Secure Bulk File Transfer (BFX™) Utility

for UNIX Operating Systems

Release 1.3.2

Software Reference Manual

Revision Record

Revision	Description
1.0	Manual released. Corresponds to Secure BFX release 1.0.
1.2	Updates corresponding to release 1.2
1.3	Updates to the BFX messages (new, modified, delete unused)
1.3-1	More updates to messages; changes for SYSV and SYSTEMD
1.3.2	Updates for running multiple BFXJS programs

Address comments concerning this manual to:

Network Executive Software, Inc.
Attn: Publications Department
6450 Wedgwood Rd N Suite 103
Maple Grove, MN 55311
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-Hxx5-1.3.2) and title of the document with your comments.

Preface

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

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

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

"Appendix A. Secure BFX Error Messages" lists Secure BFX error messages.

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

Secure BFX uses Secure NetEx/IP but the reader does not need to understand Secure NetEx/IP to use this manual and Secure BFX.

Reference Material

The following manuals contain related information.

Number	Title and Description
MAN-REF-Hxx4	*Secure NetEx/IP Software Reference Manual*

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, Secure NetEx/IP, Secure BFX**
Hewlett-Packard Corporation	**HP**
The Open Group	**UNIX**
Linus Torvalds	**Linux**
IBM	**AIX**
Oracle	**Oracle Solaris**

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 supported Secure BFX products is explained in the appendices at the end of this manual, “Appendix B. H805 (Linux) and H625 (AIX) 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 Secure NetEx/IP software that dynamically converts the host data from one character set to another.

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.

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.

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

NetEx is a registered trademark of Network Executive Software.

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 Secure Bulk File Transfer (BFX™) utility is a software package designed to be used with Secure NetEx® software provided by Network Executive Software. Secure BFX and Secure NetEx/IP provide the capability of privately 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 Secure NetEx/IP products installed (Secure UNIX BFX requires the corresponding Secure NetEx/IP).

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

Secure BFX can be used to transfer files between different hosts that may require specialized record or data conversion. Secure BFX allows the use of Secure NetEx/IP code conversion.

Supported Configurations

Secure BFX uses the Secure NetEx/IP communications 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 Secure BFX and Secure NetEx/IP. Sending your job to your own BFXJS (does this.)

Sample Network Configuration

Secure BFX requires that copies of the Network Executive Secure BFX programs and user-written sets of commands that invoke Secure BFX reside in both the source and destination CPUs, as shown in Figure 1 on page 1. Secure 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).

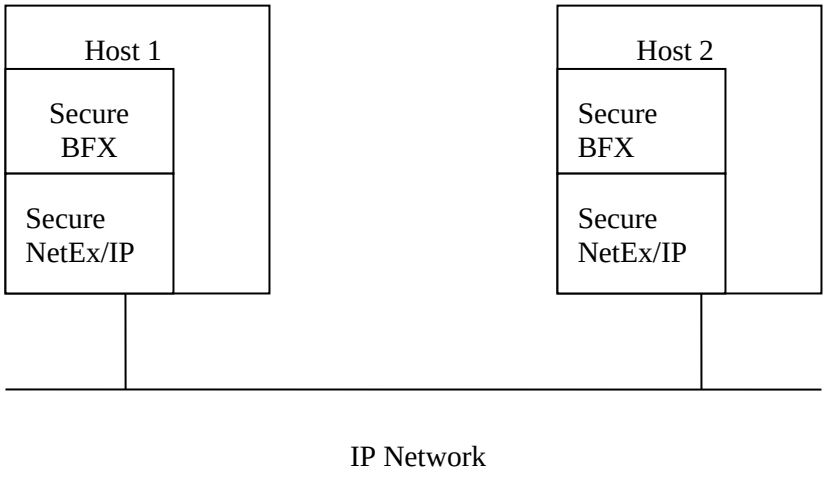

Figure 1. Sample Configuration Using Secure BFX

Figure 1 is a simple example showing the following points:

- Secure BFX and the Secure 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 Secure BFX and Secure NetEx/IP are installed and active. Secure BFX uses Secure NetEx/IP and software code conversion.
- Since the user directly invokes the Secure BFX utility, (that in turn calls Secure NetEx/IP), the Secure BFX user does not need to learn to use Secure NetEx/IP itself.
- Secure BFX is an application program that can be used like any other file-processing program. Normal users without regard to the other Secure BFX and Secure NetEx/IP applications that may be using sNetEx/IP at the same time can directly invoke it.

Three Secure BFX Programs

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

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

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

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

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

Using Secure BFX

To use Secure 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 Secure BFX processes the user requests.

Manual Job Submission

Manual job submission is the most basic application of Secure 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 Secure BFX commands for the host they will be executed on.

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

To begin the file transfer, an operator on one host submits the local command set and an operator on the other host submits the remote command set. When the jobs are executed, the BFXTR program connects to the BFXTI program (via Secure 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.

