BITNET on MTS

University of Michigan Information Technology Division Consulting and Support Services

Contents

1.	Introduction			3	
	1.1	Overvi	ew of Network Services	3	
	1.2	Additi	onal Information	4	
2.	Getting Started				
			ET Addresses		
			Determining an Address		
	2.2		ng Messages		
			ing Messages		
	2.4		ng Files		
			ing Files		
	2.6 Direct and Indirect Addresses				
			ges to Computers on Other Networks		
			Domains		
~				0	
3.	σ				
	3.1		Things to Consider About a File Transfer		
		3.1.1	Text Files		
		3.1.2	Object Files		
		3.1.3	Plot Files		
		3.1.4	Data Files		
		3.1.5	MTS Line Numbers		
		3.1.6	File Types		
		3.1.7	File Attributes		
		3.1.8	Encoding		
		3.1.9	Record Length Limits	13	
		3.1.10	Carriage Control	13	
		3.1.11	Extra Blanks at Ends of Lines	14	

3.2	Sending Files		14
	3.2.1		
	3.2.2	2 Using \$CONTROL and *EXPORT*, *PRINT*, and *PUNCH*	16
	3.2.3	B Displaying Status of BITNET Jobs	17
	3.2.4		
	3.2.5	5 \$CONTROL Options for *EXPORT*, *PRINT*, and *PUNCH*	18
3.3 Receiving Files			19
	3.3.1		
	3.3.2		
	3.3.3	3 Selection Criteria for *IMPORT*	20
	3.3.4		
	3.3.5		
4. Disp	patche	98	23
		Summary and Sample Terminal Session	
Sum	ımary	of Basic Procedures	26
	Send	ling Messages and Files	26
		eiving Messages and Files	
Sam		erminal Session	
	-		
Appendi	ix B:	Computers and Network Software	30
		-	
		Glossary	

1. INTRODUCTION

BITNET (Because It's There Network), a computer network joining universities and research centers, allows you to exchange messages and files with BITNET users all over the world.

1.1 OVERVIEW OF NETWORK SERVICES

BITNET is administered by Educom, a consortium of colleges and universities, which promotes cooperation in computing, communications, and information technology. It is an educational and research tool, and you are encouraged to bear in mind the following guidelines when you use it:

- (1) BITNET may only be used for non-commercial purposes.
- (2) File transfers in general should be limited to 300,000 bytes (or approximately 300,000 characters).

Host computers on BITNET are connected to one another by permanent, leased telephone lines. Information sent over the network is relayed by being passed from *host to host* (or from *node to node*) through *links* until it reaches its destination. At an intermediate node, a file is received in its entirety and stored before being forwarded to the next node. This means that if a link or node fails temporarily, you do not lose any information from your file or message because the preceding node has a complete copy of the transmission. However, all intermediate links and nodes must be available for you to exchange interactive messages.

BITNET provides the following services:

- (1) <u>File transfer</u>. A file that you send over BITNET may contain any kind of information, including text, source programs, object code, data, or listings. File transfer is *transparent*; that is, there is no restriction on the bit patterns that may occur in the data being transferred. Passwords are not required (unlike FTP).
- (2) <u>Messages</u>. Also called *electronic messages, electronic mail, E-mail*, or just *mail*, messages are sent from one user to another and are placed in the recipient's *mailbox* (a file on a host computer connected to the network). The recipient does not have to be logged on (signed on) at the time the message is delivered; it will be held in the recipient's mailbox until it can be viewed.
- (3) <u>Dispatches</u>. An immediate person-to-person communication that is displayed on the recipient's terminal. If the recipient is not logged on, the dispatch is discarded and the sender is notified.

Some services, which are provided by other types of computer networks, are *not* offered by BITNET. These include:

- (1) $\frac{\text{Remote log-on.}}{\text{BITNET.}}$ You cannot establish a terminal session on a remote node through
- (2) <u>Program interface</u>. You cannot use a program you have written to establish and terminate network connections with programs at other nodes.

BITNET originated as a quick-and-dirty way of exchanging mail among a group of universities in the northeastern U.S., all of whom ran standard IBM operating systems. There are now versions of the BITNET software available for other operating environments, but BITNET is really based on the IBM network job entry system. BITNET messages can really be thought of as virtual card decks traveling over long-distance wires from virtual punch to virtual reader.

Because it is easy and relatively inexpensive to connect to BITNET for hosts running IBM or DEC VMS systems, BITNET has grown very quickly in a short time. There are well over 1000 machines on BITNET in the United States today. BITNET also extends to Canada (where it is called NETNORTH) and to Europe (where it is called EARN).

1.2 ADDITIONAL INFORMATION

The first part of this memo is introductory. To use it, all you need to know about MTS is how to sign on, and you should probably also know how to edit a file. This introductory information about MTS can be found in the publication

Introduction to MTS—Getting Started, Tutorial T7003

If you want to know more about the MTS command language and MTS Message System (messages sent from MTS through BITNET start out in the Message System), see

MTS Volume 1: The Michigan Terminal System, Reference R1001 MTS Volume 23: Messaging and Conferencing in MTS, Reference R1023

Information on how to use various public terminals to MTS can be found in the following publications:

Introduction to Terminals and Microcomputers—Getting Connected, Tutorial T7005 MTS Volume 4: Terminals and Networks in MTS, Reference R1004

"VM" in this document refers to the IBM VM/SP (Virtual Machine System Product) operating system. More information about this operating system can be found in the IBM manual, *Virtual Machine System Product, Quick Guide for Users, Release 3.*

2. GETTING STARTED

This section is intended to get you started using BITNET. Much more information is contained in the later sections of this memo. See Appendix A for a summary of the basic procedures explained in this section and for a sample terminal session that illustrates their use.

All you need to use BITNET is an account on one of the following computers:

System	Computer	Location
UM-MTS	IBM 3090	Computing Center
UB-MTS	IBM 3090	Computing Center

2.1 BITNET ADDRESSES

When you send a message or transfer a file through BITNET, you specify a destination with a two-part address that uniquely identifies the recipient. The first part of the address is the *userID* (also called *log-on ID* or *signon ID*) of the recipient on the destination host computer, and the second part of the address is the BITNET node name of that computer. You enter the address in the form:

userID@node

The userID is from one to eight characters long, and although systems vary, the userID will generally consist only of letters and numbers.

The *node* part of the address is also a unique one- to eight-character name consisting of letters and numbers. Current node names for U-M host computers are:

System	Node Name
UM-MTS	UMICHUM
UB-MTS	UMICHUB

When your file or message is received, the system informs the recipient of your BITNET address. In the case of messages, this allows you to make a direct reply. Once you know the BITNET address of a recipient, you can use either of the methods (file or message) to communicate with the user identified by that address.

2.1.1 Determining an Address

To find out the BITNET address of another installation from MTS, you can send an inquiry to the postmaster:

SEND TO POSTMASTER

From any other mail system, send an inquiry to

POSTMASTER@UM.CC.UMICH.EDU

There are several ways to find out a person's individual BITNET address:

- (1) you may call or write to the person beforehand and ask them for their address, or
- (2) give the person your address and ask them to send you a message (this will give you their return address).

The postmaster normally does not know the addresses of individuals at other installations.

