; will default to the name of the source file if omitted from destination file specification **** | Wildcards allowed |
| .FILE-TYPE | If not supplied in source file specification it is null; will default to the type of the source file if omitted from the destination file specification | Wildcards allowed |
| .VERSION | No default is provided, the remote node performs the defaulting | Wildcards allowed |

* If the directory is wildcarded, the access control information (/USER:, /ACCOUNT:, /PASS:) must be valid for every directory included in the wildcard specification. The user is NOT prompted for this information when a new directory is accessed.

** A local file can be on any of the following devices: DSK, LPT, CDP, CDR, PLT, MTA, TTY, and NUL.

*** A remote file must be on a disk device. If the remote file device is a logical name, the logical name will be processed appropriately for that node, except that NFT-20 will always insert file name and file type.

**** The files .;* and .*;* cannot be copied to or from 11s or VAXs.
Examples:

When a sequential file transfer is between two DECSYSTEM-20s, you may allow all fields except the filespec fields to be defaulted by omitting the switches and the node specification that represents the local node.

Each COPY command in the first three examples is valid for a TOPS-20 to TOPS-20 transfer. LOCAL is the name of the local node; REMOTE is the name of the remote node.

Example 1.

```
     @NFT(RET)

             (ESC)              (ESC)
               ↓                  ↓
     NFT>COPY (FROM) *.MAC.* (TO) REMOTE::*.MAC.*(RET)

     PS:<USER>ABC.MAC.3 => REMOTE::PS:<USER>ABC.MAC.3 [OK]
     PS:<USER>XYZ.MAC.7 => REMOTE::PS:<USER>XYZ.MAC.7 [OK]
```

The system responds, in this case, by naming all transferred files ending in .MAC. This use of the wildcard function is permitted only if both nodes support wildcarding.

Example 2.

```
     @NFT RET

           ESC               ESC
            ↓                 ↓
     NFT>COPY (FROM)  ZOOM.*.* (TO) REMOTE::ZOOM.* RET

     PS:<USER>ZOOM.EXE.4 => REMOTE::PS:<USER>ZOOM.EXE.1 [OK]
     PS:<USER>ZOOM.MAC.6 => REMOTE::PS:<USER>ZOOM.MAC.4 [OK]
```

In the above example, the system interprets the wildcard designation for file type, transfers the two files beginning with the file name ZOOM, and defaults the unspecified FROM node name to the local node name. Note also the complete file specification inserted by the system in both examples. You did not need to type the structure (PS:) or user (<USER>).

Example 3.

```
     @NFT RET

           ESC                        ESC
            ↓                          ↓
     NFT>COPY (FROM)  REMOTE::ABC.TXT.3 (TO)  ABC.TXT RET

     REMOTE::PS<USER>ABC.TXT.3 => PS:<USER>ABC.TXT.3 [OK]
```

In this example, the file is being transferred from the remote to the local node, whereas in the first two examples the files were being transferred from the local to the remote site.

Example 4.

```
       NFT RET
     NFT>COPY TPARS.MAC SY5101::DB0:TPARS.MAC RET
     TPARS.MAC.1 => SY5101::DB0:TPARS.MAC;1 [OK]

     NFT>DIRECTORY SY5101::DB0:*.MAC RET

       SY5101::DB0:[200,200]
     TPARS.MAC;1;P775600            6 11264(8)    15-Aug-79 17:02:07
```

The above example differs from the first three examples in two ways. First, a file is being copied from the local system to a non-TOPS-20 system. (SY5101 is an RSX operating system.) Second, guidewords are not used.

Specifying Destination File Processing Mode

For each source file processing mode specified, there is a default destination file processing mode. This default value will be assumed if no destination file processing mode switch is specified. The defaults are:

| Specified Source Mode | | Default Destination Mode/Record Length |
|---|---|---|
| /ASCII | (TO) | /ASCII or /ASCII/VARIABLE |
| /ASCII/FIXED | (TO) | /ASCII |
| /ASCII/VARIABLE | (TO) | /ASCII |
| VMS print file format | (TO) | /ASCII |
| /IMAGE | (TO) | /IMAGE |
| /IMAGE/FIXED | (TO) | /IMAGE/FIXED |
| /IMAGE/VARIABLE | (TO) | /IMAGE/VARIABLE |
| /MACY11 | (TO) | /IMAGE/VARIABLE |
| /MACY11/FIXED | (TO) | /IMAGE/FIXED |
| /MACY11/VARIABLE | (TO) | /IMAGE/VARIABLE |

TOPS-20 NFT permits only the following source/destination file processing mode combinations when transferring a file TO a remote system:

| Local Mode | | Remote Mode |
|---|---|---|
| /ASCII | (TO) | /ASCII |
| /ASCII | (TO) | /ASCII/VARIABLE |
| /IMAGE | (TO) | /IMAGE |
| /IMAGE/FIXED | (TO) | /IMAGE/FIXED |
| /IMAGE/VARIABLE | (TO) | /IMAGE/VARIABLE |
| /MACY11 | (TO) | /IMAGE/VARIABLE |
| /MACY11/VARIABLE | (TO) | /IMAGE/VARIABLE |
| /MACY11/FIXED | (TO) | /IMAGE/FIXED |
| /MACY11/IMAGE | (TO) | /IMAGE |

TOPS-20 NFT permits only the following source/destination file processing mode combinations when transferring a file FROM a remote system:

| Remote Mode | | Local Mode |
|---|---|---|
| /ASCII | (TO) | /ASCII |
| /ASCII/FIXED | (TO) | /ASCII |
| /ASCII/VARIABLE | (TO) | /ASCII |
| /IMAGE | (TO) | /IMAGE |
| /IMAGE | (TO) | /MACY11/IMAGE |
| /IMAGE/FIXED | (TO) | /IMAGE/FIXED |
| /IMAGE/FIXED | (TO) | /MACY11/FIXED |
| /IMAGE/VARIABLE | (TO) | /IMAGE/VARIABLE |
| /IMAGE/VARIABLE | (TO) | /MACY11/VARIABLE |
| /IMAGE/VARIABLE | (TO) | /MACY11 |

### 7.2.4 DELETE Command

The DELETE command deletes files from a remote node.