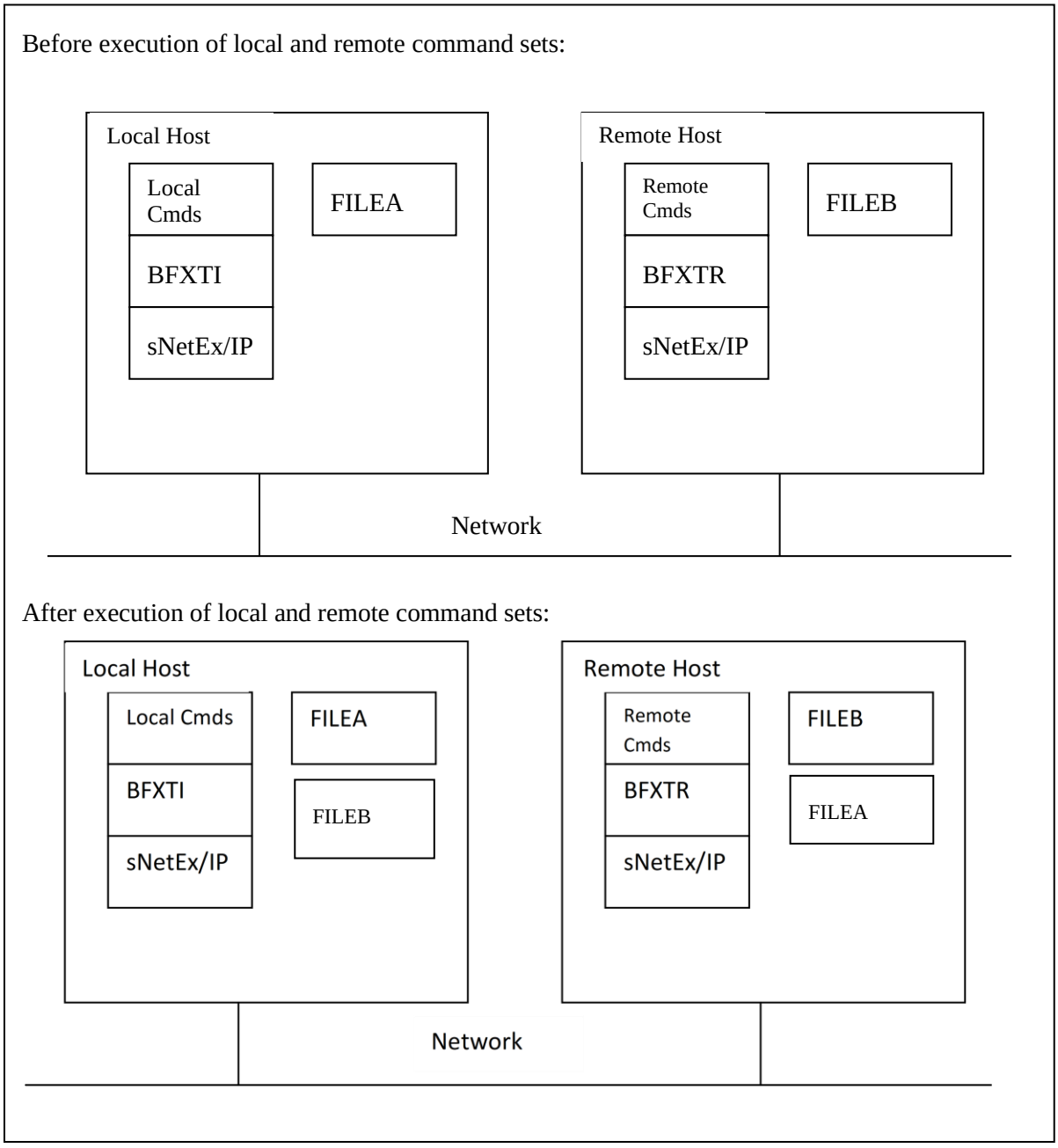

Figure 2. Manual Job Submission Example

Notice in Figure 2 above 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 Secure NetEx/IP.

Manual Job Submission Example – UNIX Specific

The following is a UNIX specific example of manual job submission. This example assumes that the Secure 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 Secure 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 Secure BFX. When using automatic job submission, the programmer must write local and remote command sets, which use commands suitable for each host. The command sets are similar to those used for manual job submission, using SEND and RECEIVE commands.

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

Secure Automatic Job Submission

If the job is to transfer securely, BFXTI opens a secure connection to BFXJS 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. In this case the Authority file is checked to see that the submitting user/Secure NetEx/IP hostname is authorized to submit a job on this system. If the user is not authorized, the job is not submitted and an error returned to BFXTI. Otherwise, the job is run and if the transfer is to be secure, the subsequent connection(s) opened on the SEND/RECEIVE is secure.

The Authorization file is configured by the system administrator and secured by file access rights to the user that starts BFXJS. The location of the Authority file is /usr/share/nesi/sbfx/bfxauth.cfg . Each line of the file contains an initiating Secure NetEx/IP Hostname and OS specific UserID separated by whitespace. (All trailing characters following the UserID are ignored.) The file is processed from the beginning to the end looking for a match of the initiating Hostname and the UserID submitting the job. If the Hostname or UserID specified in the file is an asterisk (*), it will match any string. Once a match for the Hostname and UserID is met, no other lines are inspected in the file and the job will be submitted. If no match is met, the job will not be submitted and an error is returned to BFXTI. (The Authority file is only inspected for Secure Automatic Job submission.)

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 Secure NetEx/IP connection with BFXJS on the remote host. BFXTI then transfers the remote job to BFXJS on the remote host.

After execution of the first part of the local command set, BFXJS submits the remote job to run BFXTR on the remote host. BFXTR then establishes a Secure 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 programs are ready for the next job.