When you give your colleagues your electronic mail address, you will need to supply your MTS userID and the name of the node you will be using in the form

USERccid@UMICHUM or USERccid@UMICHUB

For example if your userID is ABCD, your UM-MTS mailbox address on BITNET is

userabcd@umichum

where "abcd" is your MTS userID. In general, systems other than MTS are unable to generate message addresses containing personal names. So, to instruct MTS that the left half of the message address is not a personal name, you should tell your colleagues to prefix USER to your userID, as in the example above.

2.2 SENDING MESSAGES

To use BITNET to send electronic messages from MTS, simply sign on, enter the command \$MESSAGE, and supply the appropriate BITNET address, as described above. For example (using the BITNET address CUNYVM for the system at the City University of New York), to send a message from MTS to the log-on ID JANEJONZ on the CUNYVM system, first sign on to UM-MTS or UB-MTS and issue the commands:

\$message
send to janejonz@cunyvm

You will then be prompted for the text of your message. After you finish typing the text, press the ENTER key twice, and you will be asked whether you want to EDIT, DISPLAY, or POST your message. When you are ready to send your message, enter

post

(Until you give the POST command, your message is not actually released.)

2.3 RECEIVING MESSAGES

Incoming messages are gathered in a *mailbox*, and each time you sign on, and whenever you issue the \$MESSAGE command, you will be notified of any new messages in your mailbox. You can then retrieve them with the commands:

\$message retrieve

6 Getting Started

Note, however, that if the incoming transmission contains over 300,000 bytes (approximately 300,000 characters), it will be considered a file, and you will not be notified that it has arrived. Enter the command \$SYS Q INCOMING, as described in "Receiving Files," below to determine if you have a file waiting for you.

2.4 SENDING FILES

You can send a file from MTS to another system on BITNET by supplying a network destination on the MTS \$CONTROL command and then copying the file to one of the pseudodevices *EXPORT*, *PRINT*, or *PUNCH*. You can also specify other information on the \$CONTROL command, such as a file name, file class, and encoding type, if you wish. But you must always give DESTINATION. For example, you might enter the command:

```
$control *export* destination=janejonz@cunyvm
```

Then, you would enter the following commands to copy your file to *EXPORT* and release it:

```
$copy filename *export*
$release *export*
```

The various options that you can use on the \$CONTROL command for sending files are described in detail in the section "\$CONTROL Options for *EXPORT*, *PRINT*, and *PUNCH*."

2.5 RECEIVING FILES

Like incoming messages, files sent to you on MTS must have an address in the form

USERabcd

where *abcd* is your MTS userID. (Files you send *from* MTS are properly identified automatically.)

Files arriving in MTS are placed in an incoming file queue for your ID. (You can only pick up files in the ID to which they are addressed; it is not possible to intercept files for other IDs.) Files that you do not copy from the queue within four weeks are destroyed.

To see a list of the incoming files queued for your ID, enter the command

\$SYS Q INCOMING (short for \$SYSTEMSTATUS QUEUE INCOMING)

The response will show a job name, job number, time, date, type (print or punch), class, and file name. In general, these will have the values assigned to them by the sender of the file.

You transfer files from the queue to a file in your ID by using the MTS pseudodevice *IMPORT*. Typically, you would use the MTS \$COPY command to copy *IMPORT* to an MTS file; for example:

\$copy *import* myfile

where "myfile" is a permanent MTS file (that you have created previously) or a temporary file (one that begins with a minus sign). Additional options for *IMPORT* are described in the section "Receiving Files."

2.6 DIRECT AND INDIRECT ADDRESSES

Messages are sent either directly to a single recipient or indirectly to more than one recipient through a message relay system, called a *mailer*. The MTS Message System manages the details of mailer vs. non-mailer messages (and also encoding). You simply address your messages with one or more recipients, as in:

\$message
send to janejonz@cunyvm, johndoe@cunyvm

2.7 MESSAGES TO COMPUTERS ON OTHER NETWORKS

BITNET connects to several other computer networks, so host computers connected to these networks can communicate with BITNET sites. As a result, you can exchange messages with colleagues at these *non-BITNET* sites. All you need to do is add a *domain* name to the userID and node address; this associates a message destination with the correct network.

2.7.1 Domains

Domain names keep the thousands of network addresses organized so messages will get to the right host computer. BITNET is itself a domain. Implicitly, all messages constructed in the form *userid@node* are addressed to the BITNET domain.

Within each domain, every address is unique, just as the telephone numbers within a particular area code are unique, but two telephones in different area codes could have the same number. There is no confusion because the area code serves to "qualify" the telephone number.

When you dial from one telephone to another in the same area code, you do not dial the area code itself. When, however, you call a telephone in a different area code, you must give the area code. Similarly, when you send a message from one BITNET host computer to another, you do not need to specify a domain.

When you send a message to a colleague on a host computer on a network other than BITNET, the domain name *must* be given, as in:

userid@node.domain

For example, the domain address of the postmaster at the University of Wisconsin looks like this:

```
postmaster@wiscvm.wisc.edu
```

In this address, WISCVM.WISC.EDU is a domain name where EDU is a top-level domain. (Some other top-level domains are COM, for commercial sites, and MIL, for military sites.) WISC.EDU is a subdomain of EDU, and WISCVM.WISC.EDU is a further subdomain (which usually is the node name, or host computer).

3. MORE ABOUT TRANSFERRING FILES

BITNET allows you to share programs, data, and documents with colleagues on other systems. Although BITNET attaches no significance to the content of a file, there are some conventions for including additional information (such as a file name or file class) to indicate how it should be treated at the receiving system. For example, you should usually send a file containing a program listing as a "print" type file, because it is capable of being printed at the receiving system. These conventions are described in this section.

When you send a file over BITNET, the sending system makes a copy of the file and holds it in a separate storage area, called a *spool file*, until it is transmitted on the network. After this spool copy has been made (a process that takes only a few seconds), you can destroy or alter the original file or sign off the system without disturbing the spool copy. The spool file is never destroyed until delivery of the entire file is accepted by the next node on the delivery path.

To receive a file, you must take explicit action, such as copying the contents of the spool file into a permanent file under your userID. Permanent files are never automatically created on the recipient's userID; a person or program must authorize the receipt of each file.

Some items of note:

- (1) A separate agent (person or program) controls each end (sending and receiving) of the file-transfer process; you use certain commands to send a file and certain other commands to copy a file that has been sent to you.
- (2) You can only initiate file transfer *to* another user; you cannot initiate the sending of another user's file to your own userID. (Note, however, that some sites may have file servers that can send files to you in response to commands.)
- (3) You do not log on to the remote system as part of file transfer, and thus, no passwords are necessary.
- (4) During file transfer, it does not matter whether the sender or the recipient are logged on and engaged in other activity.

3.1 SOME THINGS TO CONSIDER ABOUT A FILE TRANSFER

Files are assumed to consist of 8-bit bytes and are transferred with complete transparency, that is, there are no reserved bit patterns that indicate boundaries between lines or the end of the file. All bytes are considered to be data. Because of transparency, it is not necessary for BITNET to inspect or understand the structure of the data it is transferring.

Except for the effects of blank padding and truncation, as explained in following sections, the received file is an exact copy of the sender's file. (You can use encoding to ensure that padding and truncation do not occur; see below the section "Encoding").