The format of the DELETE command is as follows:

```
NFT>DELETE (REMOTE FILES) node::filespec/switches
```

Example:

```
     @ nft<RET>
                        <ESC>
                          ↓
     NFT> information (ABOUT)  decnet<RET>
      Accessible DECNET nodes are:  KL2102, KL1031, KS4097, DN200,
     DN20A
     NFT> delete KL2102::sep.txt<RET>
     Access information for node KL2102::/USER:cirino/ACCOUNT:341
     Password: password<RET>
     KL2102::PS:<CIRINO>SEP.TXT.5 [OK]
     NFT>
```

If no access information values have been established before the DELETE command, NFT will prompt for the required access information unless you supply switches with the DELETE command. These switches are effective only for the command in which you specify them. Note that if the remote node is running TOPS-20, no expunge is done.

### 7.2.5 DIRECTORY Command

The DIRECTORY command returns a directory listing of the files at the specified node. The system prints the directory heading and then lists the files in alphabetic order. For each file the following information is listed:

- Name, type, generation number
- Protection code
- Size in pages
- Length in bytes and byte size (in parentheses)
- The date and time the file was originally created or, if modified, the date last modified

The directory heading (node, structure, directory name) and the file name, type, and generation number are always in the format required by the remote site. All other information is listed in TOPS-20 format.

The format of the DIRECTORY command is as follows:

```
     NFT>DIRECTORY (OF REMOTE FILES) node::filespec/switches
```

If no access information values have been established before the DIRECTORY command, NFT will prompt for the required access information unless you supply switches with the DIRECTORY command. These switches are effective only for the command in which you specify them.

Several examples follow. The environment of the DIRECTORY command influences the input/output associated with the command. Therefore, each example is preceded by the conditions that would call for the input as shown and result in the output as shown.

Example 1.

Conditions: The NFT DIRECTORY command is for one file on your own logged-in directory. There is no NFT.INIT file. The DIRECTORY command is the first command given in this NFT session. NFT knows USER and ACCOUNT because you logged in on this node. NFT always prompts for password unless it has been established with a SET DEFAULTS command. NFT does not print passwords.

```
@NFT (RET)

     (ESC)
       ↓
NFT>dirECTORY (OF REMOTE FILES) login.cmd (RET)
Access information for node
KL2102::/USER:KAMANITZ/ACCOUNT:341
Password: password(RET)

  KL2102::PS:<KAMANITZ>
LOGIN.CMD.7;P777700      1 39(36)     14-Sep-79 14:34:01
NFT>
```

Example 2.

Conditions: Same as Example 1 except that an NFT TYPE command is given before the DIRECTORY command. There is no prompt for the password after the DIRECTORY command is given. The TYPE command was the first command and the password was entered in response to the prompt following the TYPE command. NFT remembers the password.

```
NFT>type switch.ini(RET)
Access information for node
KL2102::/USER:KAMANITZ/ACCOUNT:341
Password:password(RET)
EDIT/SAVE:5/ISAVE:5

     (ESC)
       ↓
NFT>dirECTORY (OF REMOTE FILES) login.cmd(RET)

  KL2102::PS:<KAMANITZ>
LOGIN.CMD.7;P777700      1 39(36)      14-Sep-79 14:34:01
NFT>
```

Example 3.

Conditions: The DIRECTORY command is for the complete directory on another TOPS-20 node that is a member of your network. You have an account for node KS4097. There is no NFT.INIT file. The DIRECTORY command is the first command given that requires access to node KS4097. A prompt is given for each access information parameter.

```
               (ESC)
@ NFT(RET)       ↓
NFT> DIRECTORY (OF REMOTE FILES) KS4097::(RET)
Access information for node KS4097::
User: CRUGNOLA(RET)
Account: 341(RET)
Password:password(RET)

 KS4097::PS:<CRUGNOLA>
CALEND.EXE.1;P777700     5 2560(36)  10-Apr-78 11:23:48
DIDDLE.ZZZ.1;P777700     1 6(36)      6-Aug-79 15:51:41
LA36.CMD.1;P777700       1 74(7)     13-Mar-78 16:39:48
LOGIN.CMD.2;P777700      1 21(7)     13-Mar-78 16:36:23
MAIL.TXT.1;P770400       1 175(7)    26-Jul-79 13:01:46
PTYCON.ATO.1;P777700     1 1220(7)   19-May-78 13:34:26
SWITCH.INI.2;P777700     1 39(7)     18-May-78 15:33:10
VT50.CMD.1;P777700       1 28(7)     13-Mar-78 16:37:44
VT52.CMD.1;P777700       1 60(7)     13-Mar-78 16:38:49
ZIP.Q.1;P777700          1 45(7)     14-Jun-78 17:06:25
NFT>
```

Example 4.

Conditions: Same as Example 3 except that the DIRECTORY command uses the wildcard feature to request all files of type .CMD.

```
               (ESC)
@ NFT(RET)       ↓
NFT> DIRECTORY (OF REMOTE FILES) KS4097::*.CMD(RET)
Access information for node KS4097::
User: CRUGNOLA(RET)
Account: 341(RET)
Password: password(RET)

 KS4097::PS:<CRUGNOLA>
LA36.CMD.1;P777700       1 74(7)     13-Mar-78 16:39:48
LOGIN.CMD.2;P777700      1 21(7)     13-Mar-78 16:36:23
VT50.CMD.1;P777700       1 28(7)     13-Mar-78 16:37:44
VT52.CMD.1;P777700       1 60(7)     13-Mar-78 16:38:49
NFT>
```

Example 5.

