Bulk File Transfer (BFX™) Utility

for UNIX Operating Systems

Release 3.4.2

Software Reference Manual

MAN-REF-UNXBFX-3.4.2-01

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 (092008)	• Added documentation related to "RPARM [NOCR | NOLF | NOCRLF]" • Changed document revision to "-02".
MAN-REF-UNXBFX-3.4	• Manual updated to correspond to NESi release of BFX 3.4; added secure job transfer feature.
MAN-REF-UNXBFX-3.4-01	• Add GPG signature to distribution for authentication..
MAN-REF-UNXBFX-3.4-02	• Add support for isolating the NetEx traffic (including secure job transfer) to specific IP addresses by using NTX prefixes in the hostnames and in the resolver records.
3.4.2-01	• Add Support for Solaris SPARC

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:

support@netex.com

or at our website:

http://www.netex.com

Always include the complete publication number (e.g. MAN-REF-UNXBFX-3.4.2) 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 type 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.

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

The remaining Appendices contain the Installation instructions specific to each platform supported.

BFX uses NetEx but the reader does not need to understand NetEx to use this manual and BFX

Reference Material

The following manuals contain related information.

Number	Title and Description
MAN-REF-HUNXIP	*NetEx/IP for UNIX Systems Software Reference Manual*
MAN-CNET-CONF-MGR	*"C" Configuration Manager and NetEx Alternate Path Retry (APR) User Guide*

Notice to the Reader

The material contained in this publication is for informational purposes only and is subject to change without notice. Network Executive Software is not responsible for the use of any product options or features not described in this publication. It assumes no responsibility for any errors that may appear in this publication. Refer to the revision record (at the beginning of this document) to determine the revision level of this publication.

Network Executive Software does not by publication of the descriptions and technical documentation contained herein, grant a license to make, have made, use, sell, sublicense, or lease any equipment or programs designed or constructed in accordance with this information.

This document may contain references to the trademarks of the following corporations.

Corporation Trademarks and Products

Network Executive Software	**NetEx, BFX**
Hewlett-Packard Corporation	**HP**
The Open Group	**UNIX**
Linus Torvalds	**Linux**
IBM	**AIX**
Oracle	**Oracle Solaris**
SPARC International, Inc.	**SPARC**

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

Installation of all BFX products is explained in the appendices at the end of this manual, “Appendix B. H801 (Linux) and H621 (AIX) Installation” and “Appendix C. H691 (Solaris) Installation”. 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

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 NetEx software that dynamically converts the host data from one character set to another.

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

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 networks. (2) The communications facility used to interconnect two different networks.

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

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 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.

path: A route that can reach a specific host or group of devices or an order of searching for an executable in a UNIX shell.

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 NetEx products installed (UNIX BFX requires the corresponding 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 code conversion.

Supported Configurations

BFX uses the NetEx/IP communications subsystem and 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 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 1. 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).

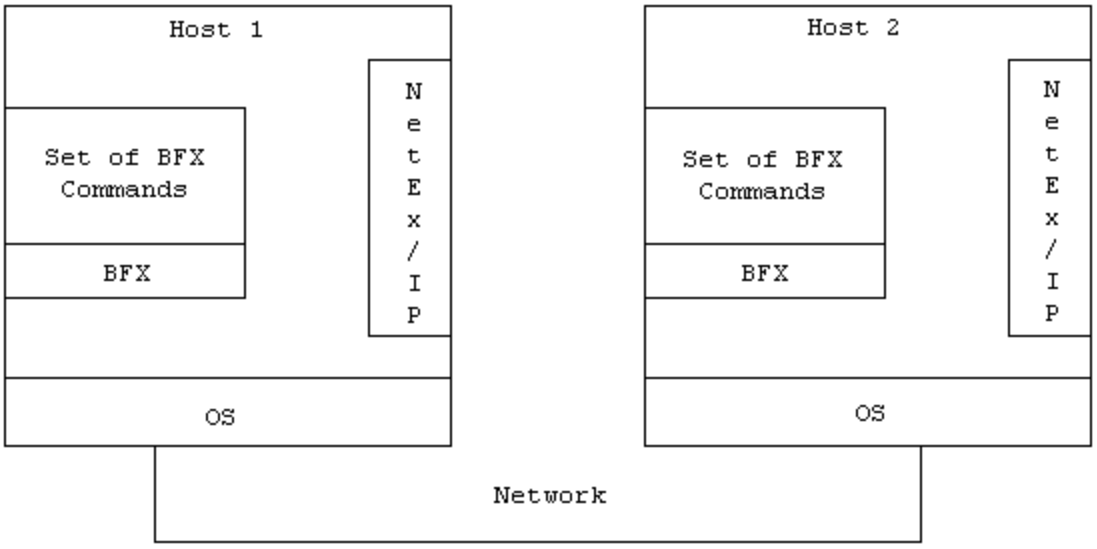

Figure 1. Sample Configuration Using BFX

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 code 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.

Four BFX Programs

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

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

The BFXSJS program also runs in the processor that responds to the transfer request. BFXSJS securely processes jobs transferred to the responding machine.

All four 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 speci-

fies 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 illustrates the manual job submission process, using these sample local and remote files.