There are limits on the length of each record (line) of data in the file which are summarized below. If a particular record exceeds these limits, it will generally be truncated during the transfer, without notification and with a resulting loss of data.

When sending files between systems, you must be aware of the differences between the systems, and the possibility that one receiving system may interpret the data differently from another.

3.1.1 Text Files

You can generally transfer files containing readable text, such as source programs, documents, and listings, with few side effects. Except for possible truncation, the structure of the data will be meaningful at the remote system.

3.1.2 Object Files

With some exceptions, you *cannot* safely transfer object code, such as the output of compilers and assemblers. The exceptions are:

- (1) If the destination system runs MTS, you can safely transfer object files; however, some object files may suffer some truncation. Note that optimized MTS object files produced by the *OBJUTIL or *LINKEDIT utilities may have records exceeding the network truncation limit (80 bytes for punch-type files and 132 bytes for print-type files, see below the section "Record Length Limits"). You, therefore, must first process these files with one of these two utilities using the output record length option (ORL) to produce a file with sufficiently short records. (See *MTS Volume 2: Public File Descriptions*, Reference R1002, for information about *OBJUTIL and *LINKEDIT.)
- (2) If the destination system does not run MTS but is otherwise an IBM System/370-compatible machine (see Appendix B), you can safely transfer card-image (80-byte) object files. Most MTS language translators produce object files in this format. However, some translators, as well as the object-file utility programs *OBJUTIL and *LINKEDIT, produce an optimized object file format that is incompatible with other System/370 operating systems. You can use either program to convert an existing object file to card-image format so that it can be transferred (see the ORL, output record length, option in the descriptions of these programs in MTS Volume 2).

Keep in mind that even when the object code can be transferred to and loaded on the destination machine, differences between operating system subroutines and services may render the program unusable at the destination.

3.1.3 Plot Files

MTS plot files are not transferrable between systems because they use a data format unique to MTS and because they may exceed the network maximum record length. Transfer the program that generates the plot file rather than the plot file itself.

3.1.4 Data Files

Data files can be the input, intermediate result, or output of a program. Files containing human-readable information such as numerical data in a fixed format may be treated as text files and except for possible truncation, are safe to transfer. Such files would include program output produced via formatted-I/O services, such as Fortran formatted I/O.

In general, if the data is binary code, containing machine representations of integer or floating-point numbers, you can safely transfer it only to an IBM System/370-compatible machine that offers the same machine data types. Certain 2's-complement integer representations can be used across machines but only if you have detailed knowledge of the data representations of the sending and receiving machines.

An example of data containing binary information is the output produced by Fortran unformatted I/O. Files of unformatted data are often produced as the intermediate result of a Fortran program where the only purpose of the data is to be read into another program. Unformatted files contain program data in internal machine form. You can exchange these files between System/370 computers if you are careful not to exceed network maximum record lengths (see below the section "Record Length Limits").

3.1.5 MTS Line Numbers

When files are transferred between two MTS systems, line numbers are not preserved. Therefore, if you want to transfer a file where the file structure is sensitive to line numbers, such as in macro, source, or object libraries, you should delete the library directory from the file before you transmit it (by using line-number ranges, for example). The file directory should then be reconstructed at the destination using the *MACUTIL or *OBJUTIL utility programs (which are described in *MTS Volume 2*).

3.1.6 File Types

All files transferred over BITNET are one of two types: *print files* and *punch files*. Although MTS treats them in essentially the same way, some systems make sharp distinctions in using the two types. For example, usually only punch files are encoded.

A print file is a text file suitable for printing. It can, for example, be sent directly to a printer (as opposed to a person) on a destination system. Print files are restricted by most systems to containing records 132 or fewer bytes in length, and typically contain vertical formatting controls (carriage control; see below the section "Carriage Control").

A punch file contains card-image records, 80 or fewer bytes in length. Although card punches are rarely used today, a punch file is suitable to send directly to a card punch. The 80-byte record is the basis for encoded forms of files. See the descriptions of *EXPORT*, *PRINT*, and *PUNCH* in the section "Pseudodevices," for more information.

3.1.7 File Attributes

You can attach one or more attributes to a file to help process it at its destination. None are required, and some assume default values if you do not specify them.

You can supply a *file name* so that the original file name from the sending system will be preserved during transfer. This name may then be used by software at the receiving system to recreate the file with the same name. Because file-naming conventions vary, you must know the conventions of the destination system when you choose the name.

You can also specify a *file class*. File class is designated by a single-letter code that indicates how the file is to be processed at the destination. Like file names, interpretation of file class varies from system to system. When you send a file to a printer or card punch, the system may use the file class to

provide instructions to the computer operator for setting up the device. A file class of D, for example, may denote a particular type of paper. When you transfer a file to a person or program, file class is often a convenient means of grouping files with similar characteristics. Use and interpretation of the file class is by mutual agreement between you and the recipient of your file. For example, a particular server program may accept incoming files with class C as meaning the file contains commands to the server, whereas a class of S could mean that the file contains data that the server is to store locally.

One file-class convention in BITNET is that a class M file is a message or mail file. Therefore, you should avoid using class M for any file except mail.

See section "\$CONTROL Options for *EXPORT*, *PRINT*, and *PUNCH*" for detailed descriptions of file attributes and section "Using \$CONTROL and *EXPORT*, *PRINT*, and *PUNCH*" for instructions for using them.

3.1.8 Encoding

You can send files via BITNET with one of the three following encoding types. From MTS, you supply the appropriate keyword, to specify the type of encoding, on the \$CONTROL command, as in

\$control *punch* destination=userjazz@westnode encoding=netdata

(1) NONE. In this case, each record is sent as is. The maximum record length allowed by MTS is 32,760 bytes, but the more restrictive record length of 80 bytes (punch) or 132 bytes (print) may apply at the destination system. Therefore, you should not use this form for files with long records.

Transmission without encoding is mandatory when carriage control (print) or stacker selection (punch) is to be effective immediately at the destination, that is when you send a file to a remote printer or punch rather than a remote userID. (See the section "Carriage Control" for more about this subject.)

- (2) NETDATA. NETDATA breaks up long records into pieces of 80 bytes each so they will fit through a punch stream. (It is equivalent to the CMS SENDFILE command. Files sent with this type of encoding can be read on a VM system with the RECEIVE command.)
- (3) DISKDUMP. This is similar to NETDATA in that it breaks up long lines into 80-byte records, but it is older than NETDATA and is understood by some systems that do not accept NETDATA. You will have to contact the remote system to know whether this restriction applies. (This command corresponds to the CMS DISK DUMP command.)

You may want to use encoding for several reasons. Primarily, encoding is an easy way to avoid truncation, and it allows you to attach a file name. Also, encoding ensures transparency for punch files.

Encoding is reversible. When you specify ENCODING, your original file is converted to encoded form before transmission. It passes over the network in encoded form, and is decoded at the destination system. The decoded file is an exact copy of your original.

MTS can send or receive files in any of the three ways described above. Because not all systems accept all encoded forms, you should check with someone at that system before sending an encoded file.

When you receive a file on MTS, you do not have to take any special measures for dealing with encoding. MTS automatically detects and decodes NETDATA and DISKDUMP formats for you.

3.1.9 Record Length Limits

Not all BITNET network software is capable of accommodating lengthy data records (lines), so if you have long lines in a file you are transferring, you run the risk of losing characters from the ends of them.

The maximum length line, or record, that can be transferred via BITNET is 32,760 bytes. Unfortunately, the practical limit for file transfer is the shortest line length allowed at any system between the sender and the receiver, which may be as short as 80 bytes for punch-type files and 132 bytes for print-type files.

Because of the hazards of truncation, you should usually encode files containing long records, except in the following cases:

- (1) If the destination of the file transfer (for example, a remote printer) requires data in an unencoded form, you cannot use encoding. This will often be the case if each record of the file contains a carriage-control (CC) character as the first character. (See next section for more on carriage control.)
- (2) If the destination system is not capable of decoding an encoded file, you cannot use encoding.
- (3) If the file you want to transfer consists of records less than 80 bytes (for a punch-type file) or 132 bytes (for a print-type file), you can use encoding if you wish, but it is not necessary.

3.1.10 Carriage Control

Print- and punch-type files may contain carriage-control characters as the first character of each record. These indicate the vertical position of the line on the page (for a printer), or the card pocket in which to stack the card (for a punch).

There are two types of carriage control: logical and machine. They correspond to the MTS @CC and @MCC I/O modifiers, respectively. In MTS, logical carriage control is the most common. (Machine carriage control characters are hexidecimal representations.)

The carriage-control character for a line is the first character of that line, and the data in the line begins with the second character. If a line has no carriage-control character, the first character is treated as the first byte of data for that line.

Note that a single transferred file may contain a combination of logical, machine, and no carriage control. Each line of the file has its own carriage control, independently of every other line.

When you send a file, you must tell BITNET whether the file has carriage-control characters. When you receive a file, you should preserve the carriage-control information if you will want to eventually send the file to a printer or punch. Conversely, you may want to discard the carriage controls. On MTS, whether to honor or ignore carriage-control characters is controlled by the @CC and @MCC I/O modifiers. (MTS I/O modifiers are described in *MTS Volume 1: The Michigan Terminal System*, Appendix A, "Files and Devices." Logical and machine carriage control is described in the same volume in Appendix H, "Carriage Control.")

Logical Carriage Control

When you specify @CC, logical carriage-control characters are honored when copying to *PRINT* or *PUNCH*. Use @¬CC to indicate that the file does not contain logical carriage control. @CC is the default for *PRINT* and *PUNCH*, and @¬CC is the default for *EXPORT*.

When you copy a file from *IMPORT*, use @CC to specify that logical carriage-control characters in the file being received are to be preserved. The default for *IMPORT* is @¬CC, which means that carriage-control characters, if present, will be discarded. Note that specifying @CC has no effect on a line that contains either no carriage-control character or a machine carriage-control character.

Machine Carriage Control

Similarly, when you specify @MCC, machine carriage-control characters are honored when you copy a file to *PRINT* or *PUNCH*. @¬MCC is the default for *PRINT*, *PUNCH*, and *EXPORT*.

When you copy a file from *IMPORT*, specify @MCC if you want machine carriage-control characters to be preserved. @¬MCC is the default for *IMPORT*, which means that such carriage control, if present, will be discarded. Note that specifying @MCC has no effect on a line that contains either no carriage-control character or a logical carriage-control character.

Some "immediate" machine carriage-control characters denote carriage positioning only, without recording any data. The data accompanying such a carriage-control character is ignored when the file is sent to a printer. If you want to keep the data, you *must* specify @-MCC.

See the section "Carriage Control in Incoming Files" for example commands.

3.1.11 Extra Blanks at Ends of Lines

Although BITNET assures transparency for data while it is being transferred, trailing blanks may be added or removed from each record at the point of submission or reception, depending on the operating system.

In MTS, the @TRIM I/O modifier controls truncation of trailing blanks. (See *MTS Volume 1: The Michigan Terminal System*, Appendix A, "Files and Devices," for more on @TRIM.) MTS normally does not trim trailing blanks from print- and punch-type files; to trim them, specify @TRIM.

Some systems pad punch-file records to 80 bytes when they are received, by appending trailing blanks. The process of truncation and padding forces all punch-file records to a constant length of 80 bytes. You should use encoding when you want transparent file transfer to systems that perform punch record padding.

To maintain file transparency, records, by default, are written to *EXPORT* and read from *IMPORT* without trimming. You must explicitly specify @TRIM to override this behavior. Therefore, use *EXPORT* when you want complete transparency.

3.2 SENDING FILES

From MTS, you can use the following pseudodevice names on the \$CONTROL command to send files over BITNET: *EXPORT*, *PRINT*, *PUNCH*, as in:

\$control *export* destination=janejonz@cunyvm [options]

See the sections below for more information.

3.2.1 Pseudodevices

Although you can use the three pseudodevices interchangeably for many purposes, some transfers require one or the other. The one you choose depends largely on the receiving system. In general:

- (1) Use *EXPORT* for general file transfer. It provides all of the functions of *PRINT* and *PUNCH* (see below) and is most useful to transfer exact copies of files when you don't intend to print or punch the data. When unencoded, the same record-length restrictions apply as for *PRINT* and *PUNCH*. Therefore, when you send files with arbitrary record lengths, you should use some form of encoding. The default for *EXPORT* is completely transparent transmission of data; that is, the following MTS I/O modifiers are in effect: @¬CC, @¬MCC, @¬TRIM.
- (2) Use *PRINT* when you want to send a file directly to a printer, or if it will eventually be printed. You should also use *PRINT* if you are sending an unencoded file with records greater than 80 bytes to a VM system. By default, *PRINT* assumes a logical carriage-control character in column one. MTS logical carriage-control characters 1, 0, -, +, and blank are the only carriage controls recognized in the same way by both MTS and remote systems. *PRINT* files are received by VM as PRINT files and may have a record length of up to 132 bytes. Blanks are trimmed by default, so you should not use *PRINT* for exact copies of data files.
- (3) Use *PUNCH* when you want to send card-image data such as object decks or Fortran source programs, or when you are sending to a real card punch on a remote system. *PUNCH* assumes, by default, a logical stacker-selection character (carriage-control character) in column one, unless you specify @SS. A *PUNCH* file will become a PUNCH file on VM (fixed-length, 80-byte records). Because most systems can accept a maximum of only 80 bytes per record, blanks will be trimmed from lines longer than 80 bytes. Many systems will also pad short records to 80 bytes.

Summary

The characteristics of the three network output pseudodevices are summarized in the following table:

Characteristic	*EXPORT*	*PRINT*	*PUNCH*
File type	Print/Punch	Print	Punch
Carriage control	@¬CC,@¬MCC	@CC,@¬MCC	@SS,@¬MCC
Blank trimming	@¬TRIM	@¬TRIM	@¬TRIM
Encoding used?	Yes	No	Yes
Unencoded max. length	132 (Print) 80 (Punch)	132	80

3.2.2 Using \$CONTROL and *EXPORT*, *PRINT*, and *PUNCH*

EXPORT, *PRINT*, and *PUNCH* all accept a common set of control options. In addition, some options are specific to *EXPORT*. By using these options either on the MTS \$CONTROL command or with the MTS CONTROL subroutine, you can affect the way a file is sent to and processed at the destination. The form of the \$CONTROL command is:

\$CONTROL *xxxxx* DESTINATION=userid@node[option...]

where "*xxxxx*" is *EXPORT*, *PRINT*, or *PUNCH*; "userid@node" is the BITNET address of the person you are sending the file to, and "option" is one of the options listed in the section "\$CONTROL Options for *EXPORT*, *PRINT*, and *PUNCH*"; for example:

\$control *export* destination=janejonz@cunyvm encoding=netdata

You can use more than one option on the \$CONTROL command (or call to CONTROL) by separating each option from the next by one or more blanks. You must *always* specify DESTINATION for a particular job.

You can use \$CONTROL commands to create a file-transfer job or to modify the characteristics of an existing job. You can issue \$CONTROL commands, and affect the entire job, at any time before you give the command

\$RELEASE *xxxxx*

For example, if you copy data to *EXPORT*, and then issue the command

\$control *export* encoding=diskdump

followed by more data to *EXPORT*, the entire job will be sent with DISKDUMP encoding. A \$CONTROL command overrides any previous instance of that same command for the job.

When you issue the CONTROL *xxxxx* command, a network job is automatically created and held. The message

xxxxx assigned job number nnnnnn

signals the creation of the job.

When you actually send the file (using \$RELEASE *xxxxx*), the response will be:

xxxxx jobname released DEST=userid@node [attributes]

The message confirms the file destination, and "attributes" confirms the file name, file class, encoding, and other attributes that you assigned to the job on the \$CONTROL command. The job is released to the control of the network and will be transmitted as soon as possible. Between the time you create the job and the time it is released, the job is said to be *open*, and you can add more data to it, or change its attributes with \$CONTROL commands. Once released, however, no aspect of the job can be changed, except that you can cancel it.

3.2.3 Displaying Status of BITNET Jobs

After you submit a network job, you can display its status with the command

\$SYS Q * (short for \$SYSTEMSTATUS QUEUE *)

which shows all jobs you have submitted from the current userID, both local MTS and BITNET. If you want to know the status of a particular job, issue the command

\$SYSTEMSTATUS QUEUE {jobname | jobnumber}

where "jobname" is the job name you gave the job (or was assigned by MTS), and "jobnumber" is the job number. (If you wish, you can specify a job name of your choosing by adding the JOBNAME keyword, for example, JOBNAME=ANDY, on the \$CONTROL command.)

After you submit a job, it is held in a queue until it can be sent. A network job waiting for its turn is shown by the system response to the \$SYSTEMSTATUS QUEUE command as

wxyz:jobname (jobnumber) is awaiting export to userid@node, Pn, after m jobs.

where "wxyz" is the MTS userID that submitted the job, "jobname" and "jobnumber" are the job name and job number assigned to the job, "userid@node" is the destination BITNET address, "n" is the job priority, and "m" is the number of jobs that must be exported first.

A job currently being sent on a link is displayed as

wxyz:jobname (jobnumber) is being sent to userid@node

For a short time after the job is sent, you will receive the message

wxyz:jobname (jobnumber) was sent at time

where "time" is the time the job was transmitted on an outgoing link. (Please note that this does not necessarily indicate that the file has reached its destination; it may be stored at an intermediate node.)

3.2.4 Cancelling BITNET Jobs

You can cancel network jobs while open—following submission and prior to transmission—or during transmission. To cancel an open network job, issue one of the following commands:

\$CANCEL *xxxxx*

or

\$CONTROL *xxxxx* CANCEL

where "*xxxxx*" is one of the pseudodevices *EXPORT*, *PRINT*, or *PUNCH* (or *IMPORT*). You will see the confirmation message:

xxxxx jobname canceled

and all spooled data will be discarded. If you have already submitted the job, issue the command

\$CANCEL {jobname | jobnumber}

where "jobname" and "jobnumber" are those assigned to your job. If you cancel the job while it is being transmitted, transmission will cease immediately, and the adjacent node will discard the portion of the job received to that point.

3.2.5 \$CONTROL Options for *EXPORT*, *PRINT*, and *PUNCH*

In the following descriptions, underlining shows minimum abbreviations.

CLASS=c

Default: A

"c" is a one-letter class from A to Z. It only has significance at the destination system, and its meaning will vary from system to system. Class M indicates mail within BITNET.

COMMENT=comment COMMENT="comment of more than one word"

This command sets a comment that may print on the header of a remote print or punch job.

COPIES=n

This command sets the number of copies produced by a remote printer or punch. Normally, this value is ignored when a file is sent to a person.

DESTINATION=userid@node

Default: No destination

Default: No file name

You *must* include DESTINATION on all file transfers (from MTS) over BITNET. DESTINATION identifies where you want the file sent. The "node" part is the remote machine name, for example, CUNYVM. The "userid" is the individual user identifier used on that machine. You will need to know both of these to send a file (as you do to send messages).

The node name of UM-MTS is UMICHUM and UB-MTS is UMICHUB. The "userid" under MTS is your MTS userID preceded by the word USER. For example, if your MTS userID is ABCD, your "userid" for network purposes would be USERABCD.

ENCODING={NONE | NETDATA | DISKDUMP} Default: NONE

This command sends the file with the indicated encoding type. MTS files are always sent as variable-record-length files. (VM has both fixed- and variable-length types). Normally, you should encode only punch data streams; in which case, encoded records up to 32,760 bytes may be sent. Otherwise, the length is restricted to 80 bytes for punch files or 132 bytes for print files under VM.

FILE=filename FILE="file name"

This attaches "filename" to the file as it is sent. If the name contains blanks or special characters, you must enclose it in quotation marks. The enclosing quotes do not form part of the file-name value. Case is important.

When sending to a VM system, you can specify the file name, file type, and file mode portions of the VM file name by enclosing the three portions in quotation marks and separating them with blanks; for example, FILE="fn ft fm".

LINECOUNT=n

This command sets the number of lines per page. This may or may not be interpreted by the remote system.

TYPE={PRINT | PUNCH}

Default: PUNCH

This command is valid only for *EXPORT*. It allows *EXPORT* to produce either print- or punch-type files. You can use either NETDATA or DISKDUMP encodings with the default type (PUNCH).

3.3 RECEIVING FILES

As explained in the "Getting Started" section, to receive files from BITNET on MTS, you \$COPY your file from the pseudodevice *IMPORT*, as in:

\$COPY *IMPORT* myfile

(Note that *IMPORT* may be assigned to any logical I/O unit, such as SCARDS. Decoding of NETDATA or DISKDUMP files is done automatically.)

3.3.1 Carriage Control in Incoming Files

By default, carriage-control characters are removed from the file as it is read in MTS. (This is similar to the behavior under VM.) To retain the carriage control (when receiving a print file, for example), include the @CC or @MCC modifiers (explained in the section "Carriage Control"). Although @MCC is rarely used under MTS, it is common with VM systems.

A print file received from VM can be printed under MTS using the following commands:

\$COPY *IMPORT*@MCC mtsfilename \$COPY mtsfilename *PRINT*@MCC

Note that when VM files containing ANSI carriage control are printed under VM (for example, by the VM PRINT command), the ANSI carriage control is implicitly converted to machine carriage control.