Conditions: Two NFT Directory commands are directed to an RSX node that is a member of your network. You have an account on SY5101. Your logged-in directory has an NFT.INIT file with a SET DEFAULTS command that establishes values for USER, ACCOUNT, PASSWORD, and OSTYPE for SY5101. The INFORMATION DECNET command is also included in the NFT.INIT file. The DIRECTORY command is the first command given that requires access to SY5101. The directory heading and file specifications are in RSX format. All other output is in TOPS-20 format. All values apply to the remote directory. The first DIRECTORY command is for all files on structure DK0:. The second command is for all files of type .CMD on structure DB0:. The wildcard feature is allowed because it is implemented by RSX.

```
@NFT(RET)
                   (ESC)
                     ↓
NFT>iNFORMATION (ABOUT) decnet(RET)
 Accessible DECNET nodes are:  DN20H  KL1031 KL2102 KL4114
SY5101

      (ESC)
        ↓
NFT>diRECTORY (OF REMOTE FILES) SY5101::(RET)

  SY5101::DK0:[200,200]
INSTALL.CMD;17;P775600    1 1536(8)    27-Dec-78 17:44:26

      (ESC)
        ↓
NFT>diRECTORY (OF REMOTE FILES) SY5101::DB0:*.CMD(RET)

  SY5101::DB0:[200,200]
MERGE.CMD;1;P565600       1 512(8)     12-May-78 15:45:35
FLOPPY.CMD;1;P565600      1 512(8)     12-May-78 15:45:36
PLO.CMD;1;P565600         1 512(8)     12-May-78 15:45:36
PLOT.CMD;1;P565600        1 1024(8)    12-May-78 15:45:37
COPIES.CMD;3;P775600      1 512(8)      7-Sep-79 16:45:14
NFT>
```

### 7.2.6 EXIT Command

The EXIT command ends NFT execution. See the example in the HELP command which follows.

### 7.2.7 HELP Command

The HELP command displays the HELP file for NFT. The HELP file contains a description of all NFT commands and switches. The example that follows illustrates both the EXIT and HELP commands. CTRL/O was typed after the first two sentences of the HELP file. The file is approximately 4 pages and you can read it at your convenience.

Example:

```
$R NFT(RET)
NFT>HELP(RET)
NFT - Network file transfer program       6-Apr-81
```

NFT is the user interface to the network file transfer system. The services NFT provides are actually performed by a FAL (File Access Listener) process at the accessed node.

```
^O...
NFT> EXIT(RET)
$
```

### 7.2.8 PRINT Command

The PRINT command allows an ASCII file to be printed.at a remote node. The network file transfer system does not check to determine that the request is honored. This command is valid only for files that support remote file printing. Note that the file must be on the remote node and be in the format required by the remote node. The print spooling facility must be available at the remote node.

The format of the PRINT command is as follows:

```
       (ESC)
         ↓
NFT>prINT (REMOTE FILES) node::filespec/switches
```

where:

node::  is the node name of the remote host where the file is located.

filespec  is the file specification of the remote file.

/switches  are the access information switches.

### 7.2.9 SUBMIT command

The SUBMIT command allows a Batch control file or command file on a remote node to be submitted to the Batch input queue or command file processor at that node. The network file transfer system does not check to determine that the request is honored. This command is valid only for nodes that support remote command file submission. Note that the file must be on the remote node and be in the format required by the remote node. The batch or command file facility must be available at the remote node.

The format of the SUBMIT command is as follows:

```
       (ESC)
         ↓
NFT>suBMIT (REMOTE FILES) node::filespec/switches
```

where:

node::  is the node name of the remote host where the file is located.

filespec  is the file specification of the remote file.

/switches  are the access information switches.

### 7.2.10 TAKE Command

The TAKE command allows commands to be executed from a command file.

The format of the TAKE command is as follows:

```
NFT>TAKE (COMMANDS FROM) filespec1 (LOGGING OUTPUT ON)
filespec2/switches
```

where:

filespec1 is the file specification of the local command file.

filespec2 (OPTIONAL) is the file specification of the local file for logging output (default is to TTY).

When commands are executed that cause a prompt for access information, the command file execution is momentarily suspended, and you are given the prompt for the access information at your terminal. After you enter the required access information, the command file execution is resumed. This feature allows you to omit passwords from your command files.

TAKE Command Switches

| | |
|---|---|
| /DISPLAY | display program output and commands on terminal during command file execution. |
| /NODISPLAY | suppress terminal output during command file execution. Information is still recorded in the log file. |

### 7.2.11 TYPE Command

The TYPE command displays the file specified on your terminal. The file is transferred in ASCII.

The format of the TYPE command is:

```
NFT>TYPE (REMOTE FILES) node::filespec/switches
```

## 7.3 NFT ERROR MESSAGES

In the course of running NFT, you may receive error messages. NFT error messages preceded by % are warning messages. Warning messages may indicate errors or may give you information. You respond to them by taking the indicated action or adjusting your procedures on the basis of the information given. Error messages preceded by ? are fatal error messages. Except where otherwise stated, fatal errors can be handled by you alone, or by you with the help of a more experienced user.

In most messages, both the cause of the error and the action required are apparent from the text of the message. Where this is not the case, this chapter includes interpretive text following the message.

### 7.3.1 NFT Warning Messages

%File attributes do not match processing mode

> The file attributes at the remote site do not match those specified in the COPY command. This message will be received only if you are reading a file at a remote node running an operating system other than TOPS-20.

%No local node specified, assuming destination file is local

%No local node specified, assuming source file is local

%No remote node specified, assuming destination file is remote

%Password found in command or NFT.INIT file which has world read access

> Remove the password switches from the command file or change the file's protection.

%Remote OS type different from that specified with SET DEFAULT

### 7.3.2 NFT Fatal Error Messages

?Byte size of local file is unusable, 7 assumed

?Cannot do requested file format conversion

> Check the allowable source/destination file processing mode combinations (Section 7.2.3).

?Cannot establish requested mode for input

?Cannot establish requested mode for output

?Cannot get JFN for logical link ... - TOPS-20 text for JSYS error

?Cannot open command file

> Examine the file specification of your TAKE file for errors.

?Cannot open logging file

> Examine the file specification of your LOG file for errors.

?Cannot open logical link ... - TOPS-20 text for JSYS error

?Cannot open PS:NFT.INIT

?Command JSYS failed, type CONTINUE to try again

?EOF detected on logical link

?Error during TAKE file, aborting TAKE command

?Error getting list of available nodes

?Error processing PS:NFT.INIT, aborting processing

?File is not ASCII

?Illegal destination processing mode

?Illegal switch: switch

?Invalid account string

?Invalid destination processing mode

?Invalid password

?Invalid record length

?Invalid SET command

?Invalid switches for local file

?Invalid switches for remote file

?Invalid switch terminator

?Invalid use of wild cards

?Invalid user-id

?Length of account string exceeds 39 characters

?Length of password string exceeds 39 characters

?Length of userid string exceeds 39 characters

?Local status - error text, which includes (MAC:n1 MIC:n2 STV:n3)

> The error text corresponds to the octal numbers n1, n2, and n3, which are error codes defined in the DAP architecture. N1 is the MACRO or functional group reason for the error message. N2 is the specific type of error status. N3 is the secondary error status, whose value depends upon which operating system was running in the remote node. If TOPS-20 was running in the remote node, n3 is the JSYS error code. If an RMS-based operating system, such as VMS, was running in the remote node, n3 is the RMS device error code.

?Logical link reception error - reason text

> The logical link was aborted for the reason specified in reason text.

?Logical link transmission error - reason text

> The logical link was aborted for the reason specified in reason text.

?Logical link was aborted during initial connection - reason text

> The logical link was aborted for the reason specified in reason text.

?Remote file attributes not supported

?Remote node is not responding

?Remote node refused connection ... - disconnect reason text

?Remote status - error text, which includes (MAC:n1 MIC:n2 STV:n3)

The error text corresponds to the octal numbers n1, n2, and n3, which are error codes defined in the DAP architecture. N1 is the MACRO or functional group reason for the error message. N2 is the specific type of error status. N3 is the secondary error status, whose value depends upon which operating system was running in the remote node. If TOPS-20 was running in the remote node, n3 is the JSYS error code. If an RMS-based operating system, such as VMS, was running in the remote node, n3 is the RMS device error code.

?Remote system does not support default mode

?Remote system does not support file submission

The NFT SUBMIT command is not implemented at the remote node.

?Remote system does not support requested mode

?Remote system does not support spooling option

?Remote system does not support wildcard operations

?Remote to Remote transfers not supported

This message is displayed if you use the NFT COPY command to transfer a file from one remote node to another remote node.

?Syntax error in node name or error in local file specification - error text

?Syntax error is node name or local file not found

?Syntax error in remote file name - error text

?TOPS-20 text for JSYS error

This message consists of any appropriate JSYS error message not specifically covered in other messages in this list.

7.3.2.1 **Internal NFT Program Errors** - These fatal errors should not occur. If any one of these errors does occur, you will need the help of your Software Support Specialist. These errors are internal to the NFT program.

```
?Cannot abort close logical link in LLCLOS

?Dap message buffer is full

?Function not implemented

?GLXINI - Unable to obtain run-time systems

?Invalid argument block length for   D$INIT
                                     D$OPEN
                                     D$FUNC

?Invalid link index

?Logical link not open in  D$CLOS
                           LLCLOS

?Too many links requested
```

# CHAPTER 8

# SETHOST (REMOTE LOGIN CAPABILITY)

## 8.1 SETHOST PROGRAM

SETHOST allows a user at a terminal on a TOPS-20 system (running on a DECSYSTEM 2040S or 2060) to log in to a remote TOPS-20 host in a DECnet network. This chapter describes the user of SETHOST in conjunction with a server task on a TOPS-20 host.

SETHOST defines a network source task and establishes a task-to-task network connection between the source task and a target task at the remote host. SETHOST passes source terminal input to the network connection and passes the remote host's output to the source terminal. The program also provides for a special escape character by which the user can exit normally from SETHOST and return to TOPS-20 (EXEC) command level at the local node. Finally, SETHOST monitors the network connection and handles any unexpected break in the connection.

## 8.2 LOGGING IN TO A REMOTE HOST USING SETHOST

You invoke SETHOST by entering some form of the SETHOST command in response to the TOPS-20 prompt at your terminal. SETHOST expects you to specify the remote host's node-name and a special escape character to be used to exit from SETHOST. If you do not specify a special escape character, SETHOST uses CTRL/Y by default.

To specify the remote host's node-name and use CTRL/Y as the special escape character, use the following SETHOST command format:

```
     @SETHOST node-name
```

SETHOST sets CTRL/Y as the special escape character and prints the following message:

```
     [Type CTRL/Y to return to node node-name]
```

where node-name is the name of your local host. If you enter an invalid node name (or if no physical connection exists), SETHOST prints:

```
     ?Connection broken.  Reason:  39:  No path to destination node
```

and terminates processing. You are returned to EXEC command level.