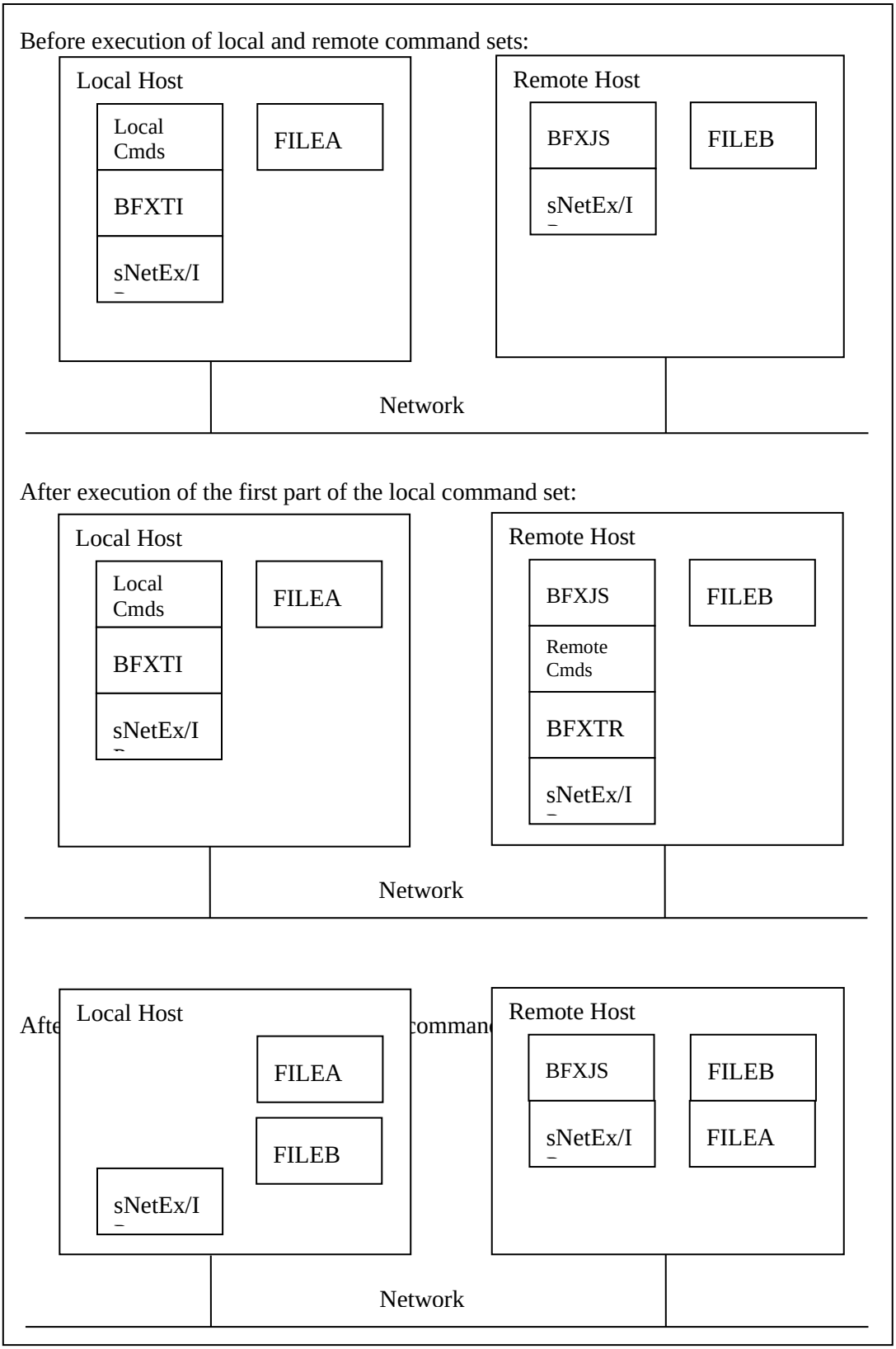

Figure 4. Automatic Job Submission Example

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` (as super-user) if it is not already running:

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

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

As a regular user, issue the following command:

```
# bfxti tijs
```

Several Secure 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. If the job submission is secured, the same checks will be enforced as for a secured automation job submission.

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

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

Remote Job Submission Example – Generalized

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

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

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

Figure 5. Remote Job Submission, General Example

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

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

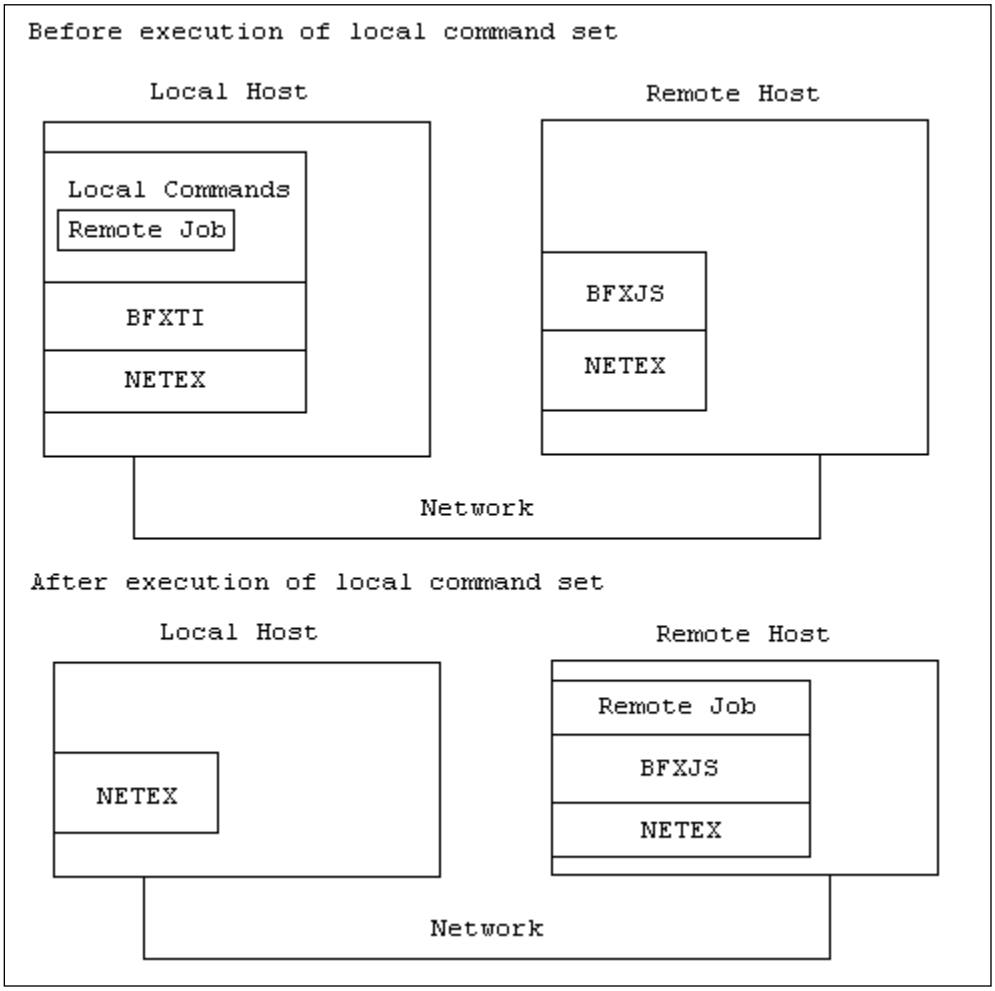

Figure 6. Remote Job Submission Example

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

Remote Job Submission Example – UNIX Specific

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

Create a file `jstest` containing the following commands:

```
# username
# password
#!/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


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

As a regular user, issue the following command:

```
$ bfxti tijstest
```

Several Secure 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 `bfxjs`, if it is not already running:

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

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

As a regular user, issue the following command:

```
$ bfxti tijstest
```

Several Secure 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.

Running Multiple BFXJS (example)

Update the BFXJS startup script to allow it to be used multiple times. The script is meant to be used as a sub-system executable program. Add parameters to allow passing of options.