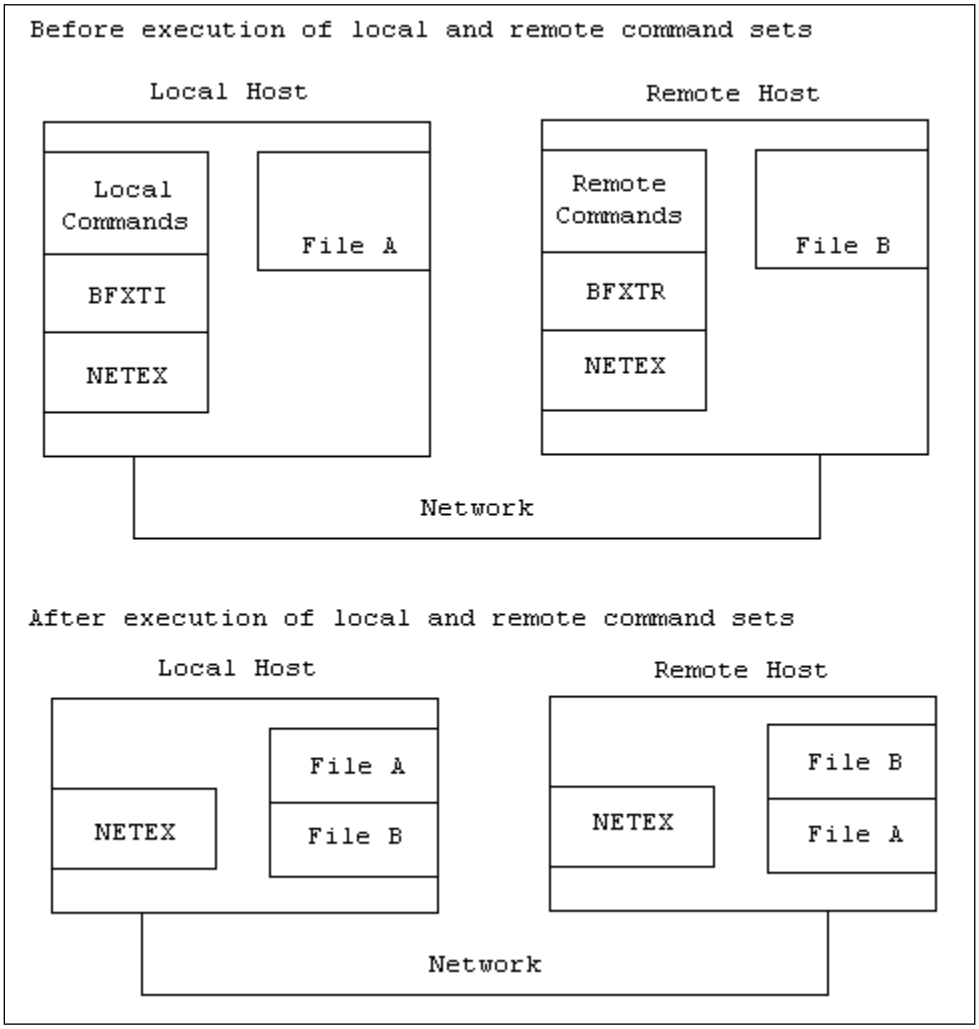

Figure 2. Manual Job Submission Example

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 or BFXSJS program on the remote host. BFXJS then automatically submits this remote command set to BFXTR for execution on the remote host. (If the job is to transfer securely, BFXTI opens a secure connection to BFXSJS and sends the remote command set to 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 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/BFXSJS on the remote host. BFXTI then transfers the remote job to BFXJS/BFXSJS on the remote host.

After execution of the first part of the local command set, BFXJS/BFXSJS submits the remote job to run 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 ready 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/BFXSJS programs are ready for the next job.