To specify a different special escape character, use one of the following SETHOST command formats:

```
     @R SETHOST
or
     @SETHOST
```

SETHOST then prompts for the special escape character:

```
Escape character ( CTRL/Y ):
```

Press the control key ( CTRL/ ) while you also type one of the following characters: A, B, D, E, G, H, K, N, P, V,. X, Z; then type RET . However, do not enter a control character you will use on the local or remote host. In addition, if you have enabled the trapping of any control character, you cannot use that control character as the SETHOST escape character. If you enter only RET , SETHOST uses CTRL/Y as the special escape character.

If you did not enter a node name, as shown in the two SETHOST command lines above, SETHOST prompts you for one as follows:

```
Host name:
```

Enter the remote host's node name. If you enter RET , SETHOST reissues the prompt until you enter a valid node name or a CTRL/C . If you enter an invalid node name, SETHOST responds by printing an error message and terminating processing.

After a successful connection to the remote host, SETHOST prints the remote host's standard ·banner message on your terminal. After the banner message is printed, you may perform any function - such as logging in - which is permitted by the remote host.

## 8.3 EXITING FROM A REMOTE HOST USING SETHOST

To exit normally from the remote host, type the special escape character selected (or defaulted) in the initial SETHOST dialogue. (See Section 8.2 for SETHOST's response and your possible actions when an abnormal disconnection occurs.) When you enter the special escape character, SETHOST prints the following message on your terminal:

```
[Connection broken, back at node node-name,
Type CONTINUE to resume connection.]
```

You may continue SETHOST execution from the point of the interrupt by entering the CONTINUE command. SETHOST responds:

```
%Reconnecting to remote node...
```

At this point, the connection is restored and the terminal is again connected to the remote host.

NOTE

> It is strongly recommended that you log off the remote system before breaking the network connection between the local system and the remote system. Jobs detached on -20s by other than a DETACH command will autologout after 5 minutes.
>
> In addition, if you do not intend to resume the connection (by typing the CONTINUE command), use the TOPS-20 RESET command to break the logical link. Failure to do this limits the number of available links for other jobs.

## 8.4 CONTROLLING SCROLLING ON A REMOTE NODE

The default characters that start and stop scrolling on the remote node sometimes differ from those that do so on the local host.

On the local host, CTRL/S is the default character that stops scrolling, and CTRL/Q is the default character that starts scrolling. Typing CTRL/Q causes scrolling to resume whether scrolling stopped because you typed CTRL/S or because the system paused at the end of a page on your terminal.

When you have used SETHOST to log in at a remote node, CTRL/S still stops scrolling, and CTRL/Q still causes scrolling to resume after you have typed CTRL/S However, DECnet-20 does not pass these characters to the remote node. It is the local host that recognizes these characters and controls scrolling upon receiving them, even when the display on the terminal is from a remote node. Also, if you have entered terminal pause mode on the remote node, and the system has paused because a display has reached the end of a page, CTRL/A is the default character for starting scrolling again. Typing CTRL/Q (since it is not passed to the remote node) has no effect in this case. In addition, when you are in terminal-pause mode on the remote node, CTRL/A is the default character that the remote node recognizes for both starting and stopping scrolling.

You can use the TERMINAL PAUSE command to cause the CTRL/A character to stop and start scrolling on the local host as well as on the remote node. You can also use the TERMINAL PAUSE command to assign any two characters of your choosing - except CTRL/S and CTRL/Q - for controlling scrolling; to do so:

1. Log in at the local host.

2. Enter a TERMINAL command that defines which keys will start and stop scrolling.

3. Use SETHOST to log in at a remote node.

4. Again enter the same TERMINAL command that you entered in step 2.

To cause these TERMINAL commands to be in effect when you log in at the local host and remote node, you can enter them into the LOGIN.CMD file in each of those nodes.

Example 4 in Section 8.5 demonstrates this procedure.

## 8.5 SAMPLE TERMINAL SESSIONS USING SETHOST

The examples in this section illustrate uses of SETHOST.

Example 1.

The following example shows user KAMANITZ at a terminal on system KL2102. He first logs in to KL2102, then uses SETHOST to connect to the remote system KL2137. By entering the remote system's node name on the SETHOST command line, KAMANITZ allows SETHOST to use (CTRL/Y) as the special escape character. He then logs in to system KL2137 as user CRUGNOLA, performs functions on system KL2137, and logs off KL2137. After logging off of system KL2137, he presses (CTRL/Y) to return to system KL2102. From KL2102, he issues a LOGOUT command to log off system KL2102.

```
 KL2102 Development System, TOPS-20 Monitor 5.1(5012)
@LOGIN KAMANITZ password 341(RET)
 Job 16 on TTY106 19-Sep-79 14:41:15
@SETHOST KL2137(RET)
[Type ^Y to return to node KL2102]

 KL2137 Load-Test System, TOPS-20 Monitor 5.1(5012)
@LOGIN CRUGNOLA password 341(RET)
 Job 12 on TTY50 19-Sep-79 14:41:27
     .
     . (User performs functions on system KL2137.)
     .
@LOGO(RET)
Killed Job 12, User CRUGNOLA, Account 341, TTY50,
  at 19-SEP-79 14:41:49, Used 0:00:02 IN 0:00:21

(User enters  (CTRL/Y)  here which is not echoed.)

[Connection broken, back at node KL2102,
Type CONTINUE to resume connection]
@LOGO(RET)
Killed Job 16, User KAMANITZ, Account 341, TTY 106,
  at 19-SEP-79 14:46:53, Used 0:00:01 IN 0:05:37
```

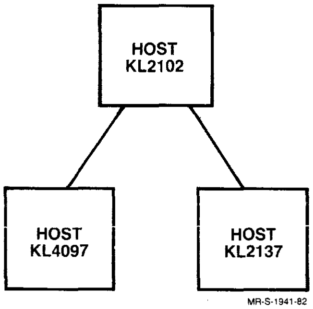
Example 2.

In the following example, user KAMANITZ logs in to system KL2102. Then, he uses SETHOST to connect to system KL2137 and specifies <CTRL/B> as the special escape character. Once he is connected to KL2137, he logs on as user CRUGNOLA and begins listing his directory. While system KL2137 is printing information about his directory, he presses <CTRL/B> to gain the attention of SETHOST on system KL2102. From KL2102, he enters CONTINUE in response to the EXEC prompt and SETHOST reconnects to system KL2137. System KL2137 resumes printing information about his directory. (Note that the directory listing does not contain the lines that would have printed during the time used to escape from and return to system KL2137.) User KAMANITZ then logs off system KL2137, returns to system KL2102, and logs off system KL2102.

```
 KL2102 Development System, TOPS-20 Monitor 5.1(5012)
@LOGIN KAMANITZ password 341<RET>
 Job 62 on TTY106 19-Sep-79 14:50:59
@SETHOST<RET>
Escape character (   <CTRL/Y>  ): (User enters <CTRL/B><RET>)
Host name:KL2137<RET>

 KL2137 Load-Test System, TOPS-20 Monitor 5.1(5012)
@LOGIN CRUGNOLA password 341<RET>
 JOB 13 ON TTY50 19-Sep-79 14:53:09
@VDIR<RET>

  PS:<CRUGNOLA>
CALEND.EXE.1;P777700       5 (User enters    <CTRL/B>  . It does not
echo.)
[Connection broken, back at node KL2102,
Type CONTINUE to resume connection]
@CONTINUE<RET>
Reconnecting to node...
10-6-AUG-79 15:51:41
 LA36.CMD.1;P777700      1 74(7)   13-MAR-78 15:39:48
 LOGIN.CMD.2;P777700     1 21(7)   13-MAR-78 15:36:23
 MAIL.TXT.1;P770404      1 175(7)  26-JUL-79 13:01:46
 SWITCH.INI.2;P777700    1 39(7)   18-MAY-78 15:33:10
 VT50.CMD.1;P777700      1 28(7)   13-MAR-78 15:37:44
 VT52.CMD.1;P777700      1 60(7)   13-MAR-78 15:38:49
 ZIP.Q.1;P777700         1 45(7)   14-JUN-78 17:06:25

 TOTAL OF 13 PAGES IN 9 FILES
@LOGO<RET>
Killed Job 13, User CRUGNOLA, Account 341, TTY 50
  at 19-Sep-79 14:55:45, Used 0:00:03 IN 0:02:36

(User enters   <CTRL/B>   .  It does not echo.)

[Connection broken, back at node KL2102,
Type CONTINUE to resume connection]
@LOGO<RET>
Killed Job 62, User KAMANITZ, Account 341, TTY 106,
  at 19-Sep-79 14:56:53, Used 0:00:01 IN 0:05:37
```

Example 3.

This example shows the use of SETHOST to log in to a remote host that is not an adjacent node in the network. The network is configured according to the following diagram:



User CRUGNOLA on host KL4097, a DECSYSTEM 2040S or 2060, wants to log in to host KL2137, also a DECSYSTEM 2040S or 2060. To do so, he first logs in to host KL4097. Then, he uses SETHOST to establish a network connection to host KL2102, which is running DECnet-20 Version 3.0, and uses the default special escape character. Once the connection to host KL2102 is established, user CRUGNOLA logs in to host KL2102 as user KAMANITZ. From host KL2102, KAMANITZ uses SETHOST to establish a network connection to host KL2137, specifying CTRL/B as the special escape character.

Note that a different special escape character must be used for the connection between hosts KL2102 and KL2137. If the same special escape character were used for both network connections, SETHOST on KL4097 would trap the special escape character and return control of the user's terminal to host KL4097. This would interrupt the connection between hosts KL4097 and KL2102. Once this occurred, no means would exist to gain access to host KL2102 on the existing connection.

After the network connection is established between hosts KL2102 and KL2137, user KAMANITZ logs in to host KL2137 as user OSMAN. OSMAN performs some functions on host KL2137, then logs off and enters CTRL/B to return to host KL2102. Once at host KL2102, user KAMANITZ logs off and enters CTRL/Y to return to host KL4097. Once there, user CRUGNOLA logs off.

```
 KL4097 Load-Test System, TOPS-20 Monitor 5.1(5012)
@LOG CRUGNOLA password 341 RET
 Job 7 on TTY33 25-Sep-79 10:18:05
@SETHOST KL2102 RET
[TYPE  CTRL/Y  TO RETURN TO NODE KL4097]

 KL2102 Development System, TOPS-20 Monitor 5.1(5012)
@LOG KAMANITZ password 341 RET
 Job 42 on TTY217 25-Sep-79 10:19:17
@SETHOST RET
Escape character ( CTRL/Y ): (User enters  CTRL/B  )
Host name: KL2137 RET

 KL2137 - Gus The Languages System, TOPS-20 Monitor 4(3046)
@LOG OSMAN password RET
 Job 20 on TTY214 25-Sep-79 10:20:58
    .
    .  (User performs functions on host KL2137)
    .
@ LOGO RET
Killed Job 20, User OSMAN, Account MONITOR, TTY 214
  at 25-Sep-79 10:21:30, Used 0:00:02 in 0:00:31

(User enters  CTRL/B  .  It does not echo.)

[Connection broken, back at node KL2102,
Type CONTINUE to resume connection]
@LOGO RET
Killed Job 42, User KAMANITZ, Account 341, TTY 217
  at 25-Sep-79 10:23:18, Used 00:00:01 in 0:05:00

(User enters  CTRL/Y  . It does not echo.)

[Connection broken, back at node KL4097,
Type CONTINUE to resume connection]
@ LOGO RET
Killed Job 7, User CRUGNOLA, Account 341, TTY 33,
  at 25-Sep-79 10:26:31, Used 0:00:08 in 0:07:26
```

Example 4.

The following example demonstrates how to cause pressing the "a" key to start scrolling and the "b" key to stop scrolling on both the local host and the remote node. User KAMANITZ logs in to system KL2102. He uses the TERMINAL command to define the keys that will control scrolling. He uses SETHOST to connect to system KL2137 as user CRUGNOLA, and types the same TERMINAL command that he typed while logged in at system KL2102. He performs functions on system KL2137, and as he does so, types the letter "a" any time he needs to start scrolling and the letter "b" any time he needs to stop scrolling. He logs off system KL2137 and types CTRL/Y to return to system KL2102.

He performs functions on system KL2102, and again types the letter "a" any time he needs to start scrolling and the letter "b" any time he needs to stop scrolling. Finally, he issues a LOGOUT command to log off system KL2102.

```
 KL2102 Development System, TOPS-20 Monitor 5.1(5012)
@LOGIN KAMANITZ password 341(RET)
 Job 16 on TTY106 19-Sep-79 14:41:15

 (ESC)               (ESC)      (ESC)     (ESC)
   ↓                   ↓          ↓         ↓
@terMINAL (MODE IS) pauSE (ON) chaRACTER 142 AND UNPAUSE ON 141(RET)

@SETHOST KL2137(RET)
[Type   (CTRL/Y)   to return to node KL2102]

 KL2137 Load-Test System, TOPS-20 Monitor 5.1(5012)
@LOGIN CRUGNOLA password 341(RET)
 Job 12 on TTY50 19-Sep-79 14:41:27

 (ESC)               (ESC)      (ESC)     (ESC)
   ↓                   ↓          ↓         ↓
@terMINAL (MODE IS) pauSE (ON) chaRACTER 142 AND UNPAUSE ON 141(RET)

(The user performs functions on system KL2137, and as he does so,
types the letter "a" any time he needs to stop scrolling and the
letter "b" any time he needs to start scrolling.)

@LOGO(RET)
KILLED JOB 12, USER CRUGNOLA, ACCOUNT 341, TTY 50,
  AT 19-SEP-79 14:41:49, USED 0:00:02 IN 0:00:21

(User enters   (CTRL/Y)   here which is not echoed.)

[Connection broken, back at node KL2102, Type CONTINUE to  resume
connection]

(The user performs functions on system KL2102, and as he does so,
types the letter "a" any time he needs to stop scrolling and the
letter "b" any time he needs to start scrolling.)

@LOGO(RET)
KILLED JOB 16, USER KAMANITZ, ACCOUNT 341, TTY 106,
  AT 19-SEP-79 14:46:53, USED 0:00:01 IN 0:05:37
```

# APPENDIXES

# APPENDIX A

# DISCONNECT OR REJECT CODES

The disconnect or reject codes in Table A-1 are defined by NSP and are sent and retrieved by network tasks with the network functions of the MTOPR monitor call.

Table A-1
Disconnect or Reject Codes

| Symbol | Value | Meaning |
|---|---|---|
| .DCX0 | 0 | No special error |
| .DCX1 | 1 | Resource allocation failure |
| .DCX2 | 2 | Destination node does not exist |
| .DCX3 | 3 | Node shutting down |
| .DCX4 | 4 | Destination process does not exist |
| .DCX5 | 5 | Invalid name field |
| .DCX6 | 6 | Process too busy |
| .DCX7 | 7 | Unspecified error |
| .DCX8 | 8 | Third party aborted the logical link |
| .DCX9 | 9 | User abort (asynchronous disconnect) |
| .DCX11 | 11 | Undefined error code |
| .DCX21 | 21 | Connect initiate (CI) with illegal destination address |
| .DCX24 | 24 | Flow control violation |
| .DCX32 | 32 | Too many connections to node |
| .DCX33 | 33 | Too many connections to destination process |
| .DCX34 | 34 | Access not permitted |
| .DCX35 | 35 | Logical link services mismatch |
| .DCX36 | 36 | Invalid account |
| .DCX37 | 37 | Segment size too small |
| .DCX38 | 38 | Process aborted |
| .DCX39 | 39 | No path to destination node |
| .DCX40 | 40 | Link aborted due to data loss |
| .DCX41 | 41 | Destination logical link address does not exist |
| .DCX42 | 42 | Disconnect confirmation |
| .DCX43 | 43 | Image data field too long |

# APPENDIX B

## DECnet OBJECT TYPES

The object types listed in Table B-1 are taken from the Network Services Protocol, Version 3.2 documentation. Object type codes are expressed in decimal. DECnet-20 will, in addition, recognize a number of object names in place of object types. Object names that are currently supported are shown in Table B-1.

Object type 0 (TASK) can only be used by a source task in order to address a target task. Object types 1 through 127 can be used by any system task; however, the task must have WHEEL or OPERATOR privileges enabled. Object types 128 through 255 are available to all network tasks.

Table B-1
DECnet Object Types

| Object Type Code | Object Name | Function |
|---|---|---|
| 0 | TASK | General task, user process |
| 1 | | File Access (FAL/DAP-Version 1) |
| 2 | | Unit Record Services |
| 3-4 | | Reserved for DECnet use |
| 5 | | RSX-11M Task Control-Version 1 |
| 6 | | Reserved for DECnet use |
| 7 | NRM | Node Resource Manager |
| 8-14 | | Reserved for DECnet use |
| 15 | | RSX-11M Task Control-Version 2 |
| 16 | | System TALK Utility |
| 17 | FAL | File Access (FAL/DAP-Version 4) |
| 18 | | RSX-11S Remote Task Loader |
| 19 | NCU | NICE process |
| 20-62 | | Reserved for DECnet use |
| 63 | | DECnet Test Tool (DTR) |
| 64-127 | | Reserved for DECnet control |
| 128-255 | | Reserved for customer extensions |

APPENDIX C

# GLOSSARY

asynchronous transmission

Transmission in which the time intervals between transmitted characters may be of unequal length because each character contains its own start and stop bits. This is also known as start/stop transmission.

computer network

An interconnection of computer systems, I/O devices, and communications facilities.

connect

The process of creating a logical link.

connect password

A 1- to 39-character password used to validate access privileges between tasks on a network.

DAP (Data Access Protocol)

A set of standardized formats and procedures that facilitate the creation, deletion, transfer, and access of files between a user process and a file system existing in a network environment.

data transmission

The sending of data from one computer to another over a physical link, or from one task to another over a logical link.

DDCMP

Digital Data Communications Message Protocol. A formal set of conventions designed to provide error-free, sequential transmission of data over physical links.

disconnect

The process of closing a logical link.

DMC11

A single line microprocessor-based interface to the network. The DMC11 is a synchronous Direct Memory Access DMA device.

DN20

A communications front end.

DNMAC

The DNMAC program is the cross assembler for PDP-11 macro source files.

down-line loading

Transmitting a program's memory image over a logical link and loading and starting the program on a computer at another node.

DTE20

The hardware interface between the KL10 main processor in a DECSYSTEM-2040S/2060 and the PDP-11 processor in the DN20 communications front end.

duplex

Simultaneous independent transmission in both directions. Also referred to as full-duplex or two-way simultaneous.

FAL (File Access Listener)

The FAL program resides on a DECnet host and acts as the target for requests made by the NFT programs residing on remote DECnet hosts. The FAL program is responsible for determining a user's access privileges to a requested file and the subsequent honoring or rejecting of the request.

full-duplex

See duplex.

half-duplex

Transmission in either direction, but not in both directions simultaneously. Also referred to as two-way alternate.

host computer

A computer at a network node that primarily provides services such as computation, data base access, special programs, or programming languages to other nodes in the network.

interrupt message

A high-priority message used to inform another task of some significant event.

local node

A relative term indicating the node at which your task is executing or at which your terminal is logged in.

local NSP

NSP executing in the local node.

local task

A task executing at a local node.

logical link

A virtual data path between two tasks in a network that permits them to communicate. A physical link can contain many logical links.

loop-back

A mode of operation where data transmitted by a network task is reflected at some point along the communication path and is returned to the originating task.

modem

In networks, a device that makes computer signals capable of being transmitted over telephone lines (also known as a dataset).

MOP

Maintenance Operation Protocol. A formal set of messages and rules used to load and dump computer memory as well as test a communications link between two adjacent nodes.

NCP

NCP is the name of the network control program that processes network control commands. For DECnet-20, NCP is part of the program, NMLT20.

network

An interconnected or interrelated group of nodes. In this manual, network is synonymous with computer network.

network dialogue

An exchange of information between two tasks in a network.

network task

A task engaged in, or willing to engage in, a network dialogue. In NSP documentation, a network task is also referred to as a network object.

NFT (Network File Transfer)

A program that allows you to access or delete files residing on DECnet hosts that provide network file access capabilities. NFT initiates the service requests that will be carried out by the FAL program.

NICE protocol

NICE is the acronym for the Network Information and Control Exchange protocol that enables various DIGITAL computers to access the information and control facilities of remote nodes on the same network.

NMLT20

The Network Management Layer running under TOPS-20. NMLT20 provides network management functions for DECnet-20.

node

An endpoint of any branch of a network, or a junction common to two or more branches of a network. A node is a processor plus communications software and constitutes one end of a physical link in a network. In this manual, the DECSYSTEM-20 and any communications front ends are all considered nodes.

node name

A 1- to 6-character name uniquely identifying a node within a network. Node names can be any combination of the characters A through Z, and 0 through 9. A node name must contain at least one alphabetic character.

node number

A number uniquely identifying a node within a network.

NSP

Network Services Protocol. A formal set of conventions used in DECnet to perform network management and to exchange messages over logical links. NSP also refers to the software that implements the NSP protocol. (In the text, NSP refers to the software; NSP protocol is used to refer to the protocol.)

packet

A group of bits, comprising data and control information, which is transmitted as a composite whole over a physical link. The data, control, and possibly error control information are arranged in a specified format.

physical link

A communications path between two adjacent nodes. This can be in the form of a dial-up line, leased line, radio, satellite link, or a channel-to-channel connector such as a DTE.

protocol

A formal set of conventions or rules governing the format and relative timing of message exchange.

remote node

A node in a network that is not your local node.

remote NSP

NSP executing in a remote node.

remote task

A task executing in a remote node.

server task

Also known as a target task, an alternate designation for a task that has declared itself willing to accept a network connection, usually to provide some system service.

source node

The node at which the request for a connection is initiated or from which a message is transmitted.

source task

The task in which the request for a connection is initiated or from which a message is transmitted.

synchronous transmission

Transmission in which time intervals between transmitted characters are of equal length. Multiple characters can be transmitted without start or stop bits following an initial synchronizing sequence.

target node

Also known as a server task, the node at which the request for a connection is accepted or rejected or to which a message is transmitted.

target task

Any task that has declared itself willing to accept a network connection.

# INDEX

DECnet–20 User's Guide
AA–J679A–TM

## 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 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 _______________________________ Telephone __________________

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 PUBLICATIONS**
200 FOREST STREET MRO1–2/L12
MARLBOROUGH, MA 01752

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

Cut Along Dotted Line