NETEX®

Network Executive Software

Bulk File Transfer (BFX™) Utility

for UNIX Operating Systems

Release 3.2

Software Reference Manual

MAN-REF-UNXBFX-02

Revision Record

Revision	Description
01 (09/86)	Manual released. Corresponds to BFX release 1.0.
02 (04/91)	Revision packet (460820-01) released. Added note to explanation of transfer to non-HP systems and clarified maximum block size.
03 (07/92)	Manual updated to correspond to BFX release 2.0.
MAN-REF-HUNXBFX-01 (05/02)	Manual updated to correspond to Network Executive Software release of BFX 3.2.
MAN-REF-HUNXBFX-02 (09/2008)	● Added documentation related to “RPARM [NOCR | NOLF | NOCRLF]” ● Changed document revision to “-02”.

Address comments concerning this manual to:

Network Executive Software
Attn: Publications Department
6420 Sycamore Lane N Suite 300
Maple Grove, MN 55369
USA

Comments may also be submitted over the Internet by addressing them to:

mailto:pubs@netex.com

or at our website:

http://www.netex.com

Always include the complete publication number (e.g. MAN-REF-H321-02) and title of the document with your comments.

Preface

This manual describes the user interface to the Bulk File Transfer (BFX™) utility for supported platforms running UNIX or Linux operating systems. BFX is used in conjunction with the NetEx® family of software products provided by Network Executive Software, Inc.

The first section of this manual is an introduction to BFX. It includes a description of BFX, network configurations which can support BFX, and the programs which compose BFX and how they interact.

The second section presents user control statements and parameters. This section also shows the commands used when running BFX.

The last section describes how to write BFX user modules. The user modules are called from user exits within BFX. These user exits were carefully placed within BFX to allow the user modules to perform code conversion when not provided by hardware or software (Network Executive Software does provide for many types of code conversion), to transfer non-sequential files, or to perform other special processing.

“Appending A. BFX Error Messages” lists BFX error messages.

BFX uses NetEx but the reader does not need to understand NetEx to use the first three sections of this manual and BFX. However, users creating their own user modules are assumed to be familiar with NetEx.

Reference Material

The following manuals contain related information.

Number	**Title and Description**
MAN-REF-HUNXIP-02	*NetEx/IP for UNIX Systems Software Reference Manual*
MAN-CNET-CONF-MGR-01	*"C" Configuration Manager and NetEx Alternate Path Retry (APR) User Guide*
460280	*NCT Loader Software Reference Manual*
460580	*NetEx Application Programmer’s Interface Software Reference Manual*

Other vendor’s manuals are listed below:

From StorageTek:

Number	**Title and Description**
460806	*DX NDPV1/PDPV1 NetEx Provider Coprocessor Customer Reference Manual*
460807	*PDNT3/NDNT3 DX NetEx Coprocessor Customer Software Reference Manual*
460527	*HYPERchannel® Message Formats and Protocol Encapsulation*

Notice to the Reader

Corporation Trademarks and Products

Network Executive Software	**NetEx, BFX, PFX, USER-Access, NESiGate**
Storage Technology Corp.	**StorageTek, STK, Network Systems, HYPERchannel, HYPERbus, NSC, RDS, Link Adapter, DX, DXE**
Hewlett-Packard Corporation	**HP, HP-UX, HP9000**
The Open Group	**UNIX**
Red Hat, Inc.	**Red Hat**
Linus Torvalds	**Linux**

These references are made for informational purposes only.

The diagnostic tools and programs described in this manual are **not** part of the products described.

Notice to the Customer

Installation:

There are no installation instructions in this manual. Installation of various BFX products is explained in each particular version’s Memo-To-Users. E.g., Installation of H801, BFX for the Linux Operating System is detailed in MTU-H801-0x. Installation instructions are for use by experienced Systems Programmers.

Document Conventions

The following notational conventions are used in this document.

Format	Description
`displayed information`	Information displayed on a CRT (or printed) is shown in `this font`.
`user entry`	*`This font`* is used to indicate the information to be entered by the user.
UPPERCASE	The exact form of a keyword that is not case-sensitive or is issued in uppercase.
MIXedcase	The exact form of a keyword that is not case-sensitive or is issued in uppercase, with the minimum spelling shown in uppercase.
bold	The exact form of a keyword that is case-sensitive and all or part of it must be issued in lowercase.
lowercase	A user-supplied name or string.
value	Underlined parameters or options are defaults.
<label>	The label of a key appearing on a keyboard. If "label" is in uppercase, it matches the label on the key (for example: <ENTER>). If "label" is in lowercase, it describes the label on the key (for example: <up-arrow>).
<key1><key2>	Two keys to be pressed simultaneously.
No delimiter	Required keyword/parameter.

Glossary

asynchronous: A class of data transmission service whereby all requests for service contend for a pool of dynamically allocated ring bandwidth and response time.

ASCII: Acronym for American National Standard Code for Information Interchange.

buffer: A contiguous block of memory allocated for temporary storage of information in performing I/O operations. Data is saved in a predetermined format. Data may be written into or read from the buffers.

code conversion: An optional feature in the adapter or host DX interface that dynamically converts the host data from one character set to another. An adapter configured with the code conversion has a special 1K RAM that is used for code conversion. This RAM can be loaded with any type of code (for example, ASCII, EBCDIC, et cetera).

Configuration Manager: A utility that parses a text NCT file into a PAM file.

Coprocessor NETwork EXecutive (CP NetEx): Resides on some types of Processor Interface (PI) boards and uses the processing and storage capacity of the board. This allows minicomputer users to use NetEx with minimal impact on host storage and processing.

Data Exchange Unit (DX unit or DXU): A chassis containing a nucleus processor, multiple customer-selectable interfaces, and/or coprocessors.

DX NetEx: A version of NetEx product specifically designed to operate from a DX unit, driven by software running on a host. DX NetEx resides on the P/NDNTx board.

Fiber Distributed Data Interface (FDDI): An American National Standards Institute (ANSI)-specified standard (X T9.5) for fiber optic links with data rates up to 100 Mbps. The standard specifies: multimode fiber; 50/125, 62.5/125, or 85/125 core-cladding specification; an LED or laser light source; and 2 kilometers for unrepeated data transmission at 40 Mbps.

header: A collection of control information transmitted at the beginning of a message, segment, datagram, packet, or block of data.

host: A data processing system that is connected to the network and with which devices on the network communicate. In the context of Internet Protocol (IP), a host is any addressable node on the network; an IP router has more than one host address.

Internet Protocol (IP): A protocol suite operating within the Internet as defined by the *Requests For Comment* (RFC). This may also refer to the network layer (level 3) of this protocol stack (the layer concerned with routing datagrams from network to network).

ISO: Acronym for International Standards Organization.

link: (1) A joining of any kind of DX networks. (2) The communications facility used to interconnect two trunks/busses on a network.

Network Configuration Table (NCT): An internal data structure that is used by the NetEx configuration manager program to store all the information describing the network.

Network Configuration Table Loader (NCTL): An interactive NetEx application program used for configuring local or remote DX NetEx boards, updating their NetEx configuration parameters and/or Network Control Table. The NCT Loader takes a pamfile created by the Configuration Manager and transfers it to the NetEx Coprocessor through a NetEx connection.

NETwork EXecutive (NetEx): A family of software designed to enable two or more application programs on heterogeneous host systems to communicate. NetEx is tailored to each supported operating system, but can communicate with any other supported NetEx, regardless of operating system.

NetEx can reside on the host, on a processor interface board (obsolete), or in a DX unit. The latter two cases use host-resident drivers as interfaces.

NetEx is a registered trademark of Network Executive Software.

Open Systems Interconnection (OSI): A seven-layer protocol stack defining a model for communications among components (computers, devices, people, and et cetera) of a distributed network. OSI was defined by the ISO.

processor interface (PI): A PI interfaces a minicomputer with an adapter. The PI is a board(s) that contains a microprocessor and memory. The processor interface is generally installed in the host. Some types of PIs contain NetEx.

path: A route that can reach a specific host or group of devices.

TCP/IP: An acronym for Transmission Control Protocol/Internet Protocol. These communication protocols provide the mechanism for inter-network communications, especially on the Internet. The protocols are hardware-independent. They are described and updated through *Requests For Comment* (RFC). IP corresponds to the OSI network layer 3, TCP to layers 4 and 5.

Contents

Figures

Tables

Introduction

Network Executive Software's Bulk File Transfer (BFX™) utility is a software package designed to be used with NetEx® software provided by Network Executive Software. BFX and NetEx provide the capability of rapidly moving large amounts of sequential file data between processors. The processor mainframes may use different operating systems or be of different manufacture, provided they have the proper HYPERchanne1® and NetEx products installed (UNIX BFX requires the corresponding UNIX FD or IP host-based NetEx).

BFX has all the options of a user program to access data base systems, or other sequential files. BFX can be run as a batch job or from a command terminal.

BFX can be used to transfer files between different hosts that may require specialized record or data conversion. BFX allows the use of NetEx or hardware assembly/disassembly and code conversion. For other code conversion, BFX has a set of user exits that allow user modules to process data both before and after transfer over the network. These user modules may take responsibility for accessing data. This makes it possible to copy and convert complex data between different machines, or copy direct access files or databases. "User Modules" on page 23 describes how to create user modules.

Supported Configurations

BFX uses the NetEx/IP communications subsystem and HYPERchannel or IP for its data communications. It uses all the capabilities of these products to move files at multi-megabit speeds over the following types of configurations:

- Local HYPERchannel networks connecting hosts via coaxial cables, fiber optic cables, or both. HYPERchannel speeds allow data transfer rates that are only limited by the data channels, operating system I/O overhead, or the file access methods.
- Multiple HYPERchannel networks connected via StorageTek Link Adapters™ . Private, 44.7 terrestrial links allow data transfer to be limited only by channel and access method speeds. Lower speed terrestrial links can have most of their line capacity used by BFX.
- Multiple HYPERchannel networks connected via StorageTek Satellite Link Subsystems. NetEx software allows high-speed transfer of data over communications satellites.
- Local and wide area IP networks.
- Intra-host processing allows application testing using just the local host computer, exercising the software capabilities of BFX and NetEx. Sending your job to your own NetEx and BFXJS does (this.)

Sample Network Configuration

BFX requires that copies of the Network Executive BFX programs and user-written sets of commands that invoke BFX reside in both the source and destination CPUs, as shown in Figure 1 on page 2. BFX may also be used in a way that allows the user to place all of the commands in one of the hosts (BFX must reside in both).

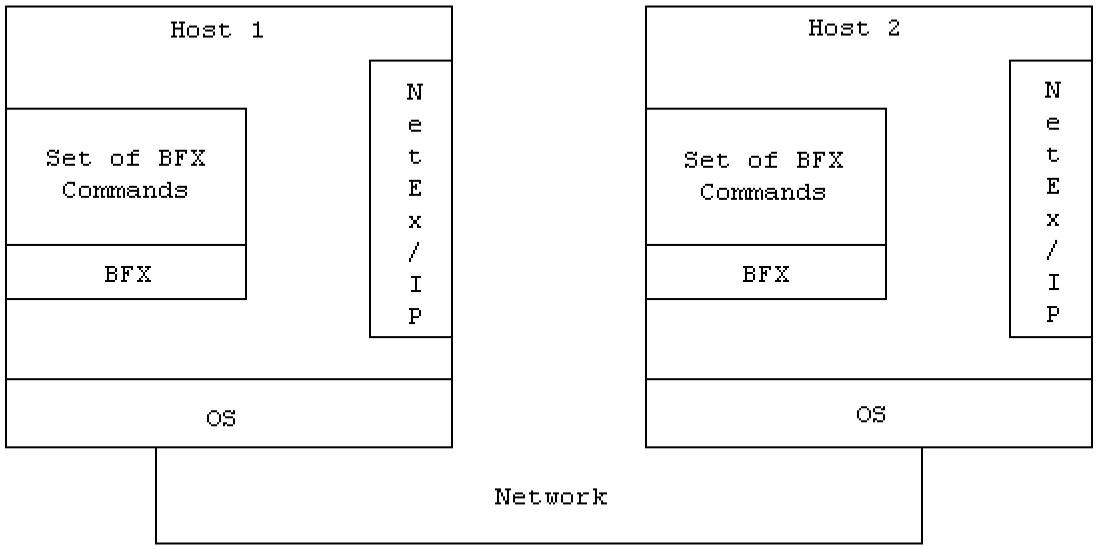

Figure 1. Sample Configuration Using BFX

Figure 1 is a simple example showing the following points:

- BFX and the NetEx/IP must both reside in both hosts. If the two hosts use different operating systems and/or are of different manufacture, data transfer is still supported provided the proper versions of BFX and NetEx/IP are installed and active. BFX uses NetEx/IP and software or hardware code conversion, and allows user modules to perform more elaborate data conversion.
- Since the user directly invokes the BFX utility, (that in turn calls NetEx/IP), the BFX user does not need to learn to use NetEx/IP itself.
- BFX is an application program that can be used like any other file-processing program. Normal users without regard to the other BFX and NetEx applications that may be using NetEx at the same time can directly invoke it.

Three BFX Programs

BFX is a system that consists of three separate programs: BFX Transfer Initiator (BFXTI), BFX Transfer Responder (BFXTR), and BFX Job Submitter (BFXJS).

The BFXTI program runs in the processor that initiates the transfer request. BFXTI processes job and file transfer requests on the initiating machine.

The BFXTR program runs in the processor that responds to the transfer request. BFXTR processes file transfer requests on the responding machine

The BFXJS program also runs in the processor that responds to the transfer request. BFXJS processes jobs transferred to the responding machine, including jobs to be submitted to BFXTR

All three of these programs normally run in each host, making each host capable of initiating or responding to BFX requests.

Using BFX

To us BFX, the programmer must write two sets of commands and parameters. These sets will be called the local command set and the remote command set. The structure and contents of the local and remote sets vary according to the application. There are three basic applications

- Manual job submission
- Automatic job submission
- Remote job submission

The following pages describe the contents of these user procedures for each application, and how BFX processes the user requests.

Manual Job Submission

Manual job submission is the most basic application of BFX. When using manual job submission, the programmer must write a local command set that uses BFXTI, and a remote command set that uses BFXTR. Both the local and remote command sets are written using the control language and BFX commands for the host they will be executed on.

Both the local and remote command sets contain matched sets of SEND and RECEIVE BFX commands. A SEND in one command set must match a RECEIVE in the other command set. The SEND command specifies the file to be written on the receiving host. The SEND and RECEIVE commands may also specify other information about the data being transferred.

To begin the file transfer, an operator on one host submits the local command set and an operator on the other host submits the remote command set. When the jobs are executed, the BFXTR program connects to the BFXTI program (via NetEx/IP and the network) and the files are transferred.

Manual Job Submission Example – Generalized

The following is a manual job submission example that is generalized so that the user can understand the general concept of this application. (Actual command sets include other control language statements excluded from this example.)

Assume that a user on the Local host wants to send file A to the Remote host, and receive file B from the Remote host using manual job submission. The user writes the following local command set:

```
(Create job file, login, and so on.)
$ RUN BFX-ROOT:BFXTI          Invoke BFXTI
SEND    FILE FILEA -          File to send
        TO REMOTE  -          Partner host
        ID BFXJOB  -          Connection ID
        NOSUBMIT              Selects manual job submission
RECEIVE FILE FILEB            File to receive
$ EXIT
```

The ID=BFXJOB parameter uniquely identifies this connection and will also be used in the remote command set. The user also writes the following remote command set:

```
(Create job file, login, and so on.)
$ RUN BFX-ROOT:BFXTR            Invoke BFXTR
RECEIVE FROM LOCAL -            Partner host
        ID BFXJOB  -            Connection ID
        FILE FILEA -            File to receive
SEND    FILE FILEB              File to send
$ EXIT
```

Notice that a SEND in one command set matches a RECEIVE in the other command set.

Figure 2. Manual Job Submission Example illustrates the manual job submission process, using these sample local and remote files.

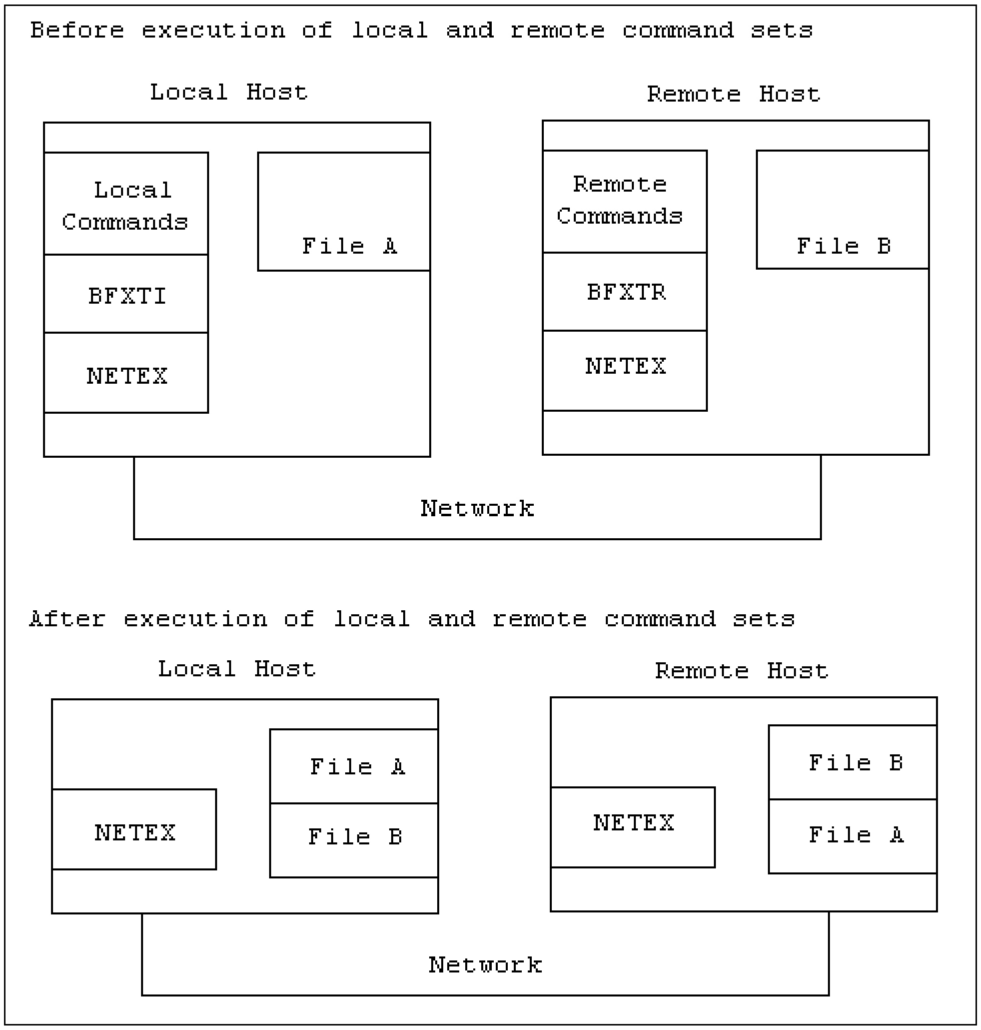

Figure 2. Manual Job Submission Example

Notice in Figure 2 that the Local command set uses BFXTI and the other command set (in this case the Remote command set) uses BFXTR. Execution of the Local and Remote command sets cause BFXTI and BFXTR to exchange the specified files using the network and NetEx/IP.

Manual Job Submission Example – UNIX Specific

The following is a UNIX specific example of manual job submission. This example assumes that the BFX directory is in the user's search path.

```
# mkdir bfxtest
# cd bfxtest
# mkdir ti
# mkdir tr
```

In the `tr` directory, create the file called `trfile` containing the following command:

```
SEND TO localhost ID TEST MSGLVL 0 MODE CHAR FILE XYZ
```

XYZ
This parameter is the name of the file containing data to be sent.

localhost
This parameter is the name of your local host.

In the `ti` directory, create the file called `tifile` containing the following command:

```
RECEIVE FROM localhost ID TEST MSGLVL 0 MODE CHAR FILE XYZ NOSUBMIT
```

localhost
This is the name of your local host.

Now issue the following commands in the `ti` directory:

```
# bfxti tifile &
# cd ../tr
# bfxtr trfile
```

Several BFX messages will be printed on the terminal. (The messages from BFXTI and BFXTR may appear on the same screen if this test is executed from a single terminal.) The last message printed should be:

```
/LCL BFX408I All file transfer have been processed.
```

Verify that the file `../ti/XYZ` has been received correctly by entering:

```
# diff XYZ ../ti/XYZ
```

Automatic Job Submission

Automatic job submission is the most common application of BFX. When using automatic job submission, the programmer must write local and remote command sets, which use commands suitable for each host. The command sets are similar to those used for manual job submission, using SEND and RECEIVE commands.

Instead of submitting the two command sets manually (as in manual job submission), the remote command set is encapsulated in the local command set. The local command set is then submitted for processing by BFXTI on the local host. BFXTI sends the remote command set over the network to the BFXJS program on the remote host. BFXJS then automatically submits this remote command set to BFXTR for execution on the remote host. The local and remote command sets then transfer the specified files as they did for the manual job submission process.

Notice that no operator intervention is required on the remote host.

Automatic Job Submission Example – Generalized

The following is an automatic job submission example that is generalized so that the user can understand the general concept of this application. (Actual command sets include control language statements excluded from this example.)

Assume that a user on the Local host wants to send file A to the Remote host, and receive file B from the Remote host using automatic job submission. The user writes the following command set:

```
(Create remote job file, and so on.)
$ JOB                    Start of remote job
$ PASSWORD               Gain access to remote system
$ RUN BFX_ROOT:BFXTR     Invoke BFXTR
RECEIVE FROM LOCAL -     Partner host
        ID BFXJOB  -     Connection ID
        FILE FILEA -     File to receive
SEND    FILE FILEB       File to send
$ EOJ
(Define end of remote job; then define the local job.)
$ RUN BFX_ROOT:BFXTI     Invoke BFXTI
SEND    FILE FILEA -     File to send
        TO REMOTE  -     Partner host
        ID BFXJOB        Connection ID
RECEIVE FILE FILEB       File to receive
$EXIT
```

Figure 3 Automatic Job Submission, General Example

Notice that the remote command set, the same remote command set that was used in the previous example, is encapsulated in the local job command statements. Appropriate control language statements would replace the housekeeping information in parentheses.

Figure 4. Automatic Job Submission Example on page 7 illustrates the automatic job submission process, using these sample local and remote files.

Figure 4 shows that BFXTI first scans to see if there is a remote job to be submitted. When the remote job is found, BFXTI establishes a NetEx/IP connection with BFXJS on the remote host. BFXTI then transfers the remote job to the BFXJS on the remote host.

After execution of the first part of the local command set, BFXJS submits the Remote job to the BFXTR on the remote host. BFXTR then establishes a NetEx/IP connection with the local host, and the specified files are transferred. While this job is running, BFXJS is read to accept another job.

After execution of the local and the remote command sets, the specified files have been transferred and are ready for use. The BFXTI, BFXTR , and BFXJS programs are ready for the next job.

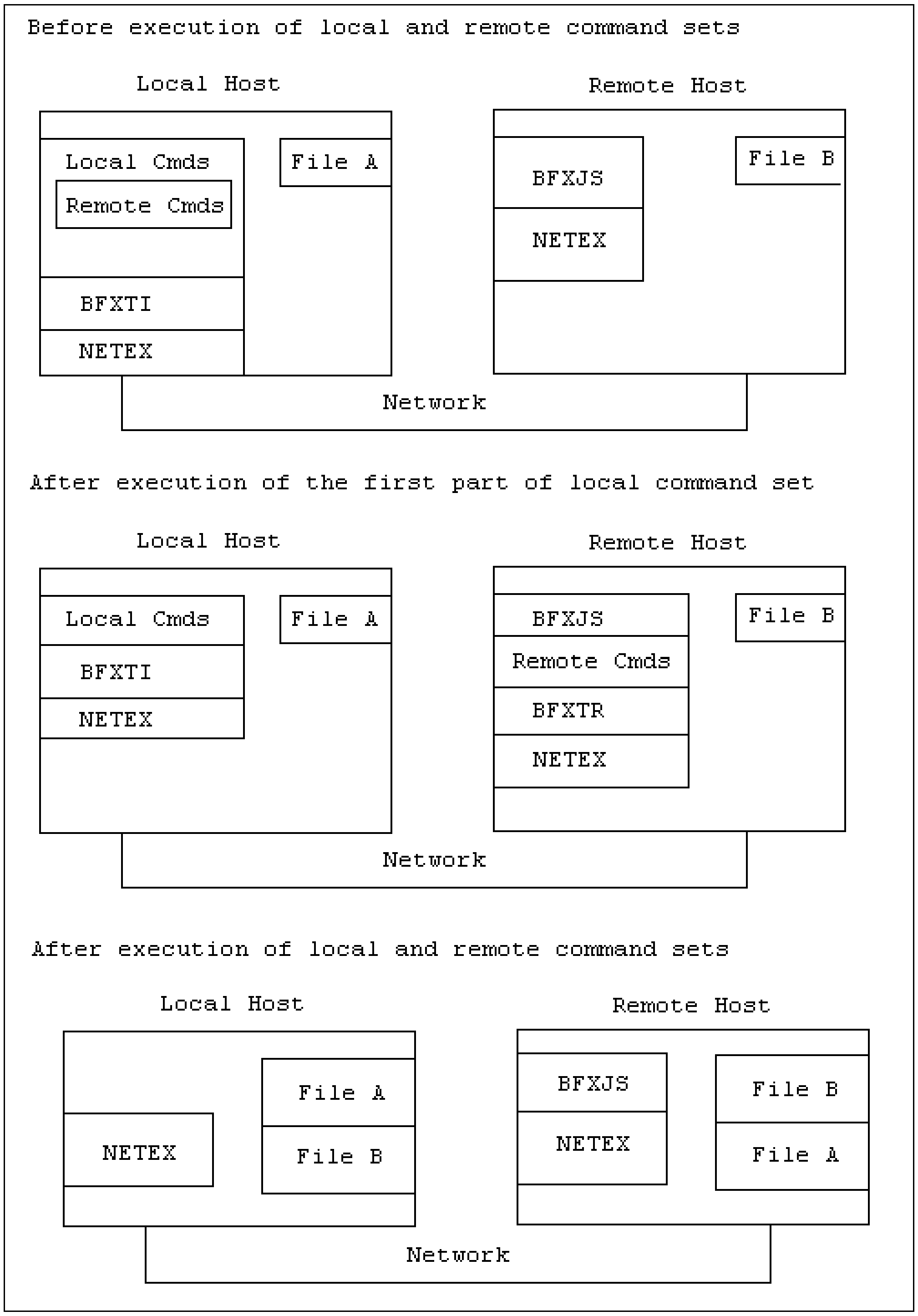

Figure 4. Automatic Job Submission Example

Automatic Job Submission Example – UNIX Specific

The following is a UNIX specific example of an automatic job submission. The file called `tijob` is written using UNIX JCL; if you want to transfer to a different system use that system's JCL in this file.

Create a file called `tijob` containing the following commands:

```
#username
#password
cat >xyz <<EOF
data123456789abcdefghijklmnopqrstuvwxyz
EOF
cat >bfx.in <<EOF
SEND TO hostname FILE xyz ID TEST -
MSGLVL 0 MODE CHAR
EOF
bfxtr bfx.in > bfx.out
```

Create a file called `tijs` containing the following commands:

```
RECEIVE FROM hostname FILE abc ID TEST -
MODE CHAR MSGLVL 0 JOBFILE tijob
```

TO run the example, start `bfxjs` from the `bfx` directory (as super-user) if it is not already running:

```
# su
# $BFXPATH/bfxstart
```

(where `$BFXPATH` is either '/usr/nesi/bfx/bin' or '/opt/bfx/bin' depending the BFX release).

As a regular user, issue the following command:

```
# bfxti tijs
```

Several BFX messages will be printed. The last message displayed should be:

```
/LCL BFX408I All file transfers have been processed.
```

Verify that the file, `abc`, was created in the user's login directory and that its contents are correct.

Remote Job Submission

A special kind of automatic job submission is remote job submission. Using remote job submission, a user can submit a batch job to the remote host (instead of a command set that uses BFXTR). This batch job is then processed on the remote host.

Again, the programmer must write a local and a remote command set. The local command set simply calls BFXTI and issues a SUBMIT command. The remote command set is card-images that are the batch job. This remote command set is encapsulated in the local command set.

The local command set is then submitted on the local host for processing by BFXTI. BFXTI sends the remote job over the network to the BFXJS program on the remote host. BFXJS then submits the remote job on the remote host. (The remote host must be able to process batch jobs.)

Remote Job Submission Example – Generalized

The following is a remote job submission example that is generalized so that the user can understand the general concept of this application. (Actual command sets include control language statements excluded from this example.)

Assume that a user on the local host wants to send job A to the remote host using remote job submission. The user writes the following local command set:

```
(Create remote job file, and so on.)
$ JOB                                Start of remote job
$ PASSWORD                           Gain access to system
$ Control statement 1
$ Control statement 2                Remote job statements
           .
           .
           .
$ Control statement n
$ EOJ                                End of remote job
(Define end of remote job; then define the local job.)
$ RUN BFX_ROOT:BFXTI                 Invoke BFXTI
JOBSUBMIT TO REMOTE -                Partner host
          JOBFILE RJEJOB             Name of job file
$ EXIT
```

Figure 5 Remote Job Submission, General Example

Notice that the remote job is encapsulated in the local job command statements.

Figure 6 below, illustrates the remote job submission process, using these sample local and job files.

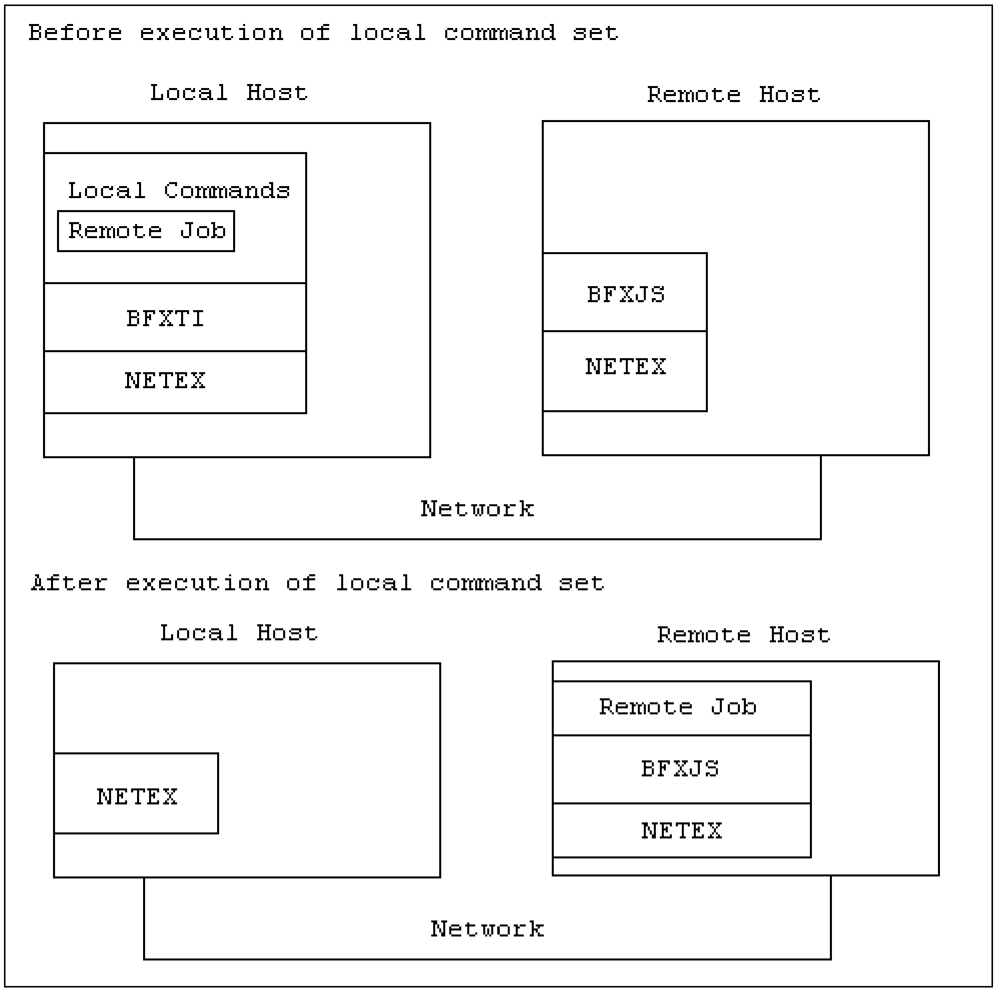

Figure 6. Remote Job Submission Example

Figure 6 shows that BFXTI first scans to see if there is a remote job to be submitted. When the remote job is found, BFXTI establishes a NetEx/IP connection with BFXJS on the remote host. BFXTI then transfers the remote job to the BFXJS on the remote host. BFXTI then terminates. BFXJS submits the remote job for processing on the remote host.

Remote Job Submission Example – UNIX Specific

The following is a UNIX specific example of remote job submission.

Create a file `jstest` containing the following commands:

```
# username
# password
echo hello > hi.there
chmod 666 hi.there
```

Create a file called `tijstest`, where `localhost` is the name of your local host:

```
JOBSUBMIT TO localhost ID BFXJS -
JOBFILE jstest -
MSGLVL 0 MODE CHAR BLOCK 4096
```

As a super-user, start `bfxjs`, if it is not already running:

```
# su
# $BFXPATH/bfxstart
```

(where `$BFXPATH` is either '/usr/nesi/bfx/bin' or '/opt/bfx/bin' depending the BFX release).

As a regular user, issue the following command:

```
# bfxti tijstest
```

Several BFX messages will be printed. The last message displayed should be:

```
/LCL BFX408I All file transfers have been processed.
```

Verify that the file called `hi.there` was created in the user's login directory.

Data Modes

The BFX utility transfers the data on a logical record basis using two selectable types of data modes for host-to-host transfers:

Bit string A verbatim transfer of a continuous string of bits sent from one host and received as a continuous string of bits by the other host.

Character Groups of bits (characters) sent from one host and received as groups of bits (characters) by the other host. The number of bits in a group (bits per character) and characters packed per word depends on the character set used and the word size available to each host. Character mode allows most sequential source or test files to be moved between dissimilar hosts in a meaningful form.

BFX is designed to read a sequential file on the sending host, transfer it to the receiver, and write a sequential file on the receiving host. If the user requires non-sequential operations, encryption, or sophisticated data conversion, the user must furnish a user module to perform the operation. For example, with a user module a non-sequential file could be converted to a sequential file (using standard methods), transferred using BFX, and converted back to non-sequential format.

Security

Because BFX is an ordinary batch job to the host, no security mechanisms are required or supplied by BFX. Security mechanisms are not required because the existing host restrictions on utilization, validation, and accounting will remain in effect.

File Size

BFX was intended to be used on files which are greater than 1 megabyte. The transfer of these large files is highly efficient using BFX. Transferring smaller files is also possible (but not as efficient because elapsed time and batch job queue overhead is higher). Note, however, that transferring multiple small files in one execution of BFX gives an effect similar to a single large file transfer. Therefore, BFX may be used to satisfy most data transfer needs.

Summary

BFX is a system that consists of three separate programs:

- An executable image, BFXTI (BFX Transfer Initiate), that is started in the processor that wishes to either send or receive a file. Input to BFXTI consists of:

  A set of parameters that describe the direction of transfer, the process with which transfer will take place, and which user exits (if any) will be used during the transfer process.

  A logical filename assignment statement specifying a file to be used for either input of output, depending on the direction of file transfer.

  A logical filename assignment statement specifying a file that contains a complete job to be submitted on the partner processor, including all job options, accounting cards, control statements, and the like. This job will invoke the next component, BFXTR, of the BFX system.

- A program that is started in the partner processor, BFXTR (BFX Transfer Responder). This program is invoked by the job provided by BFXTI in the initiating processor. Its input contains much the same information as BFXTI's, namely:

  A set of parameters describing the direction of transfer, the node that initiated the transfer process, and a unique ID of the BFXTI program.

  The equivalent of a logical filename assignment statement specifying a file to be used for either input or output, depending on the direction of data transfer.

- The last component, BFXJS, is a resident program that must be present in the non-initiating CPU when BFX is to be run. Its sole function is to accept the job destined for the responding host that was provided to the BFXTI program. BFXTI and BFXJS user NetEx/IP to transfer the job, and BFXJS submits the job, unchanged, to the internal reader of the responding machine. It should also be noted that BFXJS can be used independently of the rest of BFX to provide a simple, high speed Remote Job Entry facility.

BFX is principally a batch utility for the transfer of sequential files between processors. There are three basic ways to use BFX:

Automatic Job Submission

The initiating party sends a card image file to the BFXJS job submission program. The BFXJS program submits the file to the batch internal reader. The batch process, once started, invokes BFXTR to connect back to the original initiating party. At that stage a file or series of files can be transferred between the initiator and the batch program. This technique is well suited for the transfer of very large files or for the delivery of a file to the initiator.

Remote Job Submission

The initiating party submits a card image file to the BFXJS utility. The associated DCL statements build a file on the remote processor based on the card image data in the job. This is probably the simplest way to “send” a file to another machine.

Manual Job Submission

Two programmers (or operators) on two hosts agree to send a specific series of files in a manually coordinated way. The first party invokes the BFXTI program without submitting a job to BFXJS; the second party then invokes BFXTR to connect to the first party and transfer files.

Commands and Parameters

This section describes the format and use of commands that execute the BFX programs.

Rules for Coding Commands

Commands are read as a sequence of tokens. Tokens consist of any sequence of characters, delimited by the beginning and end of lines, spaces equal signs, or commas. Tokens are either considered command tokens or parameter tokens. Command tokens may be at most twelve characters long and must match the name of some BFX command. Command tokens and keyword parameter tokens can be abbreviated as long as they remain unique. The legitimate values of parameter tokens depend on what command they are parameters for (as described in this section).

To describe a transfer, the programmer must use several commands. A collection of commands used to describe a transfer are said to make up a logical line. A logical line is a sequence of lines where all but the last line have the command token "-". Any data after the command token "-" in a line is ignored, since logically the next line is attached at that point.

For example:

```
SEND FILE SAMPLE TO VM4341 ID BFXJOB
```

and

```
SEND FILE = SAMPLE -
       TO = VM4341 -
       ID = BFXJOB
```

Both of these logical lines contain four commands used to describe this transfer.

In keywords, all characters are treated as uppercase. In value fields, no translation is performed. Therefore '`HOST HOST1`' and '`host HOST1`' are equivalent, but '`HOST HOST1`' and '`HOST host1`' are not.

General Information About BFX Commands

The commands accepted by the BFX programs are shown in Figure 7.

```
| SEND      |
| RECEIVE   |
| JOBSUBMIT |
[ BLOCK = size of network block during data file transfer ]
[ BMOD = module to send/receive blocks ]
[ BPARM = parameter string for Block Module ]
[ DELAYTIME = delay time between SCONN's ]
[ FILE = logical (or actual) name of data file ]
[ | FROM | = remote host name ]
  | TO   |
[ ID = BFX application name ]
[ JBLOCK = size of network block during job file transfer ]
[ JID = offered name of BFXJS job ]
[ JMOD = module to handle remote job submission ]
[ JOBFILE = logical (or actual) name of job file ]
[ JPARM = parameter string for Job Module ]
[ JRMAXL = maximum record length for the transfer of a job file ]
[ MODE = character set ]
[ MSGLVL = minimum severity of messages to be logged ]
[ NEWHOST = new remote host name ]
[ NOSUBMIT ]
[ RMAXL = maximum record length for the file transfer ]
[ RMOD = module to read/write file records ]
[ RPARM = parameter string for Record Module ]
[ | SOE   | ]
  | NOSOE |
[ TIMEOFFER = timeout value for SOFFR ]
[ TIMEOUT = timeout value for SREAD ]
[ | TIMESTAMP   | ]
  | NOTIMESTAMP |
```

Figure 7. BFX Control Statements

The order in which commands are specified is not important, with the following exceptions:

- If the same parameter appears more than once between two transfer commands, the last specification is used. The same is true for contradictory commands (example: TIMESTAMP/NOTIMESTAMP).
- If a NOSUBMIT command follows a NEWHOST command, a job file will not be sent if the next transfer statement is SEND or RECEIVE; if NOSUBMIT precedes NEWHOST, however, a job file is sent.

BFX Command Defaults

Some commands may only be issued to certain BFX programs, such as BFXTI. Table 1 on page 15 summarizes where each command may be issued. Most of the parameters take default values that are in effect at the start of execution of the BFX program (also shown in Table 1). Specifying a new value for such a parameter modifies its value for the duration of the run (or until another new value is specified). A parameter will revert to its default value if the parameter name is specified in a command and no value follows the name.

Default values are established and may be changed at installation time in procedure IniTrol of module BFXUIM.

Table 1 BFX Command Defaults

Command	**Default**	**Issued From**
`SEND`	None	BFXTI, BFXTR
`RECEIVE`	None	BFXTI, BFXTR
`JOBSUBMIT`	NONE	BFXTI
`BLOCK`	32767	BFXTI, BFXTR
`BMOD`	STANDARD	BFXTI, BFXTR, BFXJS
`BPARM`	(blanks)	BFXTI, BFXTR, BFXJS
`DELAYTIME`	5	BFXTR
`FILE`	BFXFILE	BFXTI, BFXTR
`FROM`	LOOPBAK	BFXTI, BFXTR
`TO`	LOOPBAK	BFXTI, BFXTR
`ID`	"????????"	BFXTI, BFXJS
`JBLOCK`	32767	BFXTI, BFXTR
`JID`	BFXJS	BFXTI, BFXJS
`JMOD`	JOBSTANDARD	BFXTI
`JOBFILE`	RMTJOB JOBxxxxxxx	BFXTI BFXJS
`JPARM`	(blanks)	BFXTI
`JRMAXL`	80	BFXTI, BFXJS
`MODE`	CHARACTER	BFXTI, BFXTR
`MSGLVL`	4	BFXTI, BFXTR, BFXJS
`NEWHOST`	None	BFXTI
`NOSUBMIT`	None	BFXTI
`RMAXL`	1024	BFXTI, BFXTR, BFXJS
`RMOD`	STANDARD	BFXTI, BFXTR, BFXJS
`RPARM`	(Blanks)	BFXTI, BFXTR, BFXJS
`SOE`	NOSOE	BFXTI, BFXTR
`NOSOE`	NOSOE	BFXTI, BFXTR
`TIMEOFFER`	240	BFXTI, BFXJS
`TIMEOUT`	60	BFXTI, BFXTR, BFXJS
`TIMESTAMP`	NOTIMESTAMP	BFXTI, BFXTR, BFXJS
`NOTIMESTAMP`	NOTIMESTAMP	BFXTI, BFXTR, BFXJS

Command Descriptions

The following paragraphs describe the BFX commands. The BFX commands were summarized in Table 1 on page 15.

SEND Command

This command specifies that this logical line describes the transmission of a file to a remote host. It is legal in BFXTI and BFXTR. SEND takes no parameters.

RECEIVE Command

This command specifies that this logical line describes the receipt of a file from a remote host. It is legal in BFXTI and BFXTR. RECEIVE takes no parameters.

JOBSUBMIT Command

This command forces a job to be submitted to the remote host, even if this is an exchange with the same host as the previous logical command line. This command also indicates that no file is to be sent. This command is legal only in BFXTI and BFXJS. JOBSUBMIT takes no parameters.

BLOCK Command

This command is used to specify the maximum size (in addressable units) of the buffers of data to be sent through NetEx/IP to/from the other host during data file transfer. The block size must be at least large enough to accommodate the largest logical record in the file to be sent/received.

The maximum size allowed is normally 32K addressable units, but may be lowered when the BFX programs are built. If specified, it should be specified to both BFXTI and BFXTR; if the values are not equal (or one is allowed to default) then the smaller of the two sizes implicitly or explicitly specified will be used. The BLOCK value can not be greater than the NetEx/IP values of MAXBLKIN and MAXBLKOUT. This command is legal in BFXTI and BFXTR.

For more information on the use of the BLOCK parameter see “Special Considerations” on page 21.

BMOD Command

The BMOD command names the user exit block module to use in building data blocks for transmission or disassembling received data blocks. To be accessed, references to the Block Module must have been previously added to the Network Executive Software-supplied module BFXUIM, and the BFX programs rebuilt.

The BMOD command takes an unabbreviated command token, which is looked up in the list of user exits, as a parameter. If this parameter is not specified, a default Block Module is used. Normally, this will be the Network Executive Software-supplied module, which is designed to call a Record Module to obtain or store file information, and to provide or decode the protocol information needed to block and deblock records flowing over the network.

See “User Modules” on page 23 for more information on user-supplied Block and Record modules.

The Block Module name is an alphanumeric string one to eight characters in length. BMOD is valid for all BFX components.

BPARM Command

This command specifies a string of parameter information that will be passed to the specified Block Module for processing. The parameter is a token (or quoted string) of up to 64 characters. User-written Block Modules may use this string to control special processing options. The default NESi-supplied Block Module ignores the BPARM string.

This command is valid in any BFX component.

DELAYTIME Command

This command specifies how long the BFX component will delay between reconnects in attempting to reach the remote host. Although legal in all BFX components, it is only effective in BFXTR. This command takes one numeric parameter.

FILE Command

This command specifies the name of the file to send or receive. This is legal for BFXTI and BFXTR. The FILE command accepts one parameter that is a file name token and is limited to 80 characters in length.

FROM and TO Command

These commands specify which hosts will be exchanging file. FROM and TO are logically equivalent. The two forms are provided for readability only. FROM and TO are legal in BFXTI and BFXTR. FROM and TO take one parameter, which is the 1- to 8-character NetEx/IP hostname.

ID Command

This command specifies a unique name by which the initiator (BFXTI) and responder (BFXTR) will identify themselves to each other. The ID command is required for exchanging files. Only one BFX program with this ID should be present in either the local or the remote host. The ID parameters for BFXTI and BFXTR must be the same, or the file transfer will fail.

ID is legal on BFXTI and BFXTR. This command takes one parameter, the first eight letters of which are significant. It can be up to the length of a token, however subsequent letters are ignored.

JBLOCK Command

This command specifies the block size to use in exchanging the job file with the remote host. The JBLOCK command is legal in BFXTI and BFXJS. This command takes one numeric parameter. The maximum size allowed is normally 32KB, but may be lowered when the BFX programs are built (refer to "User Modules" on page 23).

JID Command

This command specifies the id of the BFX component to exchange job file with. This command is legal in BFXTI and BFXJS. The parameter is an application name, like the ID command parameter.

JMOD Command

This command specifies the name of the user exit module (the Job Submission Module) that will take responsibility for the submission of the job file to BFXJS on the remote host. To be accessed, references to the Job

Submission Module must have been previously added to the NESi-supplied module BFXUIM, and the BFX programs rebuilt.

If this parameter is not specified, a default Job Submission Module is used. Normally, this will be the NESi-supplied module, which uses the NESi-supplied Block and Record Modules to transfer character files between all types of host processors.

See “User Modules” on page 23 for more information on user-supplied Job Submission modules.

The parameter is an unabbreviated command token which is looked up in the list of user exit job modules. JMOD is valid only in BFXTI.

JOBFILE Command

This command specifies the name of the jobfile to send to the remote host. This file is sent when switching to a new host or the “JOBSUBMIT” command is given. This command is only legal in BFXTI and BFXJS. The JOBFILE command accepts one parameter that is a file name token which is limited to 80 characters in length.

JPARM Command

This command specifies a string of parameter information that will be passed to the specified Job Submission Module for processing. User-written Job Submission Modules may use this string to control special procession options. The default NESi-supplied Job Submission Module ignores the JPARM string.

The parameter is a token (or quoted string) of up to 64 characters. JPARM is legal only in BFXTI.

JRMAXL Command

This command specifies the maximum record length for the transfer of a job file. The parameter specified with the JRMAXL command is a numeric token.

The JRMAXL command is legal in BFXTI and BFXJS.

MODE Command

This command specifies the character set for the file exchange. The parameter specified with the MODE command is a command token. The value BIT is used to indicate that binary transfer is desired. CHARACTER would select the default character set. In multi-character set implementations, the names of the character sets would also be legal values.

If the remote host is the same type of processor, either transfer mode will usually result in exactly the same data as was sent being written on the receiving host. If the native character set of the remote processor is different, however, the processing is quite different. In CHARACTER mode, the text will be converted from the character set of the sending machine to the receiving machine, and the logical records of character information will be converted to the logical record structure of the receiving host. In BIT mode, the exact pattern of bits in each logical record will be sent to and stored by the receiving host as a binary logical record, presumably to be converted to a useful form at a later time.

Both sides must specify the same data transfer mode. If the two specifications are inconsistent, an error will occur.

The MODE command is legal in BFXTI and BFXTR.

MSGLVL Command

This command specifies which BFX messages will be displayed. The parameter must be specified as a decimal number in the range 0-16. Messages with severity levels greater than or equal to the specified value are written to OUTPUT. Message level 16 is not used, so setting the message level to 16 will inhibit all error messages. The meaning of the various message levels is shown below:

0-3	Only messages that are generated for diagnostic purposes are displayed. These messages will trace the flow of events in BFX in detail, and are intended only for use in diagnosing problem with BFX, NetEx/IP, or newly written user modules.
4-7	The following types of messages will be displayed: status of job submission, the start of the remote BFXTR job if applicable, and statistics on the file transferred.
8-11	If transfer of the file is normal, only a "`transfer complete`" will be generated. If the transfer of the file is not normal (for example, if an error that causes the run to finish is encountered), then the error messages will be logged.
12-15	Only errors that cause the file transfer process to be aborted will be printed.

This command can be issued in any BFX component.

NEWHOST Command

This command specifies a new host to exchange files with after one or more transactions have been performed with another host or hosts. It is effectively equivalent to the "TO" and "FROM" commands. NEWHOST takes one parameter which is the 1 to 8 character NetEx/IP hostname. This command is legal in BFXTI component.

NOSUBMIT Command

NOSUBMIT is used in cases where each half of the BFX pair will be started independently on the two hosts. In such cases, the BFXTI job with the NOSUBMIT statement should be started first; the BFXTR job should be started after the BFXTI is offered. If this command is specified, BFXTI will not send a batch job through to BFXJS on that host.

RMAXL Command

This specifies the maximum record length for the file transfer. The parameter for this is a numeric token. This command is legal in all BFX components.

RMOD Command

The RMOD command specifies the name of the user exit record module that will provide or accept the logical records of the file. To be accessed, references to the Record Module must have been previously added to the NESi-supplied module BFXUIM, and the BFX programs rebuilt.

If this command is not specified, a default Record Module is used. Normally, this will be the NESi-supplied module, which is designed to transfer character files between all types of host processors, and to move binary or structured information to another host without change on a logical record basis.

See "User Modules" on page 23 for more information on user-supplied Block and Record modules.

The Record Module name is a parameter like a command token except that abbreviation is not legal. The name is looked up in the list of user exit record modules. RMOD is valid in all BFX components.

RPARM Command

This command specifies a string of parameter information that will be passed to the specified Record Module for processing. User-written Record Modules may use this string to control special processing options.

The parameter is a token (or quoted string) of up to 64 characters. RPARM is legal in all BFX components.

An RPARM parameter has been added to allow the user to tailor how carriage returns (CR) and line feeds (LF) are handled on the receiving side of a file transfer. This parameter is currently only available in the Linux BFX product, H801, and only when H801 is on the **receiving** side of the file transfer.

BFX's default behavior is to ignore a (CR), if present, at the end of the received record and terminate the re-cord with an (LF). This means that if a (CR) is present, the (CR) will be written to the file, and since the re-cord will be terminated with an (LF), the record will effectively be terminated with a (CR)(LF). If a (CR) is not present, the record will be written to the file, and terminated with an (LF). This parameter is valid only when Linux BFX (H801) is the receiving side of the transfer. It will be ignored on the transmit-ting side of the transfer.

The format of this parameter is as follows and must be specified in all lower-case or all upper-case characters:

```
RPARM [ NOCR | NOLF | NOCRLF ]
```

where:

NOCR	This option will remove one carriage return (CR) from the end of the incoming re-cord (if present) and terminate it with a line feed (LF) character.
NOLF	This option will ignore a carriage return (CR) at the end of the incoming record (if present). It will not terminate it with a line feed (LF) character.
NOCRLF	This option will remove one carriage return (CR) from the end of the incoming re-cord (if present) and will not terminate it with a line feed (LF) character.

Example 1:

```
receive from ahost id alpha file file-name mode char -

msglvl 0 timestamp rparm NOCR
```

This will strip one (CR) from the end of each incoming record, if it has one, and terminate the record with an (LF).

Example 2:

```
receive from ahost id alpha file file-name mode char -

msglvl 0 timestamp rparm nolf
```

This will leave a (CR) intact, if present, and NOT terminate the record with an (LF).

SOE Command

The Stop On Error (SOE) command causes BFX to stop reading further commands if any one transmission has problems. This command is legal in BFXTI and BFXTR. It takes no parameters.

NOSOE Command

The NO Stop On Error (NOSOE) command reverses the effects of a previous "SOE". It causes BFX to keep reading logical commands even if one transfer fails. This command is legal in BFXTI and BFXTR. It takes no parameters.

TIMEOFFER Command

The TIMEOFFER command specifies the maximum amount of time (in seconds) that BFXTI will wait for the BFXTR job to be initiated in the remote host. BFXTI "offers" itself through NetEx/IP to the remote job. If the remote job does not respond within the time allotted by TIMEOFFER, BFXTI will abort the transfer process.

The usage of the TIMEOFFER command is affected by the specification of the RMTJOB parameter. For more information, see "Special Considerations" later in this section.

This parameter can be defaulted. It is legal in all BFX components, although it currently only has effect in BFXTI and BFXJS.

TIMEOUT Command

The TIMEOUT command specifies the maximum amount of time (in seconds) that the BFX program should wait for a read through NetEx/IP to complete.

This command should only be specified when sending data over low speed links (such as phone lines).

It should also be used by the receiving BFX when there is a possibility that the sending BFX will be experiencing long delays during file transfer. For example, if a multi-volume tape file is being sent, the sending BFX will be delayed when one reel ends and the next reel is being mounted. In such cases, TIMEOUT should be set to a very high value or to 0 (timeout disabled).

TIMESTAMP Command

The TIMESTAMP command specifies that a timestamp is to be printed with all subsequent BFX messages. The timestamp will give the time of day that the message was sent to the print file in the format hh:mm:ss on the left part of the message.

This command is legal in all BFX components and takes no parameters.

NOTIMESTAMP Command

The NOTIMESTAMP command reverses the effect of the TIMESTAMP command. NOTIMESTAMP specifies that no timestamp is to be printed with all subsequent BFX messages.

This command is legal in all BFX components and takes no parameters.

Special Considerations

Transfer to Non-UNIX Computer Systems

The code supplied by NESi is designed to support transfer of file data between "incompatible" computers in two ways (selected by the MODE command):

- Files containing only character information (program source files, text, line printer output) will be converted to an immediately useful form when sent to a different processor.
- Files containing binary information, floating point numbers, or data structures will be sent to the other computer as a continuous string of bits on a logical record basis. Depending on the type of computer systems involved, the data may be ready for direct use, or some processing of the data may be needed following the BFX run before it is ready for direct use.

BFX does not convert tab characters to blanks and vice versa. If files are sent to a system that does not use the tab character to produce “white space” on a listing, then the tabs must be removed before or after the file is sent.

Note: Consult the documentation of the other system for necessary information to transfer to non-UNIX computer systems.

Use of the BLOCK Command

The value specified for the BLOCK command (or the default value) is not necessarily the block size that is used during file transfer. The minimum of the block sizes requested by the partners in a transfer is the size actually used. However, the size requested by the BFX program can be greater than was specified by the user’s BLOCK command.

The BLOCK command is overridden internally by BFX if the specified block parameter is less than the minimum required.

It is important to note that each record sent between the BFX programs has overhead associated with it. This overhead will be a minimum of 6 bytes for bit mode transfers or 8 bytes for character mode transfer; however, it can be higher when transferring between machines with unusual word sizes. The BFX programs do not know exactly how long the overhead will be when they present their desired block sizes. Therefore, BFX assumes a worst case of 21 bytes when figuring overhead into its requested block size. So, the longest record length that can be transferred using BFX is 32 KB.

User Modules

BFX allows considerable flexibility for the incorporation of user-written code to read and write file data and to process jobs before delivery to the remote host. This section describes in detail the entry and exit conventions of these modules, and the procedures for installing these modules into the BFX subsystem.

Writing User Modules

This section describes the conventions that must be followed by user-written modules. Record modules are discussed first, followed by block modules and job submission modules.

User-written modules are called as procedures. BFX supports user code that is written in any language that adheres to the calling standard.

Record Modules

The user may substitute or add his own Record Processor module(s) to BFX. The requirements are as described for BFXSRM, BFXRRM, the standard sending and receiving record modules. The name(s) of the modules must be added to the selection code in BFXUIM.

The Record Module is designed to be entered at a single entry point. The first time the module is called, it is expected to open a file for input or output. On subsequent calls, the Record Module will either provide or accept a logical record.

It should be noted that the conventions used here are those enforced by the standard NESi Block Modules. If a user writes their own Block Modules, those Block Modules may not call Record Modules at all, or may call a “Record Module” using completely different conventions as determined by the user.

Record Modules are normally written to be either Sending or Receiving Record Modules. The NESi sending and receiving record modules are described in the following paragraphs. The parameter list passed to both types of modules is identical and is described under “Figure 8 Record Module Parameters” on page 25. The usage of the parameters differs between module type and is discussed in detail for each parameter.

BFXRRM – NESi Receiving Record Module

This is the standard RMOD on the receiving side. This module processes each record as delivered to it by BFXRBM and writes it out to the file. On the first call it opens the file; if eof or abort is set, it closes the file.

For users writing their own record modules, the calling sequence must match if called from standard block modules.

```
BFXUIM (Buff,Rlen,Rubit,Rlev,Msg,MsgLen,MsgLev,Tfile)
Buffptr Buff;        /* Buffer to extract from */
Integer *Rlen,       /* Length of data in Buff */
        *Rubit,      /* Unused bits in Buff */
        *Rlev,       /* End level of Buff */
        *MsgLen,     /* Length of data in Buff */
        *MsgLev,     /* Significance of Msg */
Char    *Msg;        /* RMod error message */
FileSendPtr Tfile;   /* File being moved */
```

The logic of the NESi module is as follows:

- If Rlev < 0, then initialize counters and call Ofile to open the file. If there was an error on the open, set Rlev = 16, form an error message, and exit.
- If Rlev is nonzero, then
  - If Rlev >= 15, then set EOFlag.
  - If Rlen = 0 (zero length record), set good status.
  - If Rlen <> 0, then call WriteData to write the record to the file, and increment the record count.
  - If there was an error, set Rlev to 16, form an error message and call Cfile to close the file.
  - If there was no error and Rlev > = 15, then call Cfile to close the file (abort or eof). Create the appropriate message for abort or eof condition.

BFXSRM – NESi Sending Record Module

This is the standard RMOD on the sending side. This module reads each record from the file as requested by BFXSBM, and writes it out to the file. On the first call it opens the file; if eof or abort is set, it closes the file.

For users writing their own record modules, the calling sequence must match if called from standard block modules.

```
BFXSRM (Buff,Rlen,Rubit,Rlev,Msg,MsgLen,MsgLev,Tfile)
Buffptr Buff;         /* Buffer to extract from */
Integer *Rlen,        /* Length of data in Buff */
        *Rubit,       /* Unused bits in Buff */
        *Rlev,        /* End level of Buff */
        *MsgLen,      /* Length of data in Buff */
        *MsgLev,      /* Significance of Msg */
Char    *Msg;         /* RMod error message */
FileSendPtr Tfile;    /* File being moved */
```

The logic of this module is as follows:

- If Rlev < 0, then initialize counters and call Ofile to open the file. If there was an error on the open, set Rlev = 16, form an error message, and exit.
- If Rlev = 16, then call Cfile to close the file and form an abort message.
- If Rlev = 1..15, then
  1. If this is the first record, form a start message.
  2. Call Read Data to get a record from the file.
  3. If there was an error, set Rlev to 16, form an error message and call Cfile to close the file.
  4. If there was no error and Rlev = 15 (eof), then call Cfile to close the file (abort or eof). Create the appropriate message for abort or eof condition.
  5. If there was no error and Rlev <> 15, then set Rlev = 1 (normal record) and increment record count.

Record Module Parameters

The parameter list to the Record Module is illustrated using a sample C language sequence in Figure 8.

```
RMOD (Buff,Rlen,Rubit,Rlev,Msg,MsgLen,MsgLev,Tfile)
Buffptr Buff;         /* Buffer to extract from */
Integer *Rlen,        /* Length of data in Buff */
```

```
        *Rubit,      /* Unused bits in Buff */
        *Rlev,       /* End level of Buff */
        *MsgLen,     /* Length of data in Buff */
        *MsgLev,     /* Significance of Msg */
Char    *Msg;        /* RMod error message */
FileSendPtr Tfile;   /* File being moved */
```

Figure 8 Record Module Parameters

Buff Record Module Parameter

For a Receiving Record Module, Buff will contain a logical record for the Record Module to handle.

A Sending Record Module should obtain a logical record and put it in Buff.

Rlen Record Module Parameter

A Receiving Record Module will be given the length of the logical record (Buff) in Rlen.

A Sending Record Module should return the length of the outgoing record in Rlen.

Rlen is given in characters when MODE > 0, in bits when MODE = 0.

When the Record Module is called for the first time, Rlen will contain the BLOCK parameter specified by the user (or defaulted). The Record Module should return in Rlen the maximum logical record length (in bytes) that it expects to process.

Rubit Record Module Parameter

Rubit is the NetEx Unused Bit Count as described in the NetEx manuals and specifications. This value is principally used for sending precise amounts of data to machines whose word sizes are not a multiple of eight bits.

A Sending Record Module will be passed a zero in Rubit. If the Record Module wishes to use the Unused Bit Count facility, it should place a nonzero value in the field. NetEx will use the datamode and total number of useful bits sent and will provide a resultant length and unused bit count to be delivered to the receiving Record Module. The number of bits in the last addressable unit that is not meaningful information.

Rlev Record Module Parameters

A Receiving Record Module will be given the “delimiter level” of the logical record in Rlev. Normally, this is a binary value in the range 1 to 15. The value 1 is used to indicate the normal end of a logical record. A value of 15 indicates that this record is to be the last record received, called End of Information in the BFX specifications. Intermediate values are used by computer operating systems that have a “hierarchy” of end delimiters in their file structure, such as the EOR, EOF, and EOI indications in CDC CYBER data files.

The Record Module will not be called again after being called with Rlev = 15; therefore, it must do any termination processing required before returning.

A Receiving Record Module should expect to handle zero-length logical records (Rlen = 0). When the module is called with Rlen = 0 and Rlev = 15, however, it should interpret this to mean that the previous call had the last logical record to be received and that the current call is merely for termination purposes.

A Sending Record Module will normally be called with Rlev = 0 and will return the “delimiter level” of the outgoing logical record. The value 1 is used to indicate the normal end of a logical record. A value of 15 indicates that the returned record is to be the last record sent. Intermediate values may be used; no distinction is made between the values 1 through 14.

Rlev is also used to signal when the Record Module is entered for the first time. On that first entry, Rlev will have a value of –1, it should perform any required initialization before returning.

A value of 16 in Rlev is used to indicate that a failure has occurred, and the transfer is being aborted. When a Record Module is called with Rlev = 16, it should perform any termination processing required. If a Record Module finds a non-recoverable error internally, it should perform termination processing and return with Rlev = 16.

Msg Record Module Parameter

Msg is an area that allows alphanumeric information to be returned to the calling Block Module and then to the main body of BFX. If the Record Module wishes to return a message, it should place a string of native character information in Msg and update Msglen and Msglev. It is good form for a Record Module to return a message when EOI (Rlev = 15) or abort (Rlev = 16) is indicated. The maximum length of the message is 128 characters.

Msglen Record Module Parameter

Msglen indicates the length of the message returned by the Record Module. Msglen is passed to the Record Module as zero; if Msglen is zero on return, assume that no message is present in MSG.

Msglev Record Module Parameter

Msglev indicates the importance of the message returned to the calling Block Module. This should be specified as a binary value between 0 and 15. If the value of Msglev is great than or equal to the value of MSGLEVEL specified by the user (or defaulted), the message will be printed on the BFX program's OUTPUT log and sent to the remote BFX program (if a connection is intact).

Tfile Record Module Parameter

Tfile contains the twelve-character string specified in a tfile parameter statement (or defaulted). This string may be used by the Record Module in any desired fashion.

Block Modules

The user may substitute or add his own Block Processor module(s) to BFX. The requirements are as described for BFXSBM and BFXRBM, the standard sending and receiving block modules. The name(s) of the modules must be added to the selection code in BFXUIM.

Block Modules differ from the simpler Record Modules described above in several ways:

Block Modules take complete responsibility for providing and accepting blocks of information from the remote BFX program. All protocol needed by the Block Modules, such as record lengths or end of file indications, must be passed from the sender to the receiver through the transferred block.

The block modules have total control over the netex datamode parameter. They can set or examine this value to send specialized data in a way that can be converted by the network.

Block Modules are almost exclusively reserved for applications that are moving binary data between dissimilar hosts. In such cases, custom DATAMODE's will often be needed to speed the work of converting floating point numbers and the like. By giving the Block Modules total responsibility for protocol, any NETEX-supported DATAMODE may be used during the file transfer process.

Block Modules are normally written to be either Sending or Receiving Block Modules. The NESi sending and receiving block modules are described in the following paragraphs. The parameter list passed to both types of modules is identical and is described under “Figure 9 Block Module Parameters” on page 28. The usage of the parameters differs between module type and is discussed in detail for each parameter.

BFXSBM – NESi Sending Block Module

This is the standard BMOD on the sending side. It calls RMOD to fetch each record from the file, packs the records into a block until full, and SWRITEs it.

For users writing their own record modules, the calling sequence must match if called from standard block modules.

The logic of this module is as follows:

- Initialize Buflen to 0, Rlev to Buflev, Rlen and Buflev to 0.
- If Rlev < 0, then set sequence number to 1 and call record module to obtain a record. If Rlen is positive or if Rlev indicates abort, then form the appropriate message.
- Else if Rlev = 16 (abort) then call record module with abort status and form a message.
- Else initialize record pointer, then for each record in the block:
    1. Point to next data area in the block.
    2. Call record module to obtain next record.
    3. Call MakeHead to create a record header.
    4. Adjust record pointer. Increment the sequence number.
- If eof or abort is set, then form the appropriate message.

BFXRBM – NESi Receiving Block Module

This is the standard BMOD on the receiving side. This module processes each received block. It calls RMOD to process each record in the block. When it is called the first time, RMOD is called to open the file. Also, at termination, RMOD is called to close the file.

For users writing their own record modules, the calling sequence must match if called from standard block modules.

The logic of this module is as follows:

- Initialize Rlev to Buflev, Rlen and Buflev to 0.
- If Rlev < 0, then set sequence number to 1 and call record module to obtain a record. If Rlen is positive or if Rlev indicates abort, then form the appropriate message.
- Else if Rlev = 16 (abort) then call record module with abort status and form a message.
- Else if Buflen > 0, then for each record in the block:
    1. Point to next data record.
    2. Break the record header into its components.

3. Verify sequence number and message/data type correct.
4. Adjust pointer to next record and call record module to write the record to the file.
5. Increment the sequence number.

- If eof or abort is set, then form the appropriate message.

Block Module Parameters

The parameter list to the Block Module is illustrated using a sample C language calling sequence in Figure 9 on page 28.

```
BFXSBM (Buff,Buflen,Buflev,Bufubit,Datamode,Msg,MsgLen,MsgLev,RMOD,
        Tfile,Seqn)
Buffptr Buff;        /* Buffer to extract from */
Integer *Buflen,     /* Length of data buffer */
        *Buflev,     /* Start/Close/Abort Flag */
        *Bufubit,    /* Unused bits in Buff */
        *DataMode,   /* Data mode received */
        *MsgLen,     /* Length of message */
        *MsgLev,     /* Message level */
        *Seqn;       /* Sequence number */
Char    *Msg;        /* Generated message */
FileSendPtr Tfile;   /* File being moved */
int (*RMOD)();       /* Record processing function */
```

Figure 9 Block Module Parameters

Buff Block Module Parameter

For a Receiving Block Module, Buff will contain a block of information from the remote BFX program.

A Sending Record Module should fill Buff with a block to be sent over netex to the remote BFX program.

Buflen Block Module Parameter

A Receiving Block Module will be given the length of the block (in bytes) in Buflen.

A Sending Block Module should return the length of the outgoing block in Buflen.

When the Block Module is called for the first time, Buflen will contain the BLOCK parameter specified by the user (or defaulted). The Block Module should return in Buflen the maximum block length (in bytes) that it expects to process.

Bufubit Block Module Parameter

Bufubit is the netex Unused Bit Count as described in the NetEx manuals and specifications. This value is principally used for sending precise amounts of data to machines whose word sizes are not a multiple of eight bits.

This value is passed to the Receiving Block Module. It may use this file or ignore it as it wishes.

A Sending Block Module will be passed a zero in Bufubit. If the Block Module wishes to use the Unused Bit Count facility, it should place a nonzero value in the field. NetEx will use the datamode and total number of useful bits sent and will provide a resultant length and unused bit count to be delivered to the receiving Block Module.

Buflev Block Module Parameter

This field has the same meaning and usage as the Buflev field for Record Modules, described earlier in this section.

DataMode Block Module Parameter

This is the DATAMODE field that is used to support the optional code conversion and Assembly/Disassembly features of NetEx.

A Receiving Block Module will be given the incoming DATAMODE of the block that is being delivered to the Block Module.

A Sending Block Module will be given the DATAMODE that was negotiated at connection time based on the machine types and the MODE parameters specified by both BFX programs. The Block Module may leave this value unchanged, or it may supply a DATAMODE in this field that will be used to transfer the block supplied by this call.

Msg Block Module Parameter

This field has the same meaning and usage as the Msg field for Record Modules, described earlier in this section.

Msglen Block Module Parameter

This field has the same meaning and usage as the Msglen field for Record Modules, described earlier in this section.

Msglev Block Module Parameter

This field has the same meaning and usage as the Msglev field for Record Modules, described earlier in this section.

Tfile Block Module Parameter

This field is a pointer to the information about the file being sent or received.

Seqn Block Module Parameter

This field is the sequence number of this record.

Block and Record Module Exit and Entry Summary

The use of the Buflen, Buflev, MSGLEN and MSGLEV parameters are somewhat confusing at first, but they are really quite consistent and flexible once understood. To aid understand, a list of the possible entry and exit conditions for Block and Record modules are shown in Table 2 on page 30 through Table 5on page 31.

Table 2. Sending Module Entry Indications

Buflen	Buflev	Msglen	Msglev	Indication
BLOCK	-1	0	--	Sending Module is being called for initialization. Module should return length of largest record or block it will return in Buflen.
0	0	0	--	Normal data transfer. Data or message should be returned.
0	16	0	--	File transfer aborted. Sending Module should do termination processing. May return message to be printed locally. Last time module is called.

Table 3. Sending Module Return Actions

Buflen	Buflev	Msglen	Msglev	Indication
> = 0	0-14	0	--	Normal data transfer. If Buflen = 0, a zero length record (block) will be sent to the receiver.
0	0-14	> 0	0-15	Normal message provided. Message printed locally and sent to receiver.
> 0	0-14	> 0	0-15	Both data and message are provided. Data will be sent to receiver. Message printed locally and sent to receiver.
> 0	15	0	--	EOI following included record. Last call to Sending Module.
0	15	0	--	EOI with last record already sent. Last call to Sending Module.
> 0	15	> 0	0-15	Last data and EOI message are provided. Data is sent to receiver. Message printed locally and sent to receiver. Last call to Sending Module.
0	15	> 0	0-15	EOI message with last record already sent. Message printed locally and sent to receiver. Last call to Sending Module.
> 0	16	0	--	Abort transfer after sending data. All data provided will be sent to receiver, followed by default abort message. Receiver will pass the abort to the Receiving Module. Last call to Sending Module.

Table 3. Sending Module Return Actions

Buflen	Buflev	Msglen	Msglev	Indication
0	16	0	--	Abort transfer. Default abort message will be sent to receiver. Receiver will pass the abort to the Receiving Module. Last call to Sending Module.
> 0	16	> 0	0-15	Abort transfer after sending data. All data provided will be sent to receiver, followed by the provided abort message. Receiver will pass the abort to the Receiving Module. Last call to Sending Module.
0	16	> 0	0-15	Abort transfer. Provided abort message will be sent to receiver. Receiver will pass the abort to the Receiving Module. Last call to Sending Module.

Table 4. Receiving Module Entry Indications

Buflen	Buflev	Msglen	Msglev	Indication
BLOCK	-1	0	--	Receiving Module is being called for initialization. Module should return length of largest record or block it expects in buflen.
>= 0	0-14	0	--	Normal data transfer. Data should be processed. If Buflen = 0, a zero-length record is being provided.
> 0	15	0	--	Included record is last one (EOI). Receiving module should perform termination after processing this record. May return message to be printed locally and sent to sender. Last call to Receiving Module.
0	15	0	--	Previous record was last one. Receiving module should perform termination processing. May return message to be printed locally and sent to sender. Last call to Receiving Module.
0	16	0	--	File transfer aborted. Receiving Module should do termination processing. May return message to be printed locally. Last call to Receiving Module.

Table 5. Receiving Module Return Actions

Buflen	Buflev	Msglen	Msglev	Indication
--	0-14	0	--	Normal return from data accept.
--	0-14	> 0	0-15	Normal return from accept with message provided. Message will be printed locally and sent to sender.
--	15	0	--	Default EOI message will be sent to sender. Last call to Receiving Module.
--	15	> 0	0-15	Provided EOI message will be sent to sender. Last call to Receiving Module.
--	16	0	--	Abort transfer. Default abort message will be printed locally and sent to sender. Last call to Receiving Module.
--	16	> 0	0-15	Abort transfer. Provided abort message will be printed locally and sent to sender. Last call to Receiving Module

Job Submission Modules

Job Submission Modules are designed to be used by those installations that have other means than the BFXJS program to submit jobs to the internal reader of the remote host, such as a job scheduler. The inclusion of a user-written job submission module allows the job to be submitted to the special job scheduler.

The calling sequence for Job Submission Modules is open. When the user-written module is installed, the user codes the call to the module into the BFXUIM module; all parameters available to UIM are available to be passed to the Job Submission Module. A restriction on the calling sequence is that the logical variable ABRTD must be cleared or set appropriately before BFXUIM is exited.

User-written Job Submission Modules are free to call any Block and/or Record Modules the user desires. The NESi standard Block and Record Modules may be used if the proper calling sequences are used.

Naming Restrictions

The routine names and common block names used by user-written modules should begin with the letters "UM"; this will prevent conflict with the names used by existing BFX routines.

Installing and Editing User-Written Modules

This section describes the procedures required to install user-written Block, Record, and Job Submission Modules into the BFX subsystem, and how to invoke the user-written modules once they have been installed.

The basic steps involved in adding a user-written module are as follows:

- The module is written to adhere to the conventions described previously.
- The routine BFXUIM in the file *bfxuim.c* is edited to include references to the new module.
- The command procedure Makefile is modified to include whatever commands are required to compile/assemble the new module.

- The *make* command procedure is executed to rebuild the BFX programs. This is done by typing "*`make HOST=host-type [CPU=cpu-type] [MYCC=compiler-name]`*".

  Note: Do not use targets "all", "clobber", or "clean"! You do not have the source code needed for these targets!

- Issue "*`make HOST=host-type install`*" which will copy the new modules to the "*`install-dir/bfx/bin`*" directory.
- Move the executable files to some generally accessible place in the file system as described in the installation section of the appropriate BFX Memo To Users document. (Invoke INSTALL script.)

At this point, the user-written module will be ready for use. The user can now write jobs that reference the module with the use of the appropriate JMOD, BMOD, or RMOD parameter statement.

Overview

The BFXUIM (user interface module) performs two functions:

- The InitTrol procedure initializes the BFX control block, either on initial call (also allocate the TROL space), or on each subsequent call. The installation may wish to modify default values which are set here.
- The BFXUIM procedure calls the BFX send or receive control routines and passes the appropriate connection routine (BFXSOF or BFXSCN) and the selected block and record modules.

BFXUIM Decision Code

For each of the three transfer types, there is a block "if" statement with the following basic structure:

```
If "STANDARD" requested,
  Use NESi-standard modules;
Else
  Report an error;
Endif.
```

User-written modules are referenced by adding to the structure:

```
If "STANDARD" requested,
  Use NESi-standard modules;
Else if "(user module 1)" requested,
  Use user-written module #1;
Else if "(user module 2)" requested,
  Use user-written module #2;
Else
  Report an error;
Endif.
```

Adding Sending Block and Record Modules

The code to reference user-written Sending Block and Record Modules is added after the line which reads:

```
***   SENDING BLOCK AND RECORD MODULES   ***
```

The following lines should be added:

```
else if((cmpstr(Tfile->BlockMod, 'UserBlockMod', 0) == 0)
 && (cmpstr(Tfile->RecMod, 'UserRecMod', 0) == 0))
 BFXSND(Tfile, Trol, ConnDat, SOFCN, userbmod, usermod, Abrtd);
```

UserBlockMod
This is the ‘logical’ name of the Block Module (for example, STANDARD).

UserRecMod
This is the ‘logical’ name of the Record Module (for example, STANDARD).

userbmod
This is the compiled name of the Block Module (for example, BFXSMB).

userrmod
This is the compiled name of the Record Module (for example, BFXSRM).

Note also that the Block and Record Modules are normally called in combination; however this is not required. If a user-written Block Module does not use a Record Module, there is no need to check RMODNM.

Adding Receiving Block and Record Modules

The code to reference user-written Receiving Block and Record Modules is added after the line which reads:

```
***   INSTALLING RECEIVING BLOCK AND RECORD MODULES   ***
```

and looks similar to the code for Sending Modules. For Receiving Modules, however, the BFXRCV routine is called instead of the BFXSND routine.

Adding Job Submission Modules

The code to reference user-written Job Submission Modules is added after the line which reads:

```
***   INSTALLING JOB SUBMISSION MODULES   ***
```

and looks similar to the code for Sending Modules with the appropriate name checking code for a User Job Submission Module.

The include file, *bfxcon.h,* declares the BFX common parameters, some of which may be of interest to the Job Submission Module. The include file, *bfxsite.h,* created automatically during installation, contains the parameter JOBSTARTER, which defines the name of the job starter file.

Appending A. BFX Error Messages

BFX generates a variety of messages during execution. Shown below is a complete list of messages with the suggested response for each. Also shown is the severity of the message (as compared with the MSGL parameter to determine if the message should be logged) and the modules that may issue the message.

```
        BFXnnns message text
```

BFX	This indicates that this is a BFX error code.
nnn	This is the error number. The BFX messages are listed in this order.
s	This indicates the message severity. The following codes are used:
	I - informational messages
	E - error messages
	S - severe error messages
	F - fatal error messages
	R - recoverable error messages
message text	This area displays the text of the message.

The following are the messages issued by BFX.

BFX001F JOB SUBMISSION FAILED.
Severity: 15 (Fatal Error)
Explanation: Transfer Initiate was unable to submit a job to the remote host. If SOE was specified, the BFX program will terminate.
User Response: The reason for job submission failure will be indicated in a previous message. Take the corrective action indicated by the previous message's description.

BFX002F BFX EXECUTION ABORTED.
Severity: 15 (Fatal Error)
Explanation: The BFX program has detected a condition that makes it impossible to successfully continue execution. If SOE was specified, the BFX program will terminate.
User Response: The reason for the terminal failure will be indicated in previous BFX messages. Take the corrective action indicated by the previous message's description.

BFX003F BFXJS EXECUTION ABORTED.
Severity: 15 (Fatal Error)
Explanation: The BFXJS program has detected a condition that makes it impossible to continue offering job submission services. Generally this is because of termination of the NetEx Communications Subsystem.
User Response: Restart NetEx or correct the error that caused NetEx to terminate. Restart BFXJS.

BFX004I BFXJS STARTED.
Severity: 4 (Detailed informational)
Explanation: The BFXJS program has been started and is ready to offer Job Submission services.
User Response: None.

BFX006S "xxxxxxxx" NOT RECOGNIZED IN CONTROL STATEMENT.
Severity: 12 (Severe Error)

Explanation: An input statement to the BFX program contains a string that is not a recognized parameter. The string will follow the error message. BFX will not transfer any files after encountering this error, but will continue to read the input file.
User Response: Correct the syntax error and resubmit the job.

BFX007W "xxxxxxxx" IS ONLY VALID FOR BFXTI JOBS – IGNORED.
Severity: 9 (Recoverable error)
Explanation: A parameter that is not applicable to the BFX program was encountered. The parameter in question will follow this message. The statement is ignored.
User Response: Although processing will continue, the probable cause is an operations or setup error. Verify that the remainder of the BFX run proceeded as intended.

BFX008W "xxxxxxxx" IS INVALID FOR THIS BFX JOB TYPE – IGNORED.
Severity: 9 (Recoverable Error)
Explanation: A parameter (such as BLOCK =) that is only used for file transfer was provided as an operand to the SUBMIT statement. The parameter is ignored and processing continues.
User Response: Although processing will continue, the probable cause is an operations or setup error. Verify that the remainder of the BFX run proceeded as intended.

BFX009W "xxxxxxxx" IS ONLY VALID FOR A FIRST BFXTF STMT – IGNORED.
Severity: 9 (Recoverable Error)
Explanation: A parameter that applies only to the first transfer (such as ID =) was encountered. The parameter in question will follow the message. The statement is ignored.
User Response: Although processing will continue, the probable cause is an operations or setup error. Verify that the remainder of the BFX run proceeded as intended.

BFX010F TO= OR FROM= HOST NAME OMITTED.
Severity: 15 (Fatal Error)
Explanation: The control parameters for the first statement of either a BFXTI or BFXTR run did not specify the name of the opposite host to allow a connection to take place. Then, a default host was not specified during installation of the BFX program.
User Response: Supply the TO= or FROM= parameter required and rerun the job.

BFX011F ID= BFX IDENTIFIER OMITTED.
Severity: 15 (Fatal Error)
Explanation: The ID parameter which uniquely identifies the BFX job on the initiating machine was not supplied. There is no default for this parameter.
User Response: Supply the ID parameter and rerun the job.

BFX012F SPECIFIED BUFFER SIZE TOO LARGE. MAXIMUM:
Severity: 15 (Fatal Error)
Explanation: The BLOCK or JBLOCK parameter specified a value greater than the indicated maximum. BFX will not transfer any files after encountering this error, but will continue to read the input file.
User Response: Correct the BLOCK or JBLOCK parameter and resubmit the job.

BFX013F "xxxxxxxx" IS NOT A SEND/RECV/SUBMIT COMMAND:
Severity: 15 (Fatal Error)
Explanation: The first operand in a control statement was not SEND, RECEIVE, or SUBMIT or a suitable abbreviation. Processing is terminate.
User Response: Correct the erroneous control statement and resubmit the job.

BFX014F ERRORS PREVIOUSLY FOUND. EXECUTION OF TRANSFER BYPASSED.
Severity: 13 (Severe Error)
Explanation: Errors were encountered parsing preceding input statements. The syntax of the input statements for following transfers will be checked, but the transfers will not occur.

User Response: Correct the errors indicated by the preceding error messages and resubmit the job.

BFX015I BFXTI STARTED.
Severity: 4 (Detailed informational)
Explanation: This message indicates that BFXTI has started and will begin to accept commands.
User Response: None.

BFX016I BFXTR STARTED.
Severity: 4 (Detailed informational)
Explanation: This message indicates that BFXTR has started and will begin to accept commands.
User Response: None.

BFX017I START OF FILE TRANSFER NUMBER nnnnn...
Severity: 3 (Detailed informational)
Explanation: This message indicates that the BFX program has started processing the input statements for the indicated file transfer.
User Response: None.

BFX018I PARAMETERS FOR THIS TRANSFER ARE:
Severity: 3 (Detailed informational)
Explanation: The BFX Program will produce a log of various file transfer parameters.
User Response: None.

BFX019S AMBIGUOUS PARAMETER "xxxxxxxx":
Severity: 12 (Severe error)
Explanation: An input statement to the BFX program contains a string that is a valid abbreviation for more than one parameter name. The string will follow the error message. BFX will not transfer any files after encountering this error, but will continue to read the input file.
User Response: Correct the syntax error and resubmit the BFX jobs.

BFX020F NETEX COMMUNICATIONS SUBSYSTEM IS NOT RUNNING.
Severity: 15 (Fatal error)
Explanation: When the BFX program attempted to establish communications, it found that the NETEX subsystem was not currently running on the local host. Processing is terminated, as data transfer is not possible without NETEX.
User Response: Consult with operations to determine whether NETEX should have been active. Resubmit the job when NETEX is active.

BFX021F NETEX COMMUNICATIONS SUBSYSTEM IS BEING SHUT DOWN.
Severity: 15 (Fatal error)
Explanation: During the connection process or in the middle of a job or file transfer, the BFX program received an indication that NETEX is abruptly terminating. This can be caused by operator cancellation of NETEX or by internal netex software problems. Processing is terminated, as no further data transfer will be possible until NETEX is restarted.
User Response: Consult with operations to determine the cause of the netex shutdown. Resubmit the job when NETEX is once again active. File cleanup procedures may be needed if a file transfer was in progress at the time of the failure.

BFX022F NETEX SYSTEMWIDE CAPACITY EXCEEDED.
Severity: 15 (Fatal error)
Explanation: During the process of establishing communications, NETEX returned an indication that it cannot handle a new connection because a limiting number of NETEX connections are already in use. Processing is terminated, as it is uncertain when the condition will clear up.
User Response: Inform operations or the NETEX system programmer of the problem. If the problem occurs frequently, netex will have to be given more resources to handle extra connections.

BFX023F REMOTE BFX PROGRAM DID NOT START.
Severity: 15 (Fatal error)
Explanation: The corresponding BFX program was not present when required. If a BFXTI program issued this message, then it waited for the TIMEOUT= interval without being connected to by the BFXTR program. If a BFXTR program issued the message, then the originating BFXTI program is no longer present to be connected to.
User Response: This is the error that will commonly occur if errors are made in the BFX setup. The most frequent causes of this error are:

- Job Control Language errors in the BFXTR job prevented successful execution of the BFXTR program.
- The TIMEOUT= value of the BFXTI job did not allow sufficient time for the BFXTR job to progress through the execution queue and connect to the originating program.
- The ID= fields of the two jobs did not agree with one another.

BFX024F REMOTE HOST CEASED COMMUNICATING.
Severity: 15 (Fatal error)
Explanation: During the transfer of a file or a job, the BFX program received an indication from netex that all communications with the other host have ceased. This is generally caused by a system crash on the remote host, abrupt failure or operator cancellation of netex on the remote host, or a hardware failure in the physical connection between the two hosts. Processing is terminated, as no further data transfer is possible.
User Response: Consult with operations to determine the cause of the failure. Resubmit the job when the connection is once again active. File cleanup procedures may be needed if a file transfer was in progress at the time of the failure.

BFX025F REMOTE BFX ABORTED EXECUTION.
Severity: 15 (Fatal error)
Explanation: During the transfer of a file or a job, the BFX program received an indication from NETEX that the BFX program on the remote host terminated abnormally. Processing is terminated, as no further data transfer is possible.
User Response: Examine the output from the job on the remote host to determine the cause of failure. Correct the error and resubmit the job.

BFX026F REMOTE HOST NetEx NOT PRESENT.
Severity: 15 (Fatal Error)
Explanation: When an attempt was made to connect to the remote BFX program, the NetEx subsystem on the local machine reported that no NetEx subsystem was present on the remote host. On some remote systems, it is possible that a "false" NRBSTAT 3500 may be returned due to the session manager on the remote NetEx being busy. BFX will attempt to connect to the remote host 6 times. Each failed connect will generate this error. However, the error report is false unless the connect attempt has failed for the sixth attempt. After the sixth attempt, processing is terminated, as data transfer is not possible without NetEx at that time.
User Response: Consult with operations to determine whether NetEx should have been present on the remote host. Resubmit the job when NetEx is available on both hosts.

BFX027F SPECIFIED HOST IS NOT ON THE NETWORK.
Severity: 15 (Fatal Error)
Explanation: When BFXTI was attempting to connect to BFXJS, or when BFXTR was attempting to connect back to the initiating BFXTI, the local NetEx subsystem returned an indication that the host name specified is not on the network specified in the Network Configuration Table. Processing is terminated, as data transfer is not possible.
User Response: The probable cause of this error is an erroneous HOST parameter. A second possibility is that the installation has changed the host names used by NetEx. Correct the error and resubmit the job.

BFX028F ACCESS TO SPECIFIED HOST DENIED.

Severity: 15 (Fatal Error)
Explanation: When BFXTI was attempting to connect to BFXJS, or when BFXTR was attempting to connect back to the initiating BFXTI, NetEx informed the program that access to the specified host has been denied by the local computer operator. Processing is terminated, as communications between the two hosts cannot take place.
User Response: Computer operations is using a feature of NetEx that can temporarily prohibit access to a host that is undergoing maintenance, performing classified or confidential work, and so on. Consult with the operations to determine when communications with the remote host will once again be permitted. Resubmit the job at that time.

BFX029F ACCESS TO LOCAL NetEx DENIED.
Severity: 15 (Fatal Error)
Explanation: When the BFX program was attempting to establish communications, NetEx informed the program that access to the local host has been denied by the local computer operator. Processing is terminated, as communications cannot take place.
User Response: Computer operations is using a feature of NetEx that can temporarily prohibit access to a host that is undergoing maintenance, performing classified or confidential work, and so on. Consult with operations to determine when communications with the remote local will once again be permitted. Resubmit the job at that time.

BFX030S NetEx ERROR: NRBSTAT = *ssss*, NRBIND = *iii*.
Severity: 12 (Severe Error)
Explanation: NetEx has reported an error to the BFX program that is not an intercepted condition. “ssss” is the four-digit status code returned by NetEx; “iii” is the data or event indication type. Processing is terminated, as the actual severity of the error is not known by the BFX program. It is a known issue that an NRBSTAT 2300 during connect processing will cause this message to be issued. That particular error will be retried prior to process termination.
User Response: Refer to NetEx documentation to determine the cause of the error. Frequently this error will be caused by earlier, more comprehensible errors. If other BFX error messages precede this one, take the corrective action suggested by those messages.

BFX031S SPECIFIED ID IS BUSY.
Severity: 12 (Severe Error)
Explanation: When BFXTI was attempting to connect to BFXJS, or when BFXTR was attempting to connect back to the initiating BFXTI, NetEx informed the program that the OFFERed application was currently in use by some other network application. The connection attempt was retried a number of times (until the DELAYBUSY time period elapsed), but the application remained in use.
User Response: If the remote application was not expected to be busy, consult with operations. Otherwise, it may be necessary to specify a higher value for DELAYBUSY and resubmit the job.

BFX032S SPECIFIED ID NOT OFFERED ON SPECIFIED HOST.
Severity: 12 (Severe Error)
Explanation: When BFXTI was attempting to connect to BFXJS, or when BFXTR was attempting to connect back to the initiating BFXTI, NetEx informed the program that the OFFERed application was not currently available. The connection attempt was retried a number of times (until the DELAYNOFR time period elapsed), but the connection was never completed. Either BFXJS (first case above) or the initiating BFXTI (second case) failed before OFFERing itself, or response times on the remote machine are very slow.
User Response: Examine the output from the remote job to determine whether the job failed before OFFERing itself. If so, correct the error that caused the failure and resubmit the job. If this situation is caused by slow response times on the remote host, it may be necessary to specify a higher value for DELAYNOFR and resubmit the job.

BFX033S NO RESPONSE FROM REMOTE BFX PROGRAM.

Severity: 12 (Severe Error)
Explanation: The local BFX program expected to receive data or a message from the remote BFX program, but none was received within a reasonable time. The probable cause of this error is that the remote BFX program has become "hung", either because of a programming error, or because of very slow response times on the remote host.
User Response: If response times are slow on the remote host, it may be necessary to specify a higher value for READTIMEOUT and resubmit the job. If a programming error is suspected and user-written modules are in use, ensure that they are not at fault.

BFX034S READ OR OFFER TIMEOUT.
Severity: 12 (Severe Error)
Explanation: The timeout specified on an SREAD or SOFFR request has expired before the request was satisfied. For SOFFR, the probable cause is that the remote job did not start in time.
User Response: If nothing unusual is reported on the other side of the transfer, try setting READTIMEOUT and/or OFFERTIMEOUT to higher values.

BFX040F NetEx COMMUNICATIONS SUBSYSTEM TERMINATED.
Severity: 15 (Fatal Error)
Explanation: During the connection process or in the middle of a job or file transfer, the BFX program received an indication that NetEx is abruptly terminating. This can be caused by operator cancellation of NetEx or by internal NetEx software problems. Processing is terminated, as no further data transfer will be possible until NetEx is restarted.
User Response: Consult with operations to determine the cause of the NetEx shutdown. Resubmit the job when NetEx is once again active. File cleanup procedures may be needed if a file transfer was in progress at the time of the failure.

BFX042F BFX PROGRAM TIMED OUT TO NETEX.
Severity: 15 (Fatal Error)
Explanation: The BFX program suspended execution for a sufficiently long time that netex terminated the connection between the two BFX programs. The current transfer is aborted, but the remaining transfers will be attempted.
User Response: This is generally because of difficulties in system tuning, or exceptionally long delays in such activities as tape mounting. If the problem was not caused by operational errors, the netex system programmer may have to raise the netex READTO parameter to compensate for the long delay.

BFX043F BFX PROTOCOL ERROR – PREMATURE DISCONNECT.
Severity: 15 (Fatal Error) BFXSND in BFXTI and BFXTR.
Explanation: The remote BFX program terminated the connection at a time when termination was not anticipated by the local BFX program.
User Response: This is an internal BFX error. It should be brought to the attention of installation BFX support personnel.

BFX044F REMOTE BFX PROGRAM TIME OUT.
Severity: 15 (Fatal Error)
Explanation: The remote BFX program suspended execution for a sufficiently long time that netex terminated the connection between the two BFX programs. The current transfer is aborted, but the remaining transfers will be attempted.
User Response: This is generally because of difficulties in system tuning, or exceptionally long delays in such activities as tape mounting. If the problem was not caused by operational errors, the netex system programmer may have to raise the netex READTO parameter to compensate for the long delay.

BFX045F BFX PROTOCOL ERROR – PREMATURE END MESSAGE.
Severity: 15 (Fatal Error) BFXRCV in BFXJS, BFXTI, and BFXTR.

Explanation: The remote BFX program sent an End-of-File message before the End-of-File record was received.
User Response: This is generally caused by user-written block and/or record modules. Re-code the user module to send the last record of the file with an EOF record level, then send the End-of-File message.

BFX046F BFX PROTOCOL ERROR – DATA AFTER EOF.
Severity: 15 (Fatal Error)
Explanation: The remote BFX program sent data after sending a record with an EOF record level.
User Response: This is generally caused by user-written block and/or record modules. Re-code the user module to send only the last record of the file with an EOF record level.

BFX047I JOB SUBMITTED.
Severity: 7 (Informational)
Explanation: The job file sent to BFXJS was submitted to the system batch queue.
User Response: None.

BFX048S JOB SUBMISSION FAILED. RC = cccccccc.
Severity: 12 (Severe Error)
Explanation: The job file submission failed. The error is described by the system status code “cccccccc”.
User Response: Correct the error indicated by the status code and resubmit the job.

BFX049I JOB FILE NOT STARTED.
Severity: 11 (Informational)
Explanation: The job file received was not started because of previously encountered errors.
User Response: Correct the error indicated by previous error messages and resubmit the job.

BFX050F BFX PROTOCOL ERROR – RUN ABORTED.
Severity: 15 (Fatal Error)
Explanation: One of the two BFX control modules detected invalid protocol in the messages exchanged between themselves. The BFX program will abend.
User Response: This is an internal BFX error that should be brought to the attention of BFX support personnel.

BFX070W PARAMETER VALUE MISSING, INVALID, OR OUT OF RANGE.
Severity: 12 (Warning)
Explanation: An invalid parameter value was supplied in the command.
User Response: Correct and reissue the command.

BFX080I FILE ffffffff RECEIVED.
Severity: 6 (Informational)
Explanation: The file ffffffff was received. This message is generated if the receiving record module does not return a message on EOF.
User Response: None.

BFX081F RECORD MODULE RETURNED A DATA RECORD DURING INITIALIZATION.
Severity: 15 (Fatal Error)
Explanation: A record module returned a data record during initialization. This is generally caused by incorrectly written user record modules.
User Response: Troubleshoot the record module and try again.

BFX082I FILE ffffffff SENT.
Severity: 6 (Informational)
Explanation: The file ffffffff was sent. This message is generated if the sending record module does not return a message on EOF.
User Response: None.

BFX083S FILE fffffff ABORT PROCESSED.
Severity: 12 (Severe Error)
Explanation: Processing of file fffffff was aborted by the block processing module.
User Response: Correct the condition that caused the user module to return the abort code and resubmit the job. It is good form for user modules to supply a message under these circumstances.

BFX084S PROTOCOL ERROR – DATA RECEIVED WHEN SENDING.
Severity: 13 (Severe Error)
Explanation: Unexpected data was received when sending.
User Response: This is generally caused by user-written block and/or record modules. Re-code the user module to correct the problem.

BFX085F MESSAGE IN DATA BLOCK.
Severity: 15 (Fatal Error)
Explanation: A message record was found inside a data block. Messages must appear in their own blocks, one to a block.
User Response: This is generally caused by user-written block module. Correct the block module and re-submit the job.

BFX086S COMMUNICATIONS LOST.
Severity: 12 (Severe Error)
Explanation: The remote BFX did not respond to an error message by this host.
User Response: Correct the problems indicated by the error messages previously logged. Resubmit the job.

BFX088S OFFER OF hhhhhhhh FAILED.
Severity: 12 (Severe Error)
Explanation: BFX could not offer. This is generally caused by insufficient privilege or NETEX is not present.
User Response: Check the job and try again.

BFX089S CONNECT TO hhhhhhhh FAILED.
Severity: 12 (Severe Error)
Explanation: BFX could not connect to host hhhhhhhh. This is generally caused by job errors on the remote host.
User Response: Check the job and try again

BFX101I FILE fffffff DONE; nnnn RECORDS SENT.
Severity: 6 (Informational)
Explanation: The NESi standard Sending Record Module has detected normal end of file on the input file specified by DDNAME "ffffffff". The total number of QSAM logical records sent for this particular file is "nnnn". At the time the message was issued, the last record of the file will already have been sent to the receiving BFX.
User Response: None.

BFX102S FILE fffffff PERMANENT I/O ERROR – RC = rrrr.
Severity: 12 (Severe Error) BFXRRM in BFXTI, BFXTR, and BFXJS.
Explanation: The QSAM access method reported a permanent I/O error reading or writing a file during transfer of the job or transfer of the file. The DDNAME of the file is indicated by "ffffffff". The SYNAD information associated with the error follows the text of the message. Transfer of this file is aborted. If batch transfer of files is being performed, BFX will attempt to transfer the rest of the specified files.
User Response: Determine the cause of the I/O error. If the error can be corrected, rerun the BFX jobs.

BFX103S CANNOT FINE RECORD MODULE rrrrrrrr.
Severity: 15 (Fatal Error)
Explanation: User specified a record module to be used by BFX but it could not be found.

User Response: Re-build BFX with the desired module(s) included or remove the RMOD parameter.

BFX104F STOP ON ERROR SET, ERRORS ENCOUNTERED.
Severity: 15 (Fatal Error)
Explanation: A previous error was detected and the SOE parameter was declared to stop on errors.
User Response: Locate the previous error and correct it.

BFX105S CANNOT FIND BLOCK MODULE mmmmmmmm.
Severity: 12 (Severe Error) BFXRCV in BFXTI, BFXTR and BFXJS.
Explanation: The Block Module specified by the BMODULE= parameter was not installed in this copy of the BFX program. Transfer of this file is aborted; if batch transfer of files is being performed, BFX will attempt to transfer the rest of the specified files.
User Response: This can be caused when the module does not exist, or if the module does exist and was generated as being the wrong type.

BFX107S CANNOT FIND JOB MODULE jjjjjjjj.
Severity: 12 (Severe Error)
Explanation: The Job Submission Module specified by the JMOD parameter was not compiled into the current copy of the BFX program. The current transfer is aborted, but the remaining transfers will be attempted.
User Response: This error will be seen either because the BFXUIM module was not updated to include the job submission module requested, or because of use of the incorrect module name.

BFX110S CANNOT OPEN INPUT FILE ffffffff.
Severity: 13 (Severe Error)
Explanation: The standard Sending Record Module attempted to open the file ffffffff. The open failed. Transfer of this file is aborted. BFX will attempt to process the rest of the command stream.
User Response: Correct the reason for the open failure. Retry the file transfer.

BFX111S RECORD MODULE INITIALIZATION FAILED.
Severity: 12 (Severe Error) BFXRBM in BFXJS, BFXTI, and BFXTR.
Explanation: A user-written Record Module returned an Abort code (Buflev = 16) when called for initialization. The module did not supply a message with the abort code, so this default message is printed. The current transfer is aborted, but the remaining transfers will be attempted.
User Response: Correct the condition that caused the user module to return the Abort code and resubmit the job. It is good form for user modules to supply a message under these circumstances.

BFX112S BLOCK MODULE INITIALIZATION FAILED.
Severity: 12 (Severe Error)
Explanation: A user-written Block Module returned an Abort code (Buflev = 16) when called for initialization. The module did not supply a message with the abort code, so this default message is printed. The current transfer is aborted, but the remaining transfers will be attempted.
User Response: Correct the reason for the open failure. Retry the file transfer.

BFX113S SENDING RECORD MODULE ABORTED TRANSFER.
Severity: 12 (Severe Error)
Explanation: A user-written Sending Record Module returned an Abort code (Buflev = 16) during the transfer of a file. The user module did not supply a message with the abort code, so this default message is printed. The current transfer is aborted, but the remaining transfers will be attempted.
User Response: Correct the condition that caused the user module to return the abort code and resubmit the job. It is good form for user modules to supply a message under these circumstances.

BFX114S RECEIVING RECORD MODULE ABORTED TRANSFER.
Severity: 12 (Severe Error)

Explanation: A user-written Receiving Record Module returned an Abort code (Buflev = 16) during the transfer of a file. The user module did not supply a message with the abort code, so this default message is printed. The current transfer is aborted, but the remaining transfers will be attempted.
User Response: Correct the condition that caused the user module to return the Abort code and resubmit the job. It is good form for user modules to supply a message under these circumstances.

BFX115S SENDING BLOCK MODULE ABORTED TRANSFER.
Severity: 12 (Severe Error)
Explanation: A user-written Sending Block Module returned an Abort code (Buflev = 16) during the transfer of a file. The user module did not supply a message with the abort code, so this default message is printed. The current transfer is aborted, but the remaining transfers will be attempted.
User Response: Correct the condition that caused the user module to return the Abort code and resubmit the job. It is good form for user modules to supply a message under these circumstances.

BFX116S RECEIVING BLOCK MODULE ABORTED TRANSFER.
Severity: 12 (Severe Error)
Explanation: A user-written Receiving Block Module returned an Abort code (Buflev = 16) during the transfer of a file. The user module did not supply a message with the abort code, so this default message is printed. The current transfer is aborted, but the remaining transfers will be attempted.
User Response: Correct the condition that caused the user module to return the Abort code and resubmit the job. It is good form for user modules to supply a message under these circumstances.

BFX117S SENDING RECORD MODULE ABORT PROCESSED.
Severity: 12 (Severe Error)
Explanation: A user-written Sending Record Module was called with an Abort code (BUFLEV = 16) during the transfer of a file. The user module did not supply a message on return, so this default message is printed.
User Response: Correct the condition indicated by previous error messages and resubmit the job. It is good form for user modules to supply a message under these circumstances.

BFX118S SENDING BLOCK MODULE ABORT PROCESSED.
Severity: 12 (Severe Error)
Explanation: A user-written Sending Block Module was called with an Abort code (BUFLEV = 16) during the transfer of a file. The user module did not supply a message on return, so this default message is printed.
User Response: Correct the condition indicated by previous error messages and resubmit the job. It is good form for user modules to supply a message under these circumstances.

BFX119S RECEIVING RECORD MODULE ABORT PROCESSED.
Severity: 12 (Severe Error)
Explanation: A user-written Receiving Record Module was called with an Abort code (BUFLEV = 16) during the transfer of a file. The user module did not supply a message on return, so this default message is printed.
User Response: Correct the condition indicated by previous error messages and resubmit the job. It is good form for user modules to supply a message under these circumstances.

BFX120S RECEIVING BLOCK MODULE ABORT PROCESSED.
Severity: 12 (Severe Error)
Explanation: A user-written Receiving Block Module was called with an Abort code (BUFLEV = 16) during the transfer of a file. The user module did not supply a message on return, so this default message is printed.
User Response: Correct the condition indicated by previous error messages and resubmit the job. It is good form for user modules to supply a message under these circumstances.

BFX121S MAXIMUM RECORD LENGTH EXCEEDS NEGOTIATED SIZE.
Severity: 12 (Severe Error)

Explanation: When the two BFX programs established a connection, the negotiated block size as determined by the user-specified parameters was insufficient to hold the largest logical record in the file to be sent.
User Response: Adjust the BLOCK parameter in one of the two BFX programs so it is sufficient to transfer the file. Note that a header of 6 bytes (bit mode) or 8 bytes (character mode) is prefixed to each record transferred.

BFX122F INVALID FILE TRANSFER PROTOCOL INFORMATION.
Severity: 15 (Terminal error; ABEND will follow)
Explanation: The file transfer information sent to the receiving BFX was found to be incorrect. This may be because of an internal BFX or NETEX integrity error, or because of a user-written Block Module that is incorrectly sending data to the standard NESi Receiving Block Module. As this error is very serious when caused by NESi code, the BFX program will unilaterally ABEND.
User Response: If the error was caused by a user-written Block Module, correct the coding error that caused incorrect data to be sent. If only NESi BFX code was used, bring the error to the immediate attention of NESi Customer Support.

BFX123F FILE TRANSFER PROTOCOL SEQUENCE ERROR.
Severity: 15 (Fatal error)
Explanation: The file transfer information sent to the receiving BFX was found to be incorrect. A record numbering check indicated that records are missing, duplicated, or out of sequence. This may be because of an internal BFX or NETEX error, or to a user-written Block Module that is incorrectly sending data to the standard NESi Receiving Block Module. The current transfer is aborted, but the remaining transfer will be attempted.
User Response: If the error was caused by a user-written Block Module, correct the coding error that caused incorrect data to be sent. If only NESi BFX code was used, bring the error to the immediate attention of NESi Customer Support.

BFX124S MODE= PARAMETERS NOT CONSISTENT FOR BOTH BFX JOBS.
Severity: 12 (Severe error) BFXSCN in BFXTI and BFXTR.
Explanation: In a BFX program pair, one side had MODE = BIT specified and the other had MODE = CHAR. The current transfer is aborted, but the remaining transfers will be attempted.
User Response: Correct the erroneous specification and transfer the files that were not sent.

BFX125S MAXIMUM RECORD LENGTH EXCEEDS BUFFER SIZE.
Severity: 12 (Severe error)
Explanation: The length of the longest record in the file to be sent (or the physical blocksize of a sequential-only device) exceeds the length of the BFX buffers. The size of the buffers is determined when the BFX programs are compiled and linked. Normally, these buffers are 32KB long (the largest block size allowed by NETEX). The current transfer is aborted, but the remaining transfers will be attempted.
User Response: If the file in question has any records that are more than 32KB long, it cannot be transferred by the NESi-supplied block and record modules; the user must supply modules to transfer the file. If the longest record in the file is less than 32KB long, this error indicates that the BFX programs have been generated with smaller-than-normal buffers. Discuss the problem with your system manager.

BFX201I FILE fffffff DONE; nnnnnnnn RECORDS RECEIVED.
Severity: 6 (Informational)
Explanation: This message is issued when the receiving BFX processes the last record of the file. When issued, it indicates that the last record was received and that the output file was successfully closed. “fffffff” is the logical name of the file used for output; “nnnnnnnn” is the number of records that were written to the output file.
User Response: None.

BFX202W FILE fffffff DONE; nnnnnnnn RECORDS RECEIVED, ttttttt TRUNCATED.
Severity: 9 (Warning)

Explanation: This message is issued when the receiving BFX program processes the last record of the file. It indicates that the last record was received and that the output file was successfully closed. However, the specified LRECL of the output file was not large enough to hold all data sent from the receiving file; the long records have been truncated. "ffffffff" is the DDNAME or FILEDEF of the file used for output, "nnnnnnnn" is the number of records written to the file; "tttttttt" is the number of records shortened because their length exceeded the LRECL of the output file. If batch execution is being used, BFX will continue to transfer the batch files.
User Response: If the truncation was an expected condition, then the message may be ignored. If it was not expected, then the DCB information for either the input or output file should be corrected and the file transferred once again.

BFX203E FILE ffffffff TRANSFER ABORTED; nnnnnnnn RECORDS RECEIVED.
Severity: 10 (Error)
Explanation: This message is issued by the receiving BFX when the file transfer process is aborted either because of the loss of NETEX communication or because of some other error detected by the BFX program. "ffffffff" is the logical name of the file used for output; "nnnnnnnn" is the number of records that were written to the output file before the abort caused the transfer to stop. The current transfer is aborted, but the remaining transfers will be attempted. The original error will be reported by other BFX messages.
User Response: Correct the error that caused the abort. Transfer the file again.

BFX204E FILE ffffffff TRANSFER ABORTED; nnnnnnnn RECORDS RCVD, tttttttt TRUNCATED.
Severity: 11 (Error)
Explanation: This message is issued by the receiving BFX code when the file transfer process is aborted either because of the loss of NETEX communication or some error detected by the sending BFX. "ffffffff" contains the DDNAME or FILEDEF of the output file; the output file before the abort caused transfer to stop. Besides the abort, records were truncated as described in message BFX101I. The transfer of this file is always aborted; subsequent files in a batch run will be transferred depending on the condition that caused transfer to abort. The original error will be reported by other BFX messages.
User Response: Correct the error that caused the abort. Transfer the file again.

BFX205I FILE ffffffff DONE; nnnnnnnn RECORDS SENT.
Severity: 6 (Informational)
Explanation: This message is issued after the sending BFX transmits the last record of the file. When issued, it indicates that the last record was sent, the input end-of-file condition was detected, and the file was successfully closed. "ffffffff" is the logical name of the file used for input; "nnnnnnnn" is the number of records that were read from the input file.
User Response: None.

BFX206E FILE ffffffff TRANSFER ABORTED; nnnnnnnn RECORDS SENT.
Severity: 10 (Error)
Explanation: This message is issued by the sending BFX when the file transfer process is aborted either because of the loss of NETEX communication or because of some other error detected by the BFX program. "ffffffff" is the logical name of the file used for input; "nnnnnnnn" is the number of records that were read from the input file before the abort caused the transfer to stop. The current transfer is aborted, but the remaining transfers will be attempted. The original error will be reported by other BFX messages.
User Response: Correct the error that caused the abort. Transfer the file again.

BFX210S CANNOT OPEN INPUT FILE ffffffff. RC = cccccccc.
Severity: 12 (Severe error).
Explanation: The sending BFX program was unable to open the input file whose logical name is specified by "ffffffff". The return code "cccccccc" is in hexadecimal.
User Response: Correct the reason for the open failure. Transfer the file again.

BFX211S CANNOT OPEN OUTPUT FILE ffffffff. RC = cccccccc.

Severity: 12 (Severe error).
Explanation: The sending BFX program was unable to open the output file whose logical name is specified by “ffffffff”. The return code “cccccccc” is in hexadecimal.
User Response: Correct the reason for the open failure. Transfer the file again.

BFX212S FILE fffffff PERMANENT I/O ERROR. RC= cccccccc.
Severity: 12 (Severe error) BFXRRM in BFXJS, BFXTI, and BFXTR.
Explanation: During the process of reading or writing the file whose logical name is “ffffffff”, a permanent I/O error occurred. The return code “cccccccc” is in hexadecimal.
User Response: Determine the cause of the I/O error. If the error can be corrected, do so and transfer the file again.

BFX220I SENDING FILE ffffffff.
Severity: 4 (Informational).
Explanation: The sending BFX has successfully opened the input file and is ready to begin transfer of data. Transmission will begin as soon as this message is issued. “ffffffff” is the logical name of the input file.
User Response: None.

BFX221I RECEIVING FILE ffffffff.
Severity: 4 (Informational).
Explanation: The receiving BFX has successfully opened the output file and is ready to receive file data. “ffffffff” is the logical name of the input file.
User Response: None.

BFX300I OFFERING ssssssss; BLOCK OUT SIZE nnnn.
Severity: 2 (Diagnostic).
Explanation: The BFX Transfer Initiate program has issued a NETEX SOFFER to wait for the BFXTR program to connect to it. The name offered is “ssssssss”, which will be the specified ID= of the user’s input parameters. The expected transfer is a receive operation; the incoming Block size that the receiving BFX would like to use is nnnn.
User Response: None.

BFX300I OFFERING ssssssss; BLOCK OUT SIZE nnnn.
Severity: 2 (Diagnostic).
Explanation: The BFX Transfer Initiate program has issued a NETEX SOFFER to wait for the BFXTR program to connect to it. The name offered is “ssssssss”, which will be the specified ID= of the user’s input parameters. The expected transfer is a receive operation; the incoming Block size that the receiving BFX would like to use is nnnn.
User Response: None.

BFX301I OFFERING ssssssss; BLOCK IN SIZE bbbb.
Severity: 2 (Diagnostic).
Explanation: The BFX program has issued a NETEX SOFFER to wait for the BFXTR program to connect to it. The name offered is “ssssssss”, which is the JID or ID parameter specified in an input statement. The block size that the offering program would like use is “bbbb”.
User Response: None.

BFX302I CONNECTING TO ssssssss ON HOST hhhhhhhh; BLOCK IN SIZE nnnn.
Severity: 2 (Diagnostic).
Explanation: When BFXTR is connecting back to its starting BFXTI, it has issued a NETEX SCONNECT to establish communications. “ssssssss” is the name to connect to as specified in the ID= or JID= user parameters; “hhhhhhhh” is the host name specified in the TO= or FROM= parameters. The direction of file transfer will cause this program to receive the file; the Block size that this program is prepared to receive is “nnnn”.

User Response: None.

BFX303I CONNECTING TO ssssssss ON HOST hhhhhhhh; BLOCK IN SIZE bbbb.
Severity: 2 (Diagnostic).
Explanation: The BFX program has issued a NETEX SCONNECT with a previously offered BFX program. The name to connect to is “ssssssss”, which is the JID or ID parameter specified in an input statement. “hhhhhhhh” is the host name specified in a HOST parameter statement. The block size that the connection program would like to use is “bbbb”.
User Response: None.

BFX304I CONNECT COMPLETE.
Severity: 2 (Diagnostic).
Explanation: A previously issued NETEX SCONNECT has completed successfully.
User Response: None.

BFX305W CONNECT FAILED; ssssssss BUSY.
Severity: 1 (Diagnostic).
Explanation: A previously issued NETEX SCONNECT did not succeed because the application named “ssssssss” was in use by some other network application. The BFX program will retry the connection at intervals determined by the DELAYTIME parameter until the SCONNECT succeeds or until the time specified by the DELAYBUSY parameter elapses.
User Response: None.

BFX306W CONNECT FAILED; ssssssss NOT OFFERED.
Severity: 1 (Diagnostic).
Explanation: A previously issued NETEX SCONNECT did not succeed because the application named “ssssssss” was not offered on the remote host. This is not always a terminal error. It can be caused by connecting to BFXJS during a small “window” between batch job submission from a heavily loaded machine to a very responsive one. The BFX program will retry the connection at intervals determined by the DELAYTIME parameter until the SCONNECT succeeds or until the time specified by the DELAYNOFR parameter elapses.
User Response: None.

BFX307I OFFER COMPLETE.
Severity: 2 (Diagnostic).
Explanation: A previous NETEX SOFFER request has completed successfully.
User Response: None.

BFX308I CONFIRM ISSUED.
Severity: 2 (Diagnostic).
Explanation: A NETEX SCONFIRM is being issued in response to a previously completed SOFFER.
User Response: None.

BFX309I CONFIRM COMPLETE.
Severity: 2 (Diagnostic).
Explanation: A previously issued NETEX SCONFIRM has completed successfully.
User Response: None.

BFX310I CONNECT CONFIRM READ ISSUED.
Severity: 2 (Diagnostic).
Explanation: Following a successful SCONNECT request, the BFX program has issued an SREAD to obtain the SCONFIRM response from the other program.
User Response: None.

BFX311I CONNECT CONFIRM COMPLETE

Severity: 2 (Diagnostic).
Explanation: The SREAD issued to accept an SCONFIRM message from the remote BFX program has completed successfully. The NETEX session establishment process is now complete.
User Response: None.

BFX312I BLOCK SIZE bbbbb, DATAMODE dddd, LCM lll.
Severity: 2 (Diagnostic) BFXSCN in BFXTI and BFXTR.
Explanation: This message is issued by the BFX program when the session negotiation process is complete. "bbbbb" is the NETEX block size that will be used during file transfer. "dddd" is four hexadecimal digits that give the NETEX DATAMODE to be used based on the requirements of the two BFX programs. "lll" is the Least Common Multiplier size negotiated, and will be greater than one when the connection is to a processor which has more than one character per "word".
User Response: None.

BFX313S OFFER OF ssssssss FAILED.
Severity: 12 (Severe error).
Explanation: The NETEX SOFFER of "ssssssss" failed. The offer is not retried.
User Response: A NETEX-type error message will have preceded this message. Take action based on that error message and resubmit the job.

BFX314S CONNECT TO ssssssss FAILED.
Severity: 12 (Severe error).
Explanation: The NETEX SCONNECT to "ssssssss" failed. The connect was retried a number of times, but continually failed.
User Response: A NETEX-type error message will have preceded this message. Take action based on that error message and resubmit the job.

BFX401S ERROR DECODING PARAMETER VALUE.
Severity: 12 (Severe error).
Explanation: The value specified in an input statement to be assigned to a parameter was not a valid integer. The FORTRAN error number returned from the READ statement that attempted to decode the value will follow the message. BFX will not transfer any files after encountering this error, but will continue to read the input file.
User Response: Correct the incorrect parameter value and resubmit the job.

BFX402S VALUE MUST BE SPECIFIED FOR THIS PARAMETER – NO DEFAULT.
Severity: 12 (Severe error).
Explanation: No value was specified in an input statement to be assigned to a parameter that does not have a default value (such as ID). BFX will not transfer any files after encountering this error, but will continue to read the input file.
User Response: Supply a value for the parameter and resubmit the job.

BFX403S MODE MUST BE BIT OR A VALID CHARACTER SET.
Severity: 12 (Severe error).
Explanation: A string (or abbreviation) other than "BIT" or "CHARACTER" was specified in a MODE= input statement. BFX will not transfer any files after encountering this error, but will continue to read the input file.
User Response: Correct the incorrect string and resubmit the job.

BFX404I INPUT STATEMENTS ARE:
Severity: 2 (Diagnostic).
Explanation: This message is generated by the BFX program before the input statements are read. The input statements will be echoed to the log file.
User Response: None

BFX405R MESSAGE LEVEL MUST BE 0 TO 16 INCLUSIVE, IGNORED.
Severity: 9 (Error).
Explanation: The value specified in an input statement to be assigned to the message level parameter (MSGLEVEL=) was outside the range 0-16 inclusive. The statement is ignored.
User Response: Supply a valid message level and resubmit the job.

BFX406S ERROR READING INPUT STATEMENT.
Severity: 12 (Severe error).
Explanation: A FORTRAN READ of an input statement failed. The FORTRAN error number returned from the failing READ statement will follow the message.
User Response: Correct the problem that caused the READ error and resubmit the job.

BFX407E PARAMETER VALUE TOO LONG; TRUNCATED. MAXIMUM:
Severity: 9 (Error).
Explanation: A string value specified in an input statement to be assigned to a parameter was longer than the parameter filed itself. The string is truncated to fit in the field. The maximum length of the filed in question is printed following the message. Processing will continue normally.
User Response: Unexpected results may occur because of this truncation. If so, supply a valid length string and resubmit the job.

BFX408I ALL FILE TRANSFERS HAVE BEEN PROCESSED.
Severity: 4 (Informational)
Explanation: End-of-file was reached on the INPUT file. All requested file transfers have been processed (although some may have aborted or may have been bypassed).
User Response: None.

BFX409S NUMERIC VALUE REQUIRED FOR xxxx, yyyy GIVEN.
Severity: 12 (Severe error).
Explanation: An input token requiring a numeric parameter was not given one.
User Response: Fix the input stream and re-submit.

BFX410S VALUE xxxx OUT OF RANGE ON yyyy COMMAND.
Severity: 12 (Severe error).
Explanation: The parameter value xxxx on command yyyy exceeded the allowed range.
User Response: Select a valid value and reissue the command.

BFX411F BFX IMPROPERLY BUILT.
Severity: 15 (Fatal error).
Explanation: The variable, PROGTYPE, was incorrectly set in module *ptype.h* when compiling and linking this BFX component.
User Response: Refer to the installation section in the appropriate Memo To Users, correct the problem, and rebuild the BFX component.

BFX801I nnnn RECORDS SENT AT mmmm BITS PER SECOND.
Severity: 15 (Informational).
Explanation: This message displays installation performance test results.
User Response: None.

BFX901I Hxx1 b.r dddd.
Severity: 15 (Informational – always issued).
Explanation: BFX sign-on line. Indicates the BFX product's release level and build date.
User Response: None.

BFX902S COMMAND INPUT FILE 'filename' CANNOT BE FOUND
Severity: 15 (Severe Error)

Explanation: BFX was unable to open the specified input command file, usually because the file does not exist or because of insufficient privilege.
User Response: Ensure that the file exists and has the proper privilege settings.

BFX903S MALLOC ERROR: CANNOT ALLOCATE ENOUGH MEMORY.
Severity: 15 (Severe error).
Explanation: BFX failed to allocate physical memory for data or message buffer space.
User Response: Retry the operation.

BFX904S ERROR: CHAR MODE MAXIMUM RECORD LENGTH IS 9999 BYTES.
Severity: 15 (Severe error)
Explanation: The maximum value of RMAXL is 9999 in CHARACTER mode. A record was read that exceeded 9999 bytes during a CHARACTER mode send.
User Response: Change the mode to BIT or modify the file to have shorter records.

Index