```
usage: bfxstartjs [-i file] [-o file] [-e env]
```

Use "-i <file>" to specify a bfxinput file.
Default is /usr/share/nesi/sbfx/bfxinput if this option is not used.

Use "-o <file>" to specify a log file.
Default is /usr/share/nesi/sbfx/bfxjslog if this option is not used.

Use "-e <var>=<value>" to set an environment variable.
This can be used multiple times if necessary.

There is no change to the default instance of the sbfxjs subsystem.
The input/output files are optional and default to the normal locations.

In this example the new subsystem name is sbfxjs2 and will use the same configuration as the default instance. A different log file is used to keep messages separated.

1. Create a subsystem.

```
# mkssys -s sbfxjs2 -p /usr/share/nesi/sbfx/bin/bfxstartjs \
         -a "-o /var/log/bfxjslog2" -u 0 -Q -S -n 15 -f 9
```

2. Start the subsystem

```
# startsrc -s sbfxjs2
```

3. Allow the subsystem to start during system boot (optional).

```
# mkitab "sbfxjs2:2:wait:/usr/bin/startsrc -s sbfxjs2 >/dev/console 2>&1"
```

To specify an alternate configuration use a separate input file.

```
# echo 'idstamp timestamp msglvl 4 nojsec' > /usr/share/nesi/sbfx/sbfxjs-ns
# mkssys -s sbfxjs-ns -p /usr/share/nesi/sbfx/bin/bfxstartjs \
         -a "-i /usr/share/nesi/sbfx/sbfxjs-ns -o /var/log/bfxjslog-ns" \
         -u 0 -Q -S -n 15 -f 9
# startsrc -s sbfxjs-ns
```

For some customers an alternate Secure NetEx is sometimes required.

```
# mkssys -s sbfxjs-ibm -p /usr/share/nesi/sbfx/bin/bfxstartjs \
         -a "-o /var/log/bfxjslog-ibm -e SNETEX_SERVICE=sntx-ibm" \
         -u 0 -Q -S -n 15 -f 9
# startsrc -s sbfxjs-ibm
```

Remote Hostname Substitution Table

This optional table is contained in bfxmap.cfg file. This table defines a set of rules that may result in a new remote hostname being substituted for an existing remote hostname that is specified on a BFX SEND, RECEIVE or SUBMIT control statement, based on matching the BFX application ID (or ID mask). The first match terminates the search.

When using this feature, the customer must create the file and add hostname substitution entries to it prior to using it from Secure BFX jobs. The minimum size of each record must be sufficient to contain three table entries, with each entry being up to 8 characters. When Secure BFX jobs are subsequently executed, this file is accessed when the BFX SEND, RECEIVE and SUBMIT control statements are processed. Using this feature does not require any changes to be made to BFX job streams.

An example of this file can be seen in Figure 7 on page 14.