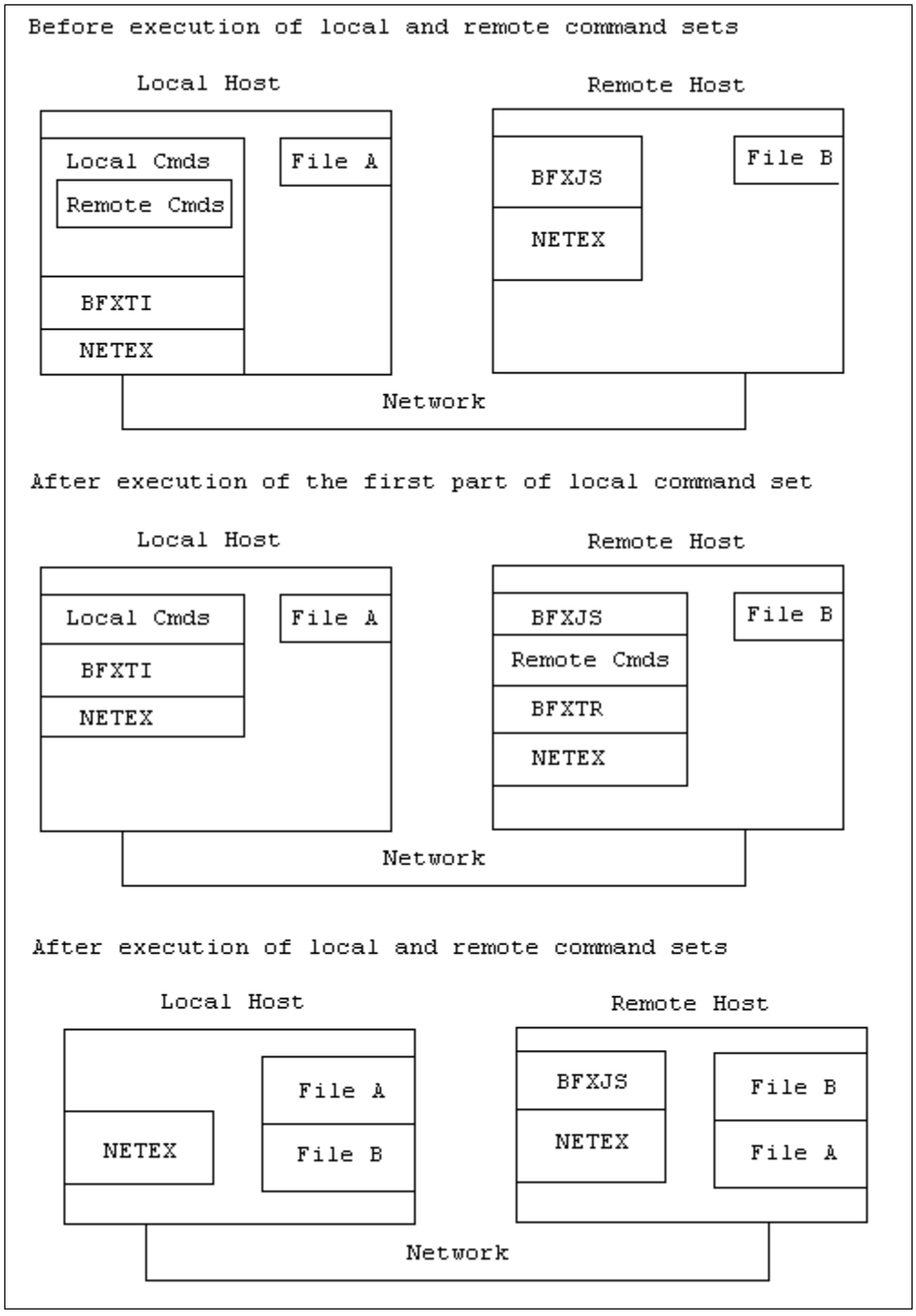

Figure 4. Automatic Job Submission Example

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
#!/bin/sh
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 and bfxsjs` (as super-user) if it is not already running:

```
AIX:
# su -
# startsrc -s bfxjs
# startsrc -s bfxsjs

Linux:
# su -
# service bfx start
```

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/BFXSJS program on the remote host. BFXJS/BFXSJS 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.

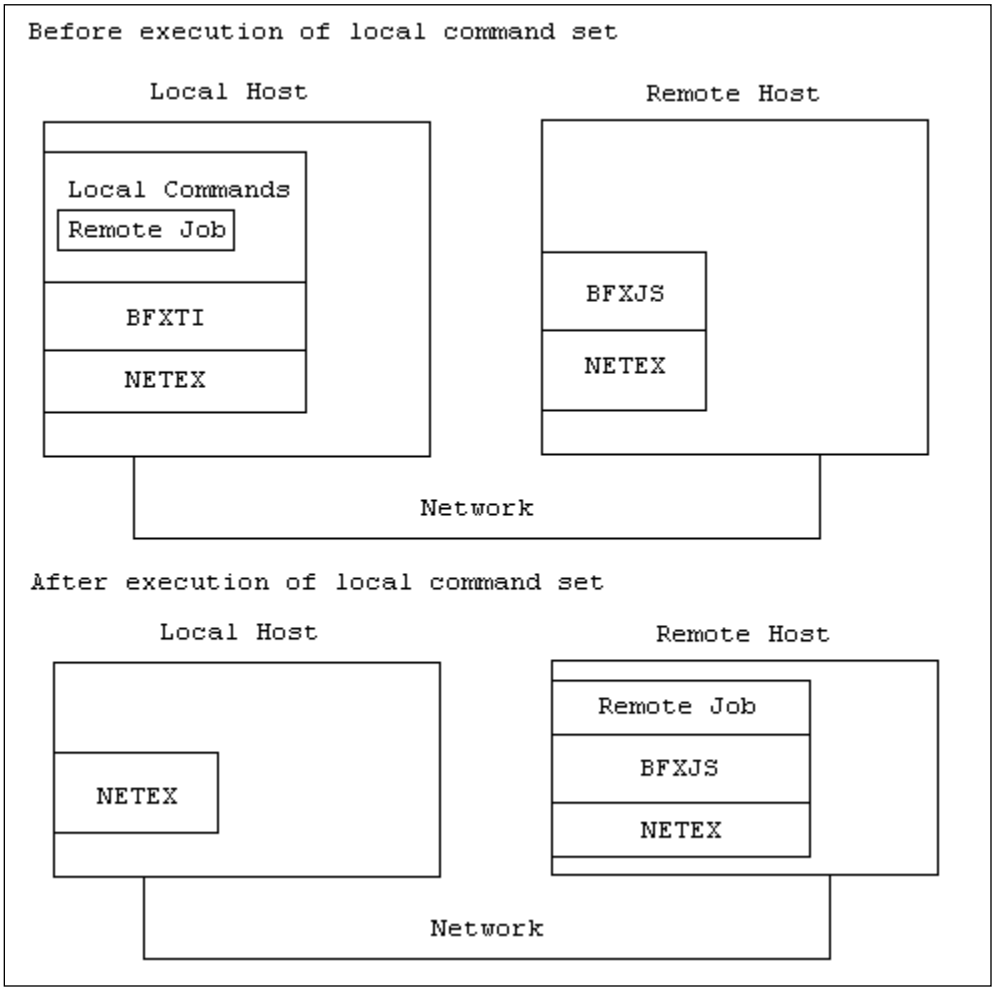

Figure 6. Remote Job Submission Example

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/BFXSJS on the remote host. BFXTI then transfers the remote job to BFXJS/BFXSJS on the remote host. BFXTI then terminates. BFXJS/BFXSJS 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
#!/bin/sh
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:

```
AIX:
$ su -
# startsrc -s bfxjs
# startsrc -s bfxsjs

Linux:
$ su -
# service bfx start
```

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.

Secure Remote Job Submission Example – UNIX Specific

The following is a UNIX specific example of secure remote job submission. In this case the job with the username and password will be transferred on an encrypted connection.

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 –
SECURE MSGLVL 0 MODE CHAR BLOCK 4096
```

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

```
AIX:
$ su -
# startsrc -s bfxjs
# startsrc -s bfxsjs

Linux:
$ su -
# service bfx start
```

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.

Note: Do not send binary files in character mode. The character transfer truncates binary zeros and file transfer will result in loss of integrity of the file even if it successfully **transfers.**

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

By utilizing the Secure Job Submission configuration parameter, all BFX jobs can be transferred utilizing an encrypted connection. The subsequent file transfers do not employ the secure connection as the existing host restrictions on utilization, validation, and accounting will remain in effect.

Summary

BFX is a system that consists of four separate programs:

- An executable image, BFXTI (BFX Transfer Initiate), that is started in the processor that wishes to either send or receive a file. Upon successful completion, it returns a return code of 255, otherwise it returns 0. 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 or 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. Upon successful completion, it returns a return code of 255, otherwise it returns 0. 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.

- 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 use 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.
- The last component, BFXSJS is a resident program that must be present in the non-initiating CPU when BFX is to be run and secure job transfer is required. Its sole function is to accept the job destined for the responding host that was provided to the BFXTI program. In this case, BFXTI and BFXSJS use an SSL connection to transfer the job, and BFXSJS submits the job, unchanged, to the internal reader of the responding machine. It should also be noted that BFXSJS 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 (or BFXSJS) job submission program. The BFXJS/BFXSJS 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/BFXSJS utility. The associated 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/BFXSJS; 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 described in this section.

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 16 summarizes where each command may be issued. Most of the parameters have 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.

Once default values are established they may be changed by commands in the bfxti.cfg, bfxtr.cfg, bfxjs.cfg and bfxsjs.cfg. These files are processed by BFXTI, BFXTR, BFXJS and BFXSJS after the defaults are set,

but prior to the redirected input or standard input processing. This is a way for a site to have custom defaults for all jobs. Redirecting an input file with the BFX Commands is done using the following syntax: bfxjs < bfxinput, or similar for bfxti and bfxtr.

Table 1. BFX Command Defaults

Command	Default	Issued From
SEND	SEND	BFXTI, BFXTR (Must be defined in BFX job file)
RECEIVE	SEND	BFXTI, BFXTR (Must be defined in BFX job file)
JOBSUBMIT	SEND	BFXTI (Must be defined in BFX job file)
BLOCK	32767	BFXTI, BFXTR
CMDPORT	6951	BFXSJS
DEBUG	OFF	BFXSJS
DELAYTIME	5	BFXTR
FILE	BFXFILE	BFXTI, BFXTR (Must be defined in BFX job file)
FROM	NTXLCL	BFXTI, BFXTR
TO	NTXLCL	BFXTI, BFXTR
ID	"????????"	BFXTI, BFXTR
JBLOCK	32767	BFXTI, BFXJS
JID	BFXJS	BFXTI, BFXJS
JOBFILE	RMTJOB JOBxxxxxxx	BFXTI BFXJS, BFXSJS (Must be defined in BFX job file)
JREPEATCONN	5	BFXTI
JRMAXL	240	BFXTI, BFXJS
LOGCMDS	ON	BFXSJS
MODE	CHARACTER	BFXTI, BFXTR (Must be defined in BFX job file)
MSGLVL	4	BFXTI, BFXTR, BFXJS, BFXSJS
MSGSYSLOG	1	BFXSJS
NEWHOST	None	BFXTI
NOSUBMIT	None	BFXTI
PORT	6950	BFXSJS
RECBUFF	0	BFXTI
RMAXL	1024	BFXTI, BFXTR, BFXJS
RPARM	(blanks)	BFXTI, BFXTR, BFXJS
REPEATCONN	20	BFXTR
SECURE	NOSECURE	BFXTI

Command	Default	Issued From
NOSECURE	NOSECURE	BFXTI
SJSPORT	6950	BFXTI
SOE	NOSOE	BFXTI, BFXTR
NOSOE	NOSOE	BFXTI, BFXTR
TIMEOFFER	240	BFXTI, BFXJS
TIMEOUT	60	BFXTI, BFXTR, BFXJS
TIMESTAMP	NOTIMESTAMP	BFXTI, BFXTR, BFXJS, BFXSJS
NOTIMESTAMP	NOTIMESTAMP	BFXTI, BFXTR, BFXJS, BFXSJS
TRACE	/usr/share/nesi/bfx/bfxsjs.log	BFXSJS

Command Descriptions

The following paragraphs describe the BFX commands. The BFX commands were summarized in Table 1 on page 16.

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. 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 cannot 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 23.

CMDPORT Command

This command specifies the TCP port BFXSJSOP will use for communication. This command takes one numeric parameter.

DEBUG Command

This command specifies whether BFXSJS will log additional information for degugging purposes. This command takes ON or OFF as parameters.

DELAYTIME Command

This command specifies how long the BFX component will delay between reconnects in attempting to reach the remote host. 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 240 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.

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.

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 240 characters in length.

JREPEATCONN Command

This command specifies the number of times BFXTI should retry the connect to BFXJS if the connect gets a 3501 (not offered) or 3502 (busy) or 2300 (read timeout) status. It will delay DELAYTIME secs in between connect attempts.

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. Default is 240, minimum is 0 and maximum is 9999.

The JRMAXL command is legal in BFXTI and BFXJS.

LOGCMDS Command

This command determines whether BFXSJSOP commands will be logged. Valid parameters are ON or OFF.

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.

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.

MSGSYSLOG Command

This command specifies where BFXSJS will send messages for logging.

0, 1, 2, or 3. A value of 0 means minimal output to the log defined by the TRACE command and no output to syslog. A value of 1 means output to the TRACE log only. A value of 2 means output to the system log. A value of 3 means output to both logs.

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.

PORT Command

This specifies the TCP port BFXSJS will listen on to accept connections.

RECBUFF Command

0, 16-*65535* – the size of the buffer to be used to accumulate records to be sent to TCP. 0 means do not use buffering – each record will be sent separately. Default: 0

REPEATCONN

This command specifies the number of times BFXTR should retry the connect to BFXTI if the connect gets a 3501 (not offered) or 3502 (busy) or 2300 (read timeout) status. It will delay DELAYTIME secs in between connect attempts.

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. Default is 1024, minimum is 0 and maximum is 9999 for character, and 32767 for bitmode.

RPARM Command

This command specifies a string of parameter information that will be passed to the default Record Module for processing.

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 UNIX BFX products and only when the UNIX BFX 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 record with an (LF). This means that if a (CR) is present, the (CR) will be written to the file, and since the record 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 a UNIX BFX 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 record (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 record (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).

SECURE Command

The SECURE command specifies that for this file transfer, the secure jobfile transfer facility will be used. This overrides any setting in the bfxsecparms.cfg configuration file.

NOSECURE Command

The NOSECURE command specifies that for this file transfer, the secure jobfile transfer facility will NOT be used. This overrides any setting in the bfxsecparms.cfg configuration file.

SJSPORT Command

The SJSPORT command specifies the port that BFXSJS is listening on. This overrides any setting in the bfxsecparms.cfg configuration file.

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.

TRACE Command

This specifies the file BFXSJS will use for logging messages.

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.

Secure Jobfile Transfer Configuration

NetEx Hostnames vs. IP Hostnames

NetEx hostnames might not be the same as IP hostnames. IP hostnames are associated with one or more IP addresses. NetEx hostnames are associated with one or more NetEx adapters (or GNAs) which in turn are associated with an IP address. To use the Secure Jobfile Transfer feature, the NetEx hostnames MUST be added to the name resolver for each IP address which is associated to its NetEx adapters/GNAs. For example if a NetEx hostname is 'netexhost' and it has two adapters with GNAs of 0111 and 0112 with 10.1.1.98 and 10.1.2.99 IP addresses respectively, then the following entries are likely in DNS:

10.1.1.98 NTX00000111

10.1.2.99 NTX00000112

To use Secure Jobfile Transfer feature, the following entries must be added to DNS:

10.1.1.98 netexhost

10.1.2.99 netexhost

If you wish to isolate the secure jobfile transfer to specific IP addresses, by adding NTX prefix to the IP hostname, the secure components will perform name resolution for the IP hostname with the NTX prefix. These entries must be also added to the name resolver database.

For example, netexhost also uses IP addresses 10.1.3.98 and 10.1.4.99. We do not want to use these addresses for the secure jobfile transfer. Additional entries using the NTX prefix should be added to DNS:

10.1.1.98 NTXnetexhost

10.1.2.99 NTXnetexhost

Certificate and Key

Contact your Security Administrator to obtain pem formatted keys and certificates. These must be place in the following files:

/usr/share/nesi/bfx/key.pem

usr/share/nesi/bfx/cert.pem

If you would like to generate a self-signed key for testing, you can utilize openSSL tools to do this. One example of this is the following command:

```
openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout
/usr/share/nesi/bfx/key.pem -out /usr/share/nesi/bfx/cert.pem
```

Configuration Files

Secure jobfile transfer uses 2 configuration files used which may be altered at the customer site.

BFXSJS Configuration File

The secure BFXSJS job submitter has a configuration file located in the install path.:

BFXSJS port 6950 cmdport 6951 trace /usr/share/nesi/bfx/bfxsjs.log debug off logcmds on msgsyslog 1

Where:

port: the TCP port on which BFXSJS will listen for secure BFXTI to connect to. Default 6950.

cmdport: the TCP port on which BFXSJS will listen for operator commands. Default 6951.

trace: the filename for debug messages and traces. Default /usr/share/nesi/bfx/bfxsjs.log

debug: on|off. Default off.

logcmds: on|off. Default on.

msgsyslog: 0, 1, 2, or 3. A value of 0 means minimal output to the bfxsjs.log and no output to syslog (if available). A value of 1 means output to the bfxsjs.log only. A value of 2 means output to the syslog (if available). A value of 3 means output to both logs. Default 1.

BFXTI Secure Parameters Configuration File

The use of the secure jobfile transfer facility can be controlled by the use of the bfxsecparms.cfg file, which resides in /usr/share/nesi/bfx. The bfxsecparms.cfg specifies whether to use the secure feature with all hosts or with certain named hosts, which port BFXSJS is listening on, and whether to buffer the jobfile records being sent to BFXSJS.

The following is the default configuration:

```
SECURE NO
SJSPORT 6950
RECBUFF 0
```

Where:

SECURE: NO: do not use the secure jobfile transfer for any hosts (except for those specified with the SECHOST parameter). YES: use the secure jobfile transfer for all hosts. Default: NO

If SECURE is specified as NO, up to 5 hosts may be specified with the SECHOST parameter which will use the secure feature.

SJSPORT: the port on which BFXSJS is listening, as specified in the BFXSJS configuration file. Default: 6950

RECBUFF: 0, 16-*65535* – the size of the buffer to be used to accumulate records to be sent to TCP. 0 means do not use buffering – each record will be sent separately. Default: 0

SECHOST hostname: this host will use secure jobfile transfer. Up to 5 SECHOST statements may be specified.

BFXSJS Output Files

BFXSJS may output to 4 files: bfxsjs.stdout, bfxsjs.stderr, bfxsjs.log, and /var/log/messages (if syslog is available). These file integrated for diagnostic purposes.

BFXSJSOP Operator Interface

The BFXSJSOP program may be used to interface to BFXSJS. It will connect to BFXSJS via the TCP cmdport specified in the BFXSJS configuration file. Default is 6951. The program is called by:

```
bfxsjsop [-p port] hostname cmd
```

Where:

port: the port BFXSJS is listening on for commands. Default is 6951.

The following are the possible commands.

help:

vers:

debug ON|OFF:

logcmds ON|OFF:

msgsyslog 0|1|2|3:

BFX Send/Receive Sample Jobs

Shell scripts named ‘bfxsend’ and ‘bfxrecv’ have been provided in `/usr/share/nesi/bfx/bin` as a BFX generic send/receive model. Ideally you can type ‘bfxsend’ or ‘bfxrecv’, answer the prompts, and transfer the given files to or from a remote host. Some sample remote hosts you might interact with are included in the scripts. Review the scripts and make your own site modifications

- bfxsend – A script to send a file to another system. Creates a jobfile to be send to common system types. May need to be modified for specific systems at local site.
- bfxrecv – A script to pull a file from another system. As above this may need to be modified for the local system type mix.
- bfxtest – A local loopback test. This should be the first job run to check out a new installation.

Appendix 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 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. TR connecting to TI will try REPEATCONN # of times and delay DELAYTIME in between each attempt. TI connecting to JS will try JREPEATCONN # of times and delay DELAYTIME in between each attempt.
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 DELAYTIME 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, and 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.

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 resubmit 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 ffffffff 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.

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.

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 ffffffff 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. “ffffffff” 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 ffffffff DONE; nnnnnnnn RECORDS RECEIVED, tttttttt 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 ffffffff 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 submissions 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 DELAYTIME 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.

BFX412E Cmd line too long. Maximum is 240.
Severity: 15 (Fatal error).
Explanation: The cmd line was longer than allowed.
User Response: Ensure the cmd lines are not longer than 240 characters (including the "-" continuation character).

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.

Appendix B. H801 (Linux) and H621 (AIX) Installation

Prerequisites

The following are prerequisites for installing H801 and H621 BFX:

- The privileges to create directories, load files from the distributed software media, create links, edit system startup, and system login file.
- An installed and operational Hxx0IP NetEx product for the platform.

Pre-Installation

The location of the executable programs from your previous installation of BFX will be required for the Installation section.

The previous installation of BFX will have to be de-installed prior to installation.

Accessing the UNIX BFX software distribution

The BFX software is available as an RPM which may be downloaded from NESi. Contact NESi Customer Support to request the download link.

Getting the NESi Public Key

The RPM software distribution package is signed to ensure integrity and authenticity. It is recommended to install the NESi public key and verify the signature of any software packages before installation.

You can download the key from http://www.netex.com/files/1813/7123/4715/RPM-GPG-KEY-netex.txt.

Importing the NESi Public Key

Install the public key as super user with the command:

```
# rpm –-import RPM-GPG-KEY-netex.txt
```

Verifying Signatures

You can verify the RPM signature to ensure that a package has not been modified since it has been signed. Verification will also check that a package is signed by the vendors or packagers key.

To verify the signature, use the -K or –checksig option to the rpm command:

```
# rpm –K Hxx1-3.4. . ..rpm
```

Installation

The Linux and AIX BFX installation uses the RPM package management utility. BFX installs in the same fashion as all other RPM-based software distributions.

A link to the distribution can be obtained from NESi Support. The installation steps are:

Execute RPM as 'root' to install or upgrade the software:

If this is an initial installation, install the software as super user with the command:

```
# yum install Hxx1-3.4...rpm
```

or

```
# rpm –i Hxx1-3.4...rpm
```

If the NESi public key has not been installed use the command:

```
# yum –-nogpgcheck install Hxx1-3.4...rpm
```

or

```
# rpm –i –-nosignature Hxx1-3.4...rpm
```

This will copy the execution environment to /usr/share/nesi/bfx/bin and lib.

For AIX installations:

Enter the following command to see if the SSL library is installed; the following shows the correct minimum level library required.

```
# lslpp -L openssl.base
Fileset                    Level  State  Type  Description (Uninstaller)
----------------------------------------------------------------------------
----
         openssl.base            0.9.8.1101    C     F    Open Secure
Socket Layer
```

If this is the case proceed with installation of the rpm with no dependency check:

```
# rpm -i --nodeps H621-3.4-1.ppc.rpm
```

If the output did not show at least that minimum level or no library at all, you will need to install the openssl package base to the above level before proceeding.

Customers do NOT have the option of installing Hxx1 into a user-defined directory location.

If BFX or its executables are not in the PATH, then hard coded paths to the executables will have to be used in all BFX jobs/scripts. This can become a significant maintenance problem for the users in the future. However if this how all your existing BFX jobs are setup you will have to add symbolic links from the current location to the path specified in your jobs.

Upgrading H801 (Linux) and H621 (AIX)

If you are upgrading an existing installation of BFX, it is strongly recommended that any running NetEx/IP and BFX process be stopped. Using the "rpm -U" command preserves any customized files in this package and the replacement files are installed with extensions of .rpmnew. Any files that are not in the package but in package directories will also be preserved. Upgrade the software as super user with the command:

```
# yum upgrade Hxx1-3.4...rpm
```

or

```
# rpm –U Hxx1-3.4...rpm
```

If the NESi public key has not been installed use the command:

```
# yum –-nogpgcheck upgrade Hxx1-3.4...rpm
```

or

```
# rpm –U –-nosignature Hxx1-3.4...rpm
```

This will copy the execution environment to /usr/share/nesi/bfx/bin and lib.

Customers do NOT have the option of installing Hxx1 into a user-defined directory location.

NOTE: If BFX or its executables are not in the PATH, then hard coded paths to the executables will have to be used in all BFX jobs/scripts. This can become a significant maintenance problem for the users in the future. However if this how all your existing BFX jobs are setup you will have to add symbolic links from the current location to the path specified in your jobs.

Removing H801 (Linux) and H621 (AIX)

During RPM removal, any customized files and log files will not be deleted. Remove the software as super user with the command:

```
# yum erase Hxx1
```

or

```
# rpm –e Hxx1
```

Removing the NESi Public Key

To remove the NESi public key, as super user issue the command:

rpm –e gpg-pubkey-3d6b35d3-51bb5907

Post Installation of H801 (Linux) and H621 (AIX)

It is recommended that BFX executables be in the PATH:

Either /usr/share/nesi/bfx/bin is put in the PATH or symbolic links to BFX executables in any place which is already in the users' path:

```
ln -sf /usr/share/nesi/bfx/bin/bfxjs      /usr/bin/bfxjs
ln -sf /usr/share/nesi/bfx/bin/bfxsjs     /usr/bin/bfxsjs
ln -sf /usr/share/nesi/bfx/bin/bfxti      /usr/bin/bfxti
ln -sf /usr/share/nesi/bfx/bin/bfxtr      /usr/bin/bfxtr
ln -sf /usr/share/nesi/bfx/lib/bfxjobstart      /usr/bin/bfxjobstart
ln -sf /usr/share/nesi/bfx/bin/bfxsjsop   /usr/bin/bfxsjsop
```

Optional:

```
ln -sf /usr/share/nesi/bfx/bin/bfxrecv    /usr/bin/bfxrecv
ln -sf /usr/share/nesi/bfx/bin/bfxsend    /usr/bin/bfxsend
ln -sf /usr/share/nesi/bfx/bin/bfxstart   /usr/bin/bfxstart
ln -sf /usr/share/nesi/bfx/bin/bfxstop    /usr/bin/bfxstop
ln -sf /usr/share/nesi/bfx/bin/bfxtest    /usr/bin/bfxtest
```

Here is an example of a technique to accomplish this:

```
for id in bfxjs bfxrecv bfxsend bfxsjs bfxsjsop bfxstart bfxstop bfxtest
bfxti bfxtr; do
    ln -s /usr/share/nesi/bfx/bin/$id /usr/bin/$id;
done
```

It is also recommended that links for "old" BFX executable locations be created to allow existing BFS jobs to execute:

Symbolic links for "old" BFX executables. In this example, /usr/nesi/bfx/bin was the "old" location.

```
ln -sf /usr/share/nesi/bfx/bin/bfxjs      /usr/nesi/bfx/bin/bfxjs
ln -sf /usr/share/nesi/bfx/bin/bfxsjs     /usr/nesi/bfx/bin/bfxsjs
ln -sf /usr/share/nesi/bfx/bin/bfxti      /usr/nesi/bfx/bin/bfxti
ln -sf /usr/share/nesi/bfx/bin/bfxtr      /usr/nesi/bfx/bin/bfxtr
ln -sf /usr/share/nesi/bfx/lib/bfxjobstart
      /usr/nesi/bfx/bin/bfxjobstart
ln -sf /usr/share/nesi/bfx/bin/bfxsjsop   /usr/nesi/bfx/bin/bfxsjsop
```

Optional:

```
ln -sf /usr/share/nesi/bfx/bin/bfxrecv    /usr/nesi/bfx/bin/bfxrecv
ln -sf /usr/share/nesi/bfx/bin/bfxsend    /usr/nesi/bfx/bin/bfxsend
ln -sf /usr/share/nesi/bfx/bin/bfxstart   /usr/nesi/bfx/bin/bfxstart
ln -sf /usr/share/nesi/bfx/bin/bfxstop    /usr/nesi/bfx/bin/bfxstop
ln -sf /usr/share/nesi/bfx/bin/bfxtest    /usr/nesi/bfx/bin/bfxtest
```

Here is an example of a technique to accomplish this:

```
mkdir -p /usr/nesi/bfx/bin
for id in bfxjs bfxrecv bfxsend bfxsjs bfxsjsop bfxstart bfxstop bfxtest
bfxti bfxtr; do
    ln -s /usr/share/nesi/bfx/bin/$id /usr/nesi/bfx/bin/$id;
done
```

Create certificate and key if using BFXSJS

Refer to the "Secure Jobfile Transfer Configuration" section of this manual for more information.

Modify bfxsecparms.cfg, bfxjs.cfg, bfxti.cfg and bfxtr.cfg files

Refer to the "Commands and Parameters" section of this manual for more information.

Starting, Stopping & Verifying the BFX Installation

For Linux systems:

The service command should be used to stop & start BFX:

```
# service bfx stop
# service bfx start
# service bfx restart
```

The chkconfig command should be used to verify installation:

```
# chkconfig  --list  bfx
```

For AIX systems:

The startsrc command or SMIT should be used to start BFX:

```
# startsrc  -s  bfxjs
# startsrc  -s  bfxsjs
```

The stopsrc command or SMIT should be used to stop BFX:

```
# stopsrc  -s  bfxjs
# stopsrc  -s  bfxsjs
```

The lssrc & lsitab commands or SMIT can be used to verify installation:

```
# lssrc -S -s bfxjs
# lssrc -S -s bfxsjs

# lsitab bfxjs
# lsitab bfxsjs
```

Completion and Testing of Installation

The BFX installation can be verified by running a simple test script called `bfxtest`. BFXJS must already be running.

Now invoke the BFX test script:

```
BFX in the PATH:
# bfxtest

BFX not in the PATH:
# /usr/share/nesi/bfx/bin/bfxtest
```

The script will take over from this point. It will give you directions and prompts for any parameters it needs to complete a simple intra-host test. If for some reason the test fails, you should first look for a job file in the home directory of the user you used for the test. This job file will contain the probable cause for failure. If, however, no job file exists in the user's home directory, view the BFXJS log file (/usr/share/nesi/bfx/bfxjslog) and look for the failure there. If secure BFX is used, errors will be found in one of the following files: bfxsjs log, bfxsjs.stderr, bfxsjs.stdout

Appendix C. H691 (Solaris) Installation

Prerequisites

The following are prerequisites for installing H691 Solaris BFX:

- The privileges to create directories, load files from the distributed software media, create links, edit system startup, and system login file.
- An installed and operational H690IP NetEx product release 7.0.2 or greater.
- Bash must be installed to utilize the send, recv and test scripts.
- An installed OpenSSL 1.0.0 user runtime libraries available to BFX users. The following is an example you may want to use to point to the OpenSSL library:
  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/ssl/lib

Pre-Installation

The location of the executable programs from your previous installation of BFX will be required for the Installation section.

The previous installation of BFX will have to be removed prior to installation.

Accessing the UNIX BFX software distribution

The BFX software is available as a package which may be downloaded from NESi. Contact NESi Customer Support to request the download link.

Upgrading H691 (Solaris)

If you are upgrading an existing installation of BFX, it is strongly recommended that any running BFX process be stopped. If you are upgrading from a package older than H691IP 3.4.2, the package should be removed using the pkgrm command. Additionally if you have BFX startup procedures in the boot process you will need to remove it (i.e. inittab, rc scripts, SMF, etc.) Once these are done, you can proceed to the Software Installation.

To find the version of the installed package use the following command:

```
# pkginfo –l NESiBFX32
```

Removing H691 (Solaris)

If you are planning on re-installing H691, copy the any modified BFX files to a temporary location.

As super user, remove the software with the pkgrm command:

```
# pkgrm NESiBFX32
```

Software Installation

Please review all the installation instructions prior to performing the install.

If this is an initial installation, install the software as super user with the command:

```
# uncompress ./H691-3.4...pkg.Z
# pkgadd -d ./H691-3.4...pkg
```

Customers do NOT have the option of installing H691 into a user-defined directory location.

NOTE: If BFX or its executables are not in the PATH, then hard coded paths to the executables will have to be used in all BFX jobs/scripts. This can become a significant maintenance problem for the users in the future. However if this is how all your existing BFX jobs are setup you will have to add symbolic links from the current location to the path specified in your jobs.

Post Installation of H691 (Solaris)

It is recommended that BFX executables be in the PATH:

Either /usr/share/nesi/bfx/bin is put in the PATH or symbolic links to BFX executables in any place which is already in the users' path:

```
ln -sf /usr/share/nesi/bfx/bin/bfxjs      /usr/bin/bfxjs
ln -sf /usr/share/nesi/bfx/bin/bfxsjs     /usr/bin/bfxsjs
ln -sf /usr/share/nesi/bfx/bin/bfxti      /usr/bin/bfxti
ln -sf /usr/share/nesi/bfx/bin/bfxtr      /usr/bin/bfxtr
ln -sf /usr/share/nesi/bfx/lib/bfxjobstart      /usr/bin/bfxjobstart
ln -sf /usr/share/nesi/bfx/bin/bfxsjsop   /usr/bin/bfxsjsop
```

Optional:

```
ln -sf /usr/share/nesi/bfx/bin/bfxrecv    /usr/bin/bfxrecv
ln -sf /usr/share/nesi/bfx/bin/bfxsend    /usr/bin/bfxsend
ln -sf /usr/share/nesi/bfx/bin/bfxstart   /usr/bin/bfxstart
ln -sf /usr/share/nesi/bfx/bin/bfxstop    /usr/bin/bfxstop
ln -sf /usr/share/nesi/bfx/bin/bfxtest    /usr/bin/bfxtest
```

Here is an example of a technique to accomplish this:

```
for id in bfxjs bfxrecv bfxsend bfxsjs bfxsjsop bfxstart bfxstop bfxtest
bfxti bfxtr; do
    ln -s /usr/share/nesi/bfx/bin/$id /usr/bin/$id;
done
```

It is also recommended that links for "old" BFX executable locations be created to allow existing BFS jobs to execute:

Symbolic links for "old" BFX executables. In this example, /usr/nesi/bfx/bin was the "old" location.

```
ln -sf /usr/share/nesi/bfx/bin/bfxjs      /usr/nesi/bfx/bin/bfxjs
ln -sf /usr/share/nesi/bfx/bin/bfxsjs     /usr/nesi/bfx/bin/bfxsjs
ln -sf /usr/share/nesi/bfx/bin/bfxti      /usr/nesi/bfx/bin/bfxti
ln -sf /usr/share/nesi/bfx/bin/bfxtr      /usr/nesi/bfx/bin/bfxtr
ln -sf /usr/share/nesi/bfx/lib/bfxjobstart
      /usr/nesi/bfx/bin/bfxjobstart
ln -sf /usr/share/nesi/bfx/bin/bfxsjsop   /usr/nesi/bfx/bin/bfxsjsop
```

Optional:

```
ln -sf /usr/share/nesi/bfx/bin/bfxrecv    /usr/nesi/bfx/bin/bfxrecv
ln -sf /usr/share/nesi/bfx/bin/bfxsend    /usr/nesi/bfx/bin/bfxsend
ln -sf /usr/share/nesi/bfx/bin/bfxstart   /usr/nesi/bfx/bin/bfxstart
ln -sf /usr/share/nesi/bfx/bin/bfxstop    /usr/nesi/bfx/bin/bfxstop
ln -sf /usr/share/nesi/bfx/bin/bfxtest    /usr/nesi/bfx/bin/bfxtest
```

Here is an example of a technique to accomplish this:

```
mkdir -p /usr/nesi/bfx/bin
for id in bfxjs bfxrecv bfxsend bfxsjs bfxsjsop bfxstart bfxstop bfxtest
bfxti bfxtr; do
    ln -s /usr/share/nesi/bfx/bin/$id /usr/nesi/bfx/bin/$id;
done
```

Create certificate and key if using BFXSJS

Refer to the "Secure Jobfile Transfer Configuration" section of this manual for more information.

Modify bfxsecparms.cfg, bfxjs.cfg, bfxti.cfg and bfxtr.cfg files

Refer to the "Commands and Parameters" section of this manual for more information.

Starting, Stopping & Verifying the BFX Installation

If you are running an OS version older than Solaris 10, you will be using init script process for starting, stopping and status of NetEx/IP.

```
# /etc/init.d/bfx start
# /etc/init.d/bfx restart
# /etc/init.d/bfx stop
# /etc/init.d/bfx status
```

If you are using SMF, you will be using the following commands to start, stop and retrieve status of NetEx/IP:

```
# svcadm enable bfx
# svcadm restart bfx
# svcadm disable bfx
# svcs –l bfx
```

Completion and Testing of Installation

The BFX installation can be verified by running a simple test script called `bfxtest`. BFXJS must already be running.

Now invoke the BFX test script:

```
BFX in the PATH:
# bfxtest

BFX not in the PATH:
# /usr/share/nesi/bfx/bin/bfxtest
```

The script will take over from this point. It will give you directions and prompts for any parameters it needs to complete a simple intra-host test. If for some reason the test fails, you should first look for a job file in the home directory of the user you used for the test. This job file will contain the probable cause for failure. If, however, no job file exists in the user's home directory, view the BFXJS log file (/usr/share/nesi/bfx/bfxjslog) and look for the failure there. If secure BFX is used, errors will be found in one of the following files: bfxsjs.log, bfxsjs.stderr, bfxsjs.stdout