3.3.2 Partially Read Files, KEEP, and CANCEL

Each time you issue the \$COPY *IMPORT* command, you copy a single file from the queue in the order shown by the \$SYS Q INCOMING command. When the file has been read completely, the queued copy is destroyed. If the incoming file is only partially read (for example, the \$COPY command ends prematurely because you are out of disk space), it is not destroyed but is placed back in the incoming queue. If you reissue the \$COPY command, the file will be read from the beginning.

It is possible to override the action of destroying the incoming file after it is read. The KEEP option (see below) requeues the file just as would be the case if it is only partially read. Each time it is

requeued, the file is reset to the beginning.

When a file is placed back in the incoming queue instead of being destroyed, you will receive the message:

IMPORT jobname requeued

No message is issued when the file is destroyed.

The CANCEL option (see below) overrides the action of keeping a partially read file. CANCEL causes the incoming file to be discarded. Cancellation takes effect when all outstanding uses (for example, MTS commands or loaded programs) of *IMPORT* are concluded, so it is possible to use a KEEP command to override a previous CANCEL, and a CANCEL command to override a previous KEEP.

3.3.3 Selection Criteria for *IMPORT*

It is not necessary to read the incoming jobs in the order shown by \$SYS Q INCOMING. There are some control options to help you receive jobs in an arbitrary order according to the following criteria that you can specify singly or jointly on a \$CONTROL *IMPORT* command (see below, "\$CONTROL Options for *IMPORT*," for detailed descriptions):

- (1) File class. You can restrict *IMPORT* to files that have the specified file class.
- (2) File name. You can restrict *IMPORT* to files that have the specified file name.
- (3) Job name. Job names assigned by MTS (when no job name is assigned by the sender) are unique. Therefore, when you specify a job name assigned by MTS you are identifying a single job. If you assign your own job names, duplicate names are possible, in which case, selection by job name restricts *IMPORT* to all jobs with the name you specify.
- (4) Job number. Job numbers are assigned by MTS and are always unique. Therefore, if you specify the job number, you will always select just one job.

If there are no jobs in your incoming queue with the particular set of criteria you select, you will receive the message

No matching *IMPORT* jobs available for ID WXYZ

3.3.4 \$CONTROL Commands

The above selection criteria are among the options you can specify on the MTS \$CONTROL command (or in the CONTROL subroutine). The \$CONTROL command has the form

\$CONTROL *IMPORT* [option]

where "option" is one of the options listed in the following section. You can list more than one option on the \$CONTROL command (or call to CONTROL) by separating each option from the next by one or more blanks.

When you issue the \$CONTROL command to *IMPORT*, you will receive the message

IMPORT held

When you are finished copying your incoming file from *IMPORT*, enter the command

\$RELEASE *IMPORT*

3.3.5 \$CONTROL Options for *IMPORT*

In the following descriptions, underlining shows minimum abbreviations.

CANCEL

This command causes the system to stop reading the current incoming job, and to destroy it. The job's data is discarded The cancellation only takes effect when there are no remaining uses of *IMPORT*, and therefore, the effect of CANCEL may itself be canceled by a subsequent KEEP command (see below).

CLASS=c

This command restricts *IMPORT* to only reading jobs of class "c". The first job with class "c" (and concurrently meeting the other selection criteria, if any) will be the next job read.

FILE=filename

FILE="file name"

This restricts *IMPORT* to reading only jobs with the specified file name. VM file names should be enclosed in quotes and separated by blanks; for example, "fn ft fm". Case is important. The first job with the specified file name (and concurrently meeting the other selection criteria, if any) will be the next job read.

HOLD

This command explicitly holds the current *IMPORT* job in the open state. It is used to prevent automatic release of the file (to be destroyed or requeued) when, for example, a \$COPY command completes. A HOLD is removed by the CANCEL or RELEASE commands or the MTS \$RELEASE *IMPORT* command.

JOBNAME=jobname

This command restricts *IMPORT* to reading only jobs with the specified job name. Under MTS, more than one job can have the same job name, so this specification is not necessarily unique. The first job with the specified job name (and concurrently meeting the other selection criteria, if any) will be the next job read.

JOB#=jobnumber

This command specifies that the next job to be read is the one with the indicated job number. Under MTS, job numbers are unique, so this is sufficient to specify the next job to be read. If the job does not concurrently meet the other selection criteria, no job will be selected.

KEEP

This command modifies the action of destroying the incoming job after it is read to the end of the file. When KEEP is in effect, the job is requeued (when all outstanding uses have ended), regardless of whether it has been read completely or not. You receive a message when this action is taken. A CANCEL command overrides the action of a previous KEEP. A KEEP command overrides the action of a previous CANCEL, if the cancellation has not yet taken effect.

RELEASE

This command explicitly releases an earlier explicit or implicit HOLD. If the file was previously read to the end and you did not specify KEEP, the job is destroyed; otherwise, it is requeued. You receive a message when this action is taken. RELEASE does not take effect until all other outstanding uses of *IMPORT* (by loaded programs, for example) have ended.

4. DISPATCHES

You can send dispatches to other Bitnet sites using the MTS Message System. Dispatches are short, immediate messages that are displayed on the recipient's terminal as they arrive.

To send an MTS dispatch, use the SEND DISPATCH command of \$MESSAGESYSTEM. Local MTS users are identified by name, userID, task number, or device name; for example, the session below sends a dispatch to Jane Doe, a registered local MTS user.

```
# $messagesystem
Mailbox WXYZ: 0 new, 7 old messages
@ send dispatch to "Jane Doe"
Text:
? Don't forget the meeting at 11!
?
Post, edit, display, destroy, help, etc.?
@ post
```

To send a Bitnet dispatch, use the Bitnet address, in the form

userid@host

substituting for userid the name, MTS userID, task number, or device name; and substituting for host the name of a Bitnet host computer (computers on networks other than Bitnet cannot be the target of dispatches).

If JANEDOE@WESTNODE is the Bitnet address of a remote correspondent at the Bitnet host named WESTNODE, the following example would send a dispatch to her.

```
@ send dispatch to janedoe@westnode
Text:
? Don't forget the meeting at 11!
?
Post, edit, display, destroy, help, etc.?
@ post
```

Receiving a dispatch sent by someone else requires no action on your part; it simply appears on your terminal as it arrives. It is prefixed by the Bitnet address of the sender so that you know where the dispatch came from, and so that you may reply if you wish:

@ from janedoe@westnode: I'll be there.

Network server machines as well as individual users can be dispatch recipients. The server treats the text of the dispatch as a command. For example, the popular LISTSERV file server responds to the command

\$mess dispatch to listserv@pucc text="Index" OK

by sending you a copy of the file called INDEX using Bitnet file transfer. (It will appear in your *IMPORT* at some later time.)

Dispatches are typically delivered within 2 to 60 seconds, depending on the distance of the destination site. If the person you are sending to is not signed on to the remote computer, you will get

an error message back. The exact text of this message depends on the operating system of the remote computer, but looks something like:

@ from @westnode: janedoe not logged on

This message, which comes from the operating system at WESTNODE and not from Jane Doe, is called a *system dispatch*. It can be distinguished by its lack of the "userid" portion of the Bitnet address.

Another example of a system dispatch is the progress notification you receive when you send a file to another Bitnet site. When you copy and release a file to *EXPORT*, it begins its journey hop-by-hop through the network. For each hop, you receive a progress dispatch:

@ From @RPICICGE: SENT FILE 0123 ON LINK UCONNVM TO WESTNODE JANEDOE
@ From @UCONNVM: SENT FILE 0123 ON LINK YALEVM TO WESTNODE JANEDOE
@ From @YALEVM: SENT FILE 0123 ON LINK CUNYVM TO WESTNODE JANEDOE

@ From @CUNYVM: SENT FILE 0123 ON LINK WESTNODE TO WESTNODE JANEDOE

MTS terminal support has recently been enhanced to give greater immediacy to incoming dispatches. Typically, this means that the terminal will interrupt a wait for input to show you a dispatch that just arrived. If you are in the visual mode of the editor, or one of the other full-screen packages, you will not see the dispatch until you exit full-screen mode or press a PF key (depending on the terminal type).

You may not want to receive dispatches all the time, or you may want to be selective about those you receive. The \$SET DISPATCHES command has been enhanced to give you greater control over which dispatches are displayed. The basic command is

\$SET DISPATCHES={ON | OFF}

which turns dispatch display on and off, respectively. In addition, through the use of *dispatch filters* you can selectively receive dispatches from local MTS users only, dispatches from remote users, or system dispatches. The form of this command is

\$SET DISPATCHES(filter,...)={ON | OFF}

There are three classes of filters, all of which must be passed for the dispatch to display:

- (1) Dispatch display may be turned on or turned off entirely (the VIEW filter).
- (2) A dispatch is either a locally originated dispatch or a network-originated (Bitnet) dispatch. Each type may be turned on or off independently (the LOCAL and NETWORK filters).
- (3) A dispatch is either a user-generated dispatch or a system-generated dispatch. Each type (the USER and SYSTEM filters) may be turned on or off independently. (Note: Some dispatches from network server machines appear as user dispatches, not system dispatches. This is because the network server is running as a user on a remote machine and is not part of the operating system.)

Here are some examples of the options you can use:

\$SET DISPATCHES (LOCAL) = OFF

turns off dispatches coming from local senders (senders on the same host);

\$SET DISPATCHES (NETWORK) = OFF

turns off dispatches coming from Bitnet senders;

\$SET DISPATCHES (SYSTEM) = OFF

turns off system dispatches from both remote hosts and local software.

The complete form of the \$SET DISPATCHES command is

 $SET DISPATCHES(VIEW | LOCAL | NETWORK | USER | SYSTEM | ALL) = {ON | OFF}$

The commands

\$SET DISPATCHES(VIEW)=ON

and

\$SET DISPATCHES=ON

are identical. VIEW is used to preserve the other filter settings if you do not want to see any dispatches. VIEW may be combined with the other options, as in:

\$SET DISPATCHES(VIEW, SYSTEM, NETWORK) =ON

You can set all options on or off simultaneously with the command:

\$SET DISPATCHES(ALL) = {ON|OFF}

The sign-on default for MTS terminal sessions is equivalent to the command:

\$SET DISPATCHES(VIEW,LOCAL,NETWORK,USER,SYSTEM) =ON

That is, all categories of dispatches are displayed by default.

You can use the command \$DISPLAY DISPATCHES to see the current dispatch filter setting, as in:

\$DISPLAY DISPATCHES
Dispatches(view,local,network,user)=on, Dispatches(system)=off

APPENDIX A SUMMARY AND SAMPLE TERMINAL SESSION

SUMMARY OF BASIC PROCEDURES

The basic procedures for sending and receiving messages and files are summarized in the following sections. See the sample terminal session below for more details.

Sending Messages and Files

While signed on to your MTS account, use the following procedures for sending messages and files.

For messages:

(1) Enter the command:

\$message

(2) Enter:

send to janejonz@cunyvm

- (3) Type your message. Press the ENTER key twice when you are finished.
- (4) Enter:

post

For files:

(1) For general file transfer to a remote system, enter the command:

\$control *export* destination=janejonz@cunyvm

To transfer a file to a remote printer, enter the command:

\$control *print* destination=janejonz@cunyvm

To transfer a file to a remote card punch, enter the command:

\$control *punch* destination=janejonz@cunyvm

(2) Enter:

\$copy myfile *export*

or \$copy myfile *print*
or \$copy myfile *print*
(3) Then:
\$release *export*
or
\$release *print*
or
\$release *punch*

Receiving Messages and Files

While you are signed on to MTS, use the following procedures to receive messages, dispatches, and files:

For messages:

- (1) Tell your colleagues to use the address userabcd@umichum or userabcd@umichub to send you messages.
- (2) You are notified when you sign on if you have new messages waiting for you.
- (3) To read new messages, enter the command:

\$message

and then

retrieve

(4) If you have more than one new message, enter NEXT to see each succeeding message.

For files:

- (1) Tell your colleagues to use the address userabcd@umichum or userabcd@umichub to send you files.
- (2) To check for incoming files, enter the command:

\$sys q incoming

(3) To copy an incoming file to a file in your account, enter:

\$copy *import* mtsfilename

(4) To copy and print an incoming VM print file, enter:

```
$copy *import*@mcc mtsfilename
$copy mitsfilename *print*@mcc
```

SAMPLE TERMINAL SESSION

```
# $sig wabc
# Enter password.
2
# Charging Rate: Terminal, Normal, Univ/Gov't, UM, IBM 3090-600E
# Last signon was at 9:48:15, Tue Jul 25/89
# User WABC signed on at 10:48:38, Wed Jul 26/89
# $message
  This is the MTS $Messagesystem.
  { Introductory information. }
@ set introduction=off autohelp=off
@ send to userwxyz@cunyvm
 Text:
? Dear Jane,
? I am texting the BITNET
? message and file transfer system.
? Do you read me? Look for a file coming your way.
                    John
?
            lete
 Post, edit, display, destroy, help, etc.?
@ edit.
: alter 2 "texting"testing"
: 2 I am testing the BITNET
: return
 Post, edit, display, destroy, help, etc.?
@ post
 Message 110844 has been posted.
# $control *export* destination=userwxyz@cunyvm
# *EXPORT* assigned job number 285239
# *EXPORT* RM285239 held
# $copy datafile *export*
# $release *export*
# *EXPORT* RM285239 (285239) released DELIVERY=CNTR ...
# signoff $
# WABC 10:48:38 to 11:10:08, Wed Jul 27/89
#
    $.50
# $17.49
```

In Jane's session:

```
# $message
@ retrieve
1 message retrieved.
Message: 110844, posted: 10:59am Wed Jul 26/89
To: userID=WXYZ
From: userID=WABC@UMICHUM
Dear Jane,
I am testing the BITNET
message and file transfer system.
Do you read me? Look for a file coming your way.
```

John Next, delete, reply, help, etc.? @ reply Text: Dear John I received you loud and clear and will look for the file. Jane ? Enter a null line when message is complete. Post, edit, display, destroy, help, etc.? @ post Message 110949 has been posted. . . . back to message 110844 from WABC. Next, history, delete, reply, help, etc? @ mts # \$sy q incoming - Job Name Job# Time Date Origin Тур С1 Recs File Name - RSCS5239 81792 11:05 Jul26 USERWABC@UMICHUM Pch A 588 # create bit File "BIT" has been created. # copy *import* bit # signoff \$ # WXYZ 10:48:38 to 11:30:08, Wed Jul 26/89 # \$1.60 # \$15.42

APPENDIX B COMPUTERS AND NETWORK SOFTWARE

The following table lists computers that may attach to BITNET. The attachment is provided by the corresponding network software. Information about the computer, operating system, and network software may be needed when communicating with a node. This information is recorded in the BITNET NAMES data base.

Computer / System	Network Software		
IBM System/370 / MTS	Resource Manager (RM) (University of Michigan and all other MTS universities)		
IBM System/370 / VM	Remote Spooling Communications Subsystem Networking (RSCS)		
IBM System/370 / MVS	Job Entry Subsystem 2 (JES2) Job Entry Subsystem 3 (JES3)		
DEC VAX / VMS	JNET (Joiner Associates) ANL/NJE (Argonne National Laboratory)		
DEC VAX / Unix	UREP (Pennsylvania State University) HOMEBREW, BERKHASP (University of California-Berkeley)		
DEC PDP-11/Unix	UREP, HOMEBREW, BERKHASP		
SUN-2/Unix	UREP		
CDC Cyber / NOS	NJEF, AMF, MTF (Control Data Corporation) TIELINE (Tel Aviv University Computing Center) NRV (Universitaet Hannover)		
Sperry 1100 / OS 1100	RTP/1100 (Sperry) TELCON (Max Planck Gesellschaft, Goettingen)		
Honeywell CP-6	(Carleton University, Canada; Honeywell)		
Prime 750 & 950 / PRIMOS	PRIMENET, HASP (Hebrew University Mount Scopus Computation Center; PR1ME)		
Data General MV 10000 / AOS/VE HASP			

(Rijksuniversiteit Utrecht)

APPENDIX C GLOSSARY

adjacent nodes

Two nodes that have a link between them. Non-adjacent nodes that wish to communicate must call upon other nodes to relay information.

agent

A network user considered without regard to whether that user is a person or a program.

BITNET

The portion of the network in the United States. Also used to refer to the network as a whole. BITNET, NETNORTH, and EARN use the same communications software and protocols and are interconnected. See also "NETNORTH" and "EARN."

card image

Describes a record or file conforming to the machine representation of a standard punched card: 80 bytes long, blank padding. Punch files generally consist of card-image records.

carriage control

A convention whereby the first character of a record denotes a forms positioning action (for print files) or stacker selection action (for punch files) to accompany the data in that record. The data is considered to begin with the second character of a record containing a carriage-control character. See also "logical carriage control" and "machine carriage control."

decoding

A transformation that recovers the original form of a file from the encoded form used for transmission over a computer network. See also "encoding."

DISKDUMP encoding

A reversible transformation for files that permits fixed- and variable-length records of arbitrary size to be encoded as 80-byte records. DISKDUMP files are generated by the VM/SP DISKDUMP command. MTS can also generate files in this format using the ENCODING option. See also "NETDATA encoding."

dispatch

An immediate person-to-person communication that is displayed on the recipient's terminal. If the recipient is not logged on, the dispatch is discarded. Some systems refer to a dispatch as a

message. See also "message."

EARN

European Academic Research Network, the European arm of BITNET. EARN, BITNET, and NETNORTH use the same communications software and protocols and are mutually interconnected. See also "BITNET" and "NETNORTH."

electronic mail

See "message."

encoding

A transformation that produces from the original file a form suitable for transmission over a computer network. See also "decoding."

file access

The process of using a file from within a program. When the file is located on a different host from the program, this is called remote file access. Remote file access differs from file transfer in that the file is not moved in its entirety to the host where it will be used, but rather is obtained piecemeal on a record-by-record basis according to the demands of the program. See also "file transfer."

file transfer

The process of sending an entire file from one host to another. See also "file access."

host

See "host computer."

host computer

A computer that provides BITNET services to its users. It is connected to BITNET. Hosts are also referred to as nodes.

link

A physical connection, typically a leased telephone line, between two network nodes. Information passing between two nodes travels over the link that connects them. See also "node."

logical carriage control

A type of carriage control using letters and digits to represent form positioning and stacker selection actions. For print files, the positioning implied by a logical carriage-control character occurs before the accompanying data is printed.

log-on

The process of establishing a terminal session with a host computer. Also a term for the terminal session itself. See also "sign-on."

machine carriage control

A type of carriage control using a machine-dependent control code to represent form positioning and stacker selection actions. For print files, the positioning implied by a machine carriage-control character occurs after the accompanying data is printed.

mail

See "message."

mailbox

A repository for incoming messages for a single recipient. The mailbox allows a received message to be stored even though the recipient is not logged on.

message

A person-to-person communication composed in memorandum style that is conveyed via BITNET. The message is stored in the recipient's mailbox until it can be viewed. See also "mailbox" and "dispatch."

NETDATA encoding

A reversible transformation for files that permits fixed- and variable-length records of arbitrary size to be encoded as 80-byte records. The encoded form also contains information about how the original file was stored. NETDATA files are generated by the VM/SP SENDFILE and NOTE commands, and by the TSO/E Interactive Data Transmission Facility. MTS can also generate files in this format using the ENCODING option. See also "DISKDUMP encoding."

NETNORTH

The Canadian arm of BITNET. NETNORTH, BITNET, and EARN use the same communications software and protocols and are mutually interconnected. See also "BITNET" and "EARN."

node

The BITNET term for a host computer connected to the network. The node has one or more links that connect it to other network nodes. Each node has a unique node name that forms part of the BITNET address of information destined for that node. See also "link."

node name

The one- to eight-character name assigned to a particular node, or host computer, on BITNET. All node names are unique. See also "userID."

path

The ordered sequence of nodes and links that data passing between two communicating nodes will follow. See also "adjacent nodes."

print file

BITNET transmission file containing print data suitable to be sent directly to a printer. It typically contains printer carriage-control characters (logical or machine) and permits record lengths of 132 bytes.

punch file

BITNET transmission file containing card-image (80-byte) data. Many systems pad short punch-file records with blanks to 80 bytes.

server

A program connected to the network in such a way that it can accept requests in the form of commands from network users and take action according to those requests. Typical actions include searching a data base and transferring a file local to the server back to the originator of the request.

sign-on

The process of establishing a terminal session with a host computer. Also a term for the terminal session itself. This term commonly applies to MTS hosts. See also "log-on."

spool

The collection of spool files maintained by the network software.

spool file

A holding area for data within the network software of a given node. Data is copied from the spool file to the link when that node is sending, and from the link to the spool file when that node is receiving. For file transfer, the use of a spool file isolates the user from the timing of the actual transfer process.

store-and-forward

A network design where a file being sent is received in its entirety at each intermediate node along the path from origin to destination before being relayed to the next node on the path. Following a network failure, transmission can be resumed using the complete copy retained at this node.

transparency

There is no restriction on the bit patterns that may occur in the data to be transferred.

userID

A one- to eight-character identifier for an agent (person or program) at a particular node. All userIDs at a particular node must be unique. The userID, along with the node name, forms the BITNET address of that agent. See also "node name."