```
#
#  Remote Hostname Substitution Table
#
#  At the start of each BFX SEND/RECEIVE operation, this table is searched for
#  a matching BFX application ID (or ID mask).
#
#  If match is found for a SEND control statement, the TO host specified on
#  the SEND statement will be replaced by the hostname specified in the
#  TO-HOST column.
#
#  If a match is found for a RECEIVE control statement, the FROM host specified
#  on the RECEIVE statement will be replaced by the hostname specified in the
#  FROM-HOST column.
#
#  As soon as a match is found, the search terminates.
#
#  Matches on the BFX application ID will be successful for either an exact
#  character match, or by a combination of one or more exact character matches
#  and one or more percent characters (%).  Each % will match one and only one
#  other single character.  The total number of characters specified by the
#  BFX application ID is limited to 8.
#
#
#  Example:
#
#    BFX Applic ID   FROM-HOST    TO-HOST
#    -------------   ---------    -------
#       S1             HOST1        HOST2
#       ID1%           HOST5        HOST6
#       ID2%%%         HOST8        HOST9
#
#  For BFX Applic ID S1 (matches entry S1 in table):
#     BFX SEND    - HOST2 replaces the TO hostname
#     BFX RECEIVE - HOST1 replaces the FROM hostname
#
#  For BFX Applic ID ID1A (matches entry ID1% in table):
#     BFX SEND    - HOST6 replaces the TO hostname
#     BFX RECEIVE - HOST5 replaces the FROM hostname
#
#  For BFX Applic ID ID2ABC (matches entry ID2%%% in table):
#     BFX SEND    - HOST9 replaces the TO hostname
#     BFX RECEIVE - HOST8 replaces the FROM hostname
#
#  For BFX Applic ID ID2AC (does not match any entries in table)
#     BFX SEND    - no hostname substitutions
#     BFX RECEIVE - no hostname substitutions
#
#  Note: comments in this file are indicated by any of these special
#        characters in column 1:
#           # * ! " /
#
#  BFX Applic ID    FROM-HOST     TO-HOST
#  -------------    ---------     -------
#  (add desired entries following this line)
```

Figure 7. Remote Hostname Substitution Table

Data Modes

The Secure 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.

Secure 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 Secure BFX, and converted back to non-sequential format.

Using SECURE BFX

Secure BFX allows users to transmit the job files and data files across the IP network in an encrypted tunnel. This prevents user-ids, passwords and data from being transmitted in clear text. Once BFX is installed and properly configured, most jobs should run without any modifications. The user has the ability to specify which job files and data files are to be transmitted in an encrypted tunnel and which will not.

The user can set global defaults for the secure transfer parameters (BFXTICFG, BFXTRCFG, BFXJSCFG). Other BFX defaults can be included in this file also.

MJSEC	Job submission security can be lowered
NOMJSEC	Job submission security CANNOT be lowered
MDSEC	Data transfer security can be lowered
NOMDSEC	Data transfer security CANNOT be lowered
SECURE JSEC	Job transmissions should be encrypted
NOSECURE NOJSEC	Job transmissions should NOT be encrypted
DSEC	Data transmissions should be encrypted.
NODSEC	Data transmissions should NOT be encrypted
SECHOST nnnnnnn	All job transmissions to NetEx/IP hostname nnnnnnnn will be encrypted
NOSECHOST nnnnnnnn	All job transmissions to NetEx/IP hostname nnnnnnnn will NOT be encrypted
JSAUTH	Bfxjs authorizes user; bfxti sends user information to job submission
NOJSAUTH	Bfxjs will NOT authorize the user; bfxti does NOT send user information to job submission

By default, the programs default to secure job submission and secure data transmissions. These can be overridden by defaults in the global configuration file located within the Secure BFX load library (BFXTICFG, BFXTRCFG, BFXJSCFG). The security settings must match on the local and remote systems. If BFXJS is offered only in a non-secure mode, a request to transfer a job file securely will receive an error message stating BFXJS is not offered. The same is true for the reverse situation.

Secure transfer was designed to run with your existing jobs. It will use the defaults from global configuration files. It will also process the normal BFX parameter file. If JSEC was specified, the job file will be securely transmitted. If NOJSEC was specified, the job file would be sent on an unencrypted connection. DSEC would result in an encrypted connection for the data file. NODSEC would result in an unencrypted connection for the data file.

Security can only be increased; it cannot be lowered. If the global configuration files specified the JSEC parameter, users could not code a NOJSEC in the user input parameters. This would be a case of lowering security. If these files also included a NOSECHOST SYS1, then job transmissions to all hosts other than SYS1 would be secure. Job transmission to SYS1 would be non-secured. The user could add the JSEC parameter. This is raising the security level and would be honored. The same rules apply to the DSEC parameters. The SECHOST parameter would be used when the NODSEC parameter was used. All job files would be sent on an unencrypted connection, except to the hosts specified on the SECHOST statement.

A total of five SECHOST and five NOSECHOST maybe specified.

Additional checks SECURE job transmissions only.

Secure BFX is installed into two libraries. The first is for BFXTI and BFXTR. Theses have minimal system security requirements. Most users should be able to read or execute programs from this library. The other

library is for BFXJS. *MOST USERS SHOULD NOT HAVE ACCESS TO THIS LIBRARY*. This library contains the BFXJS program and the BFXAUTHCFG. The BFXAUTHCFG file is a list of Secure NetEx/IP host names and the userids that are allowed to submit jobs on this system. The table consists of two alphanumeric fields and any additional comments you may wish to include. The first field is the Secure NetEx/IP host name (maximum of 8 characters). These names should all be upper case. The second field is the remote userid that is allowed to submit jobs on this system. Userids are case sensitive. (i.e. userid "user1" will NOT match userid "USER1"). * * is the current default for the BXAUTHCFG file. This says any system with any userid may be used to submit jobs on this system. If the BFXAUTHCFG file consisted of two lines:

```
sys1 *
sys2 user1
```

Any userid on system SYS1 may submit a job on this system. On SYS2, only user1 is permitted to run jobs on this system. Any job submitted from SYS3 will be rejected with a BFX protocol error. This file maybe updated at any time, and updates are effective immediately.

Summary

Secure BFX is a system that consists of three separate programs:

- An executable image, BFXTI (BFX Transfer Initiate), that is started in the processor that wishes to either send or receive a file. 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 Secure 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 Secure 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 Secure 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 Secure BFX to provide a simple, high speed Remote Job Entry facility.
- Secure BFX is principally a batch utility for the transfer of sequential files between processors. There are three basic ways to use Secure BFX:

Automatic Job Submission

The initiating party sends a card image file to the BFXJS job submission program. The BFXJS program submits the file to the batch internal reader. The batch process, once started, invokes BFXTR to connect

back to the original initiating party. At that stage a file or series of files can be transferred between the initiator and the batch program. This technique is well suited for the transfer of very large files or for the delivery of a file to the initiator.

Remote Job Submission

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

Manual Job Submission

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

Commands and Parameters

This section describes the format and use of commands that execute the Secure 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 Secure 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 has 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 Secure BFX Commands

The commands accepted by the Secure 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.

Secure BFX Command Defaults

Some commands may only be issued to certain Secure BFX programs, such as BFXTI. Table 1 on page 20 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 Secure 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, and bfxjs.cfg. These files are processed by BFXTI, BFXTR, and BFXJS 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 Secure BFX Commands is done using the following syntax: bfxjs < bfxinput, or similar for bfxti and bfxtr.

Table 1. Secure BFX Command Defaults

Command	Default	Only Interpreted by
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	65535	BFXTI, BFXTR
DELAYTIME	5	BFXTI, BFXTR
DSEC	DSEC	BFXTI, BFXTR
FILE	BFXFILE	BFXTI, BFXTR (Must be defined in BFX job file)
FROM	NTXLCL	BFXTI, BFXTR
HOSTCHK	NOHOSTCHK	BFXTI
ID	"????????"	BFXTI, BFXTR
IDSTAMP	NOIDSTAMP	BFXTI, BFXTR, BFXJS
JBLOCK	65535	BFXTI, BFXJS
JID	BFXJS	BFXTI, BFXJS
JOBFILE	RMTJOB JOBxxxxxxx	BFXTI BFXJS(Must be defined in BFX job file)
JREPEATCONN	5	BFXTI
JRMAXL	240	BFXTI, BFXJS
JSAUTH	JSAUTH	BFXTI, BFXJS
JSEC	JSEC	BFXTI, BFXJS
MDSEC	MDSEC	BFXTI,BFXTR
MJSEC	MJSEC	BFXTI
MODE	CHARACTER	BFXTI, BFXTR (Must be defined in BFX job file)
MSGLVL	4	BFXTI, BFXTR, BFXJS
NEWHOST	None	BFXTI
NODSEC	DSEC	BFXTI, BFXTR
NOHOSTCHK	NOHOSTCHK	BFXTI
NOIDSTAMP	NOIDSTAMP	BFXTI, BFXTR, BFXJS
NOJSAUTH	JSAUTH	BFXTI, BFXJS
NOJSEC	JSEC	BFXTI, BFXJS

Command	Default	Only Interpreted by
NOMDSEC	MDSEC	BFXTI, BFXTR
NOMJSEC	MJSEC	BFXTI
NOSOE	NOSOE	BFXTI, BFXTR
NOSUBMIT	None	BFXTI
NOTIMESTAMP	NOTIMESTAMP	BFXTI, BFXTR, BFXJS
RECBUFF	0	BFXTI
REPEATCONN	20	BFXTR
RMAXL	1024	BFXTI, BFXTR, BFXJS
RPARM	(blanks)	BFXTI, BFXTR, BFXJS
SOE	NOSOE	BFXTI, BFXTR
TIMEOFFER	240	BFXTI, BFXJS
TIMEOUT	60	BFXTI, BFXTR, BFXJS
TIMESTAMP	NOTIMESTAMP	BFXTI, BFXTR, BFXJS
TO	NTXLCL	BFXTI, BFXTR

Command Descriptions

The following paragraphs describe the Secure BFX commands. The Secure BFX commands were summarized in Table 1 on page 20.

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 Secure 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 64K 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 Secure 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 27.

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 Secure 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 Secure 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.

IDSTAMP Command

The IDSTAMP command specifies that a process number (PID) is to be printed with all subsequent BFX messages. The PID will identify a specific BFX process that issued the message.

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

HOSTCHK | NOHOSTCHK

This command specifies that a completing BFXTI offer will check the BFXTR connecting host name, to make sure it matches the host name specified on the BFXTI 'SEND TO hostname' or 'RECEIVE FROM hostname' statement. If it does not match, the connection is rejected. The default is NOHOSTCHK for the H625 product.

This option should not be specified if group host names are used.

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 64KB, 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.

JSAUTH Command

If JSAUTH is used, a SECURE job submission is required to pass an authorized user check using the BFX auth file. (Default is JSAUTH.)

If JSAUTH is used, a SECURE job submission will add userid to the connect data sent to BFXJS. (Default is JSAUTH.)

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.

MDSEC/NOMDSEC

This parameter specifies if the user can lessen the security mode of the data transfer. If this parameter is not supplied the default is MDSEC.

MJSEC/NOMJSEC

This parameter specifies if the user can lessen the secure mode of the job submission. If this parameter is not supplied, the default is MJSEC.

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 Secure 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. 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 Secure BFX in detail and are intended only for use in diagnosing problem with Secure BFX, Secure 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.

16 Inhibit all error messages.

This command can be issued in any Secure BFX component.

NEWHOST Command

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

NOIDSTAMP Command

The NOIDSTAMP command reverses the effect of the IDSTAMP command. NOIDSTAMP specifies that no process number (PID) is to be printed with all subsequent BFX messages.

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

NOJSAUTH Command

If NOJSAUTH is used, a SECURE job submission is not checked for authorized user. (Default is JSAUTH.)

If NOJSAUTH is used, a SECURE job submission will not add userid to the connect data sent to BFXJS. (Default is JSAUTH.)

NODSEC Command

The NODSEC command specifies that for this file transfer, the data and associated Secure BFX protocol will NOT transfer securely. NODSEC cannot be set once DSEC and NOMDSEC are set (and the data will be sent securely).

NOJSEC Command

The NOJSEC command specifies that for this file transfer, the jobfile and the associated Secure BFX protocol will NOT transfer securely. NODSEC cannot be specified once DSEC has been set (and the jobfile will be sent securely).

NOSOE Command

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

NOSUBMIT Command

NOSUBMIT is used in cases where each half of the Secure 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.

NOTIMESTAMP Command

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

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

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 Secure BFX components. Default is 1024, minimum is 0 and maximum is 32767 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 Secure 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 Secure BFX products and only when the UNIX Secure BFX is on the **receiving** side of the file transfer.

Secure 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 Secure 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).

JSEC Command

The JSEC command specifies that for this file transfer, the jobfile and the associated Secure BFX protocol will transfer securely. This overrides any lower security setting in a configuration file.

DSEC Command

The DSEC command specifies that for this file transfer, the data and associated Secure BFX protocol will transfer securely. This overrides any lower security setting in a configuration file.

SOE Command

The Stop On Error (SOE) command causes Secure BFX to stop reading further commands if any one transmission has problems. 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 Secure 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 Secure 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 Secure BFX program should wait for a read through Secure 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 Secure BFX when there is a possibility that the sending Secure BFX will be experiencing long delays during file transfer. For example, if a multi-volume tape file is being sent, the sending Secure 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 Secure 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 Secure BFX components and takes no parameters.

Special Considerations

Debugging Secure NetEx/IP problems

Secure BFX has the ability to assist in debugging problems that may occur before during and after the transmission. This is activated by five new BFX parameters that have been defined to BFX.

snxoff	Default value. Do not display any Secure NetEx/IP messages to the print file.
snxinf	Display informational messages about the connection. This includes such things as the COMAPI that is been used, and ip address information, as examples.
snxerr	Displays errors that occurred during the transfer
snxtrc	Traces all NETEX nrbs, input, processing and completion status
snxdbg	Displays all debug messages.
snxall	Displays all levels.

These parameters can be coded in the user's BFX input file or added to the various global configuration files. When setting the above parameters with Secure NetEx/IP debug settings, the result will be a logical OR'ing of the settings. Refer to the user manual for the appropriate Secure NetEx/IP product for details on additional debug settings.

Transfer to Non-UNIX Computer Systems

The code supplied by NetEx 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 Secure BFX run before it is ready for direct use.

Secure 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 Secure BFX program can be greater than was specified by the user's BLOCK command.

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

It is important to note that each record sent between the Secure 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.

Secure BFX Send/Receive Sample Jobs

Shell scripts named ‘bfxsend’ and ‘bfxrecv’ have been provided in `/usr/share/nesi/sbfx/bin` as a Secure 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 to the current working directory which you must have write permission. 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. Secure BFX Error Messages

Secure 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 Secure BFX error code.

nnn This is the error number. The Secure 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
- **W** - warning messages
- **R** - recoverable error messages
- **C** - continuation message

message text This area displays the text of the message.

The following are the messages issued by Secure 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 Secure 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.

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.

BFX005I BFX GLOBAL configuration completed.
Severity: 7 (Information)
Explanation: The global configuration has completed
User Response: None

BFX006F "xxxxxxxx" NOT RECOGNIZED IN CONTROL STATEMENT.
Severity: 15 (Fatal Error)
Explanation: An input statement to the Secure BFX program contains a string that is not a recognized parameter. The string will follow the error message. Secure 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 re-submit the job.

BFX007W "xxxxxxxx" IS ONLY VALID FOR BFXTI JOBS – IGNORED.
Severity: 9 (Recoverable error)
Explanation: A parameter that is not applicable to the Secure 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 Secure 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 Secure BFX run proceeded as intended.

BFX011F ID= BFX IDENTIFIER OMITTED.
Severity: 15 (Fatal Error)
Explanation: The ID parameter which uniquely identifies the Secure 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.

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 re-submit 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 Secure BFX program has started processing the input statements for the indicated file transfer.
User Response: None.

BFX022F NETEX SYSTEMWIDE CAPACITY EXCEEDED.
Severity: 15 (Fatal error)
Explanation: During the process of establishing communications, Secure NetEx/IP component (snxmap) returned an indication that it cannot handle a new connection. Processing is terminated, as it is uncertain when the condition will clear up.
User Response: User Response: Retry the job at a later time.

BFX023F REMOTE BFX PROGRAM DID NOT START.
Severity: 15 (Fatal error)
Explanation: The corresponding Secure 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 Secure 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 Secure BFX program received an indication from Secure NetEx/IP API 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 Secure NetEx/IP 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. Re-submit 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.

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 host name specified is not on the network (not found by the name resolver). Processing is terminated, as data transfer is not possible.
User Response: The probable cause of this error is an erroneous HOST parameter. Other possibilities are the remote host name was changed, not specified in the name resolver or the name resolver unavailable. Correct the error and re-submit the job.

BFX030S NetEx ERROR: NRBSTAT = *ssss*, NRBIND = *iii*.
Severity: 12 (Severe Error)
Explanation: Secure NetEx/IP API has reported an error to the Secure BFX program that is not an intercepted condition. "ssss" is the four-digit NRB status code; "iii" is the event indication type. Processing is terminated, as the actual severity of the error is not known by the Secure BFX program.
User Response: Refer to Secure NetEx documentation to determine the cause of the error. Frequently this error will be caused by earlier, more comprehensible errors. If other Secure BFX error messages precede this one, take the corrective action suggested by those messages.

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, Secure NetEx/IP 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 re-submit 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 re-submit the job.

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 to a higher value.

BFX035I NetEx sdisc: NRBSTAT = *ssss*, NRBIND = *iii*.
Severity: 12 (Informational)
Explanation: Secure BFX has issued a SDISC. The status is reported.
User Response: None.

BFX038F Only the BFXJS program can offer ID=BFXJS
Severity: 12 (Fatal)
Explanation: Only the BFXJS program can offer the ID of BFXJS.
User Response: Change the program to BFXJS or select a new offer id.

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

BFX045F BFX PROTOCOL ERROR – PREMATURE END MESSAGE.
Severity: 15 (Fatal Error) BFXRCV in BFXJS, BFXTI, and BFXTR.
Explanation: The remote Secure 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 Secure 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 re-submit the job.

BFX055S SECHOST $27 ignored -- length > 8 -- number remains at $30.
Severity: 12 (Severe)

Explanation: A SECHOST parameter specified a hostname greater than 8. The SECHOST parameter is ignored.
User Response: Determine the correct hostname and correct the parameter if necessary

BFX056S SECHOST $27 ignored -- max of $30 reached.
Severity: 12 (Severe)
Explanation: More than 5 SECHOST parameters were specified. The max is 5. The SECHOST parameter is ignored.
User Response: Consider changing your default to JSEC and using NOSECHOST parameter to exclude the system(s) you do not want secure transfers to occur to.

BFX057S NOSECHOST $27 ignored -- length > 8 -- number remains at $31
Severity: 12 (Severe)
Explanation: A NOSECHOST parameter specified a hostname greater than 8. The NOSECHOST parameter is ignored.
User Response: Determine the correct hostname and correct the parameter if necessary.

BFX058S NOSECHOST $27 ignored -- max of $30 reached.
Severity: 12 (Severe)
Explanation: More than 5 NOSECHOST parameters were specified. The max is 5. The NOSECHOST parameter is ignored.
User Response: Consider changing your default to NOJSEC and using SECHOST parameter to exclude the system(s) you want secure transfers to occur to.

BFX059W NOJSEC cannot be specified once JSEC has been set – sent securely
Severity: 12 (Warning)
Explanation: Once a secure transfer has been determined, the user cannot specify NOJSEC. A secure transfer will be attempted.
User Response: Correct the file that requested the transfer to run securely. If a SECHOST parameter was coded in a global parameter, a NOSECURE host may under some conditions reset the transfer.

BFX060W NODSEC cannot be specified once DSEC has been set – sent securely
Severity: 12(Warning)
Explanation: Once a secure data transfer has been determined, the user cannot specify NODSEC.
User Response: Correct the file that requested the transfer to run securely or remove the NODSEC parameter.

BFX061I Connection will be secure.
Severity: 8 (Informational)
Explanation: A secure connection will be used transfer the file. .

User Response: None**BFX062I Connection will not be secure.**
Severity: 8 (Informational)
Explanation: A non-secure connection will be used transfer the file. .
User Response: None

BFX063F Authorized users file is not accessible.
Severity: 12 (Fatal)
Explanation: The authorization file is could not be accessed.
User Response: Validate the location of the file and insure BFXJS has sufficient permissions to access it.

BFX064F User "nnnnnnnn" is not authorized from host "mmmmmmmm".
Severity: 12 (Fatal)
Explanation: The userid displayed is not authorized to submit from the displayed host.
User Response: Request an update to the authorized user file or re-run the job under a different userid.

BFX065W MJSEC cannot be specified once NOMJSEC has been set.
Severity: 12 (Warning)
Explanation: Once modification of jobfile security is turned off, it cannot be turned on. No jobfile security can be further modified.
User Response: Correct the file that requested the jobfile security modification. If a SECHOST parameter was coded in a global parameter, a NOSECURE host may under some conditions reset the transfer.

BFX066W MDSEC cannot be specified once NOMDSEC has been set.
Severity: 12(Warning)
Explanation: Once modification of data security is turned off, it cannot be turned on. No data security can further be modified.
User Response: Correct the file that requested the data security modification.

BFX067F Authorized user is required.
Severity: 12(Failure)
Explanation: The current security settings require an Authorized user be sent in the protocol.
User Response: Contact the connecting host system admin to verify the BFX is configured properly and re-run the job.

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.

BFX071F NOSECHOST only valid in a global configuration file.
Severity: 12 (Fatal)
Explanation: NOSECHOST is only valid in the BFXTICFG file.
User Response: Remove the NOSECHOST parameter and restart the job.

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

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

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

BFX083S FILE fffffffffff ABORT PROCESSED.
Severity: 12 (Informational)
Explanation: An error occurred on the remote host that stopped the transfer from continuing. The local file fffffffffff was successfully closed.
User Response: Correct the error that caused the transfer to stop. Previous log messages will explain the error.

BFX084F PROTOCOL ERROR – DATA RECEIVED WHEN SENDING.
Severity: 15 (Fatal Error)
Explanation: Unexpected data was received when sending.
User Response: This is generally caused by incorrect user-written block and/or record modules. Re-code the user module to correct the problem or contact support@netex.com

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

BFX088S OFFER OF hhhhhhhh FAILED.
Severity: 12 (Severe Error)
Explanation: Secure BFX could not offer. This is generally caused by insufficient privileges or Secure NetEx/IP components not present.
User Response: Verify privileges and the presence of the Secure NetEx/IP components (API, snxmap) and try again.

BFX089S CONNECT TO hhhhhhhh FAILED.
Severity: 12 (Severe Error)
Explanation: Secure 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.

BFX090S Host Check failed; Host Connected ssssssss; Host Requested hhhhhhhh.
Severity: 12 (Severe error).
Explanation: The parameter HOSTCHECK was active. The host requested was specified in the Secure BFX job parameters. The host that connected did not match the specified host. The run is terminated/

User Response: If the host coded in the BFX job parameters is a group name, the connected hostname is returned so HOSTCHECK must be turned off.

BFX101I FILE ffffffff DONE; nnnn RECORDS SENT.
Severity: 6 (Informational)
Explanation: The standard Sending Record Module has detected normal end of file on the input file specified by DDNAME "ffffffff". The total number of 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 Secure BFX.
User Response: None.

BFX102S File ffffffff Permanent I/O error, RC=nn.
Severity: 12 (Severe error**)**
Explanation: A permanent I/O error was detected while reading or writing a file during the transfer of the job or of the file. The return code is in octal. If batched transfer of files is being performed, Secure BFX will attempt to transfer the rest of the specified files.
User Response: Determine the cause of the I/O error. If the error can be corrected, re-run the BFX jobs.

BFX103S Cannot find Record Module mmmmmmmm
Severity: 12 (Severe error)
Explanation: The Record Module or Block Module specified **was not installed in this copy** of the Secure BFX program. Transfer of this file is aborted; if batched transfer of files is being performed, Secure BFX will attempt to transfer the rest of the specified files.
User Response: This can be caused because the module does not exist or use of the incorrect module identifier name.

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.

BFX107S Cannot find Job Submission Module fffffffff.
Severity: 12 (Severe Error)
Explanation: ??????
User Response: ??????

BFX111S Block Module Initialization failed.
Severity: 12 (Severe error)
Explanation: The Record Module returned an Abort code (BUFLEV= 16) when called at initialization. The module did not supply a message to go with the abort, so this default message is printed. Transfer of this file is aborted; if batched execution is being used, Secure BFX will attempt to transfer the subsequent files.
User Response: Correct the condition in the Record Module.

BFX113S Sending Record Module aborted transfer..
Severity: 12 (Severe error)
Explanation: During the transfer of a file, the Sending Record Module returned an Abort transfer indication (BUFLEV= 16.) The user module did not supply a message with the abort code, so this default message is printed. Transfer of this file is aborted; if batched execution is in use, Secure BFX will attempt to transfer the remaining files.
User Response: Correct the error generated by or detected by the module and re-submit the jobs.

BFX114S Receiving Record Module aborted transfer.
Severity: 12 (Severe error)
Explanation: During the transfer of a file, the Receiving Record Module returned an Abort transfer indication (BUFLEV= 16.) The user module did not supply a message with the abort code, so this default message is printed. Transfer of this file is aborted; if batched execution is in use, Secure BFX will attempt to transfer the remaining files.
User Response: Correct the error generated by or detected by the module and re-submit the jobs.

BFX115S Sending Block Module aborted transfer.
Severity: 12 (Severe error)
Explanation: During the transfer of a file, the Sending Block Module returned an Abort transfer indication (BUFLEV= 16.) The user module did not supply a message with the abort code, so this default message is printed. Transfer of this file is aborted; if batched execution is in use, Secure BFX will attempt to transfer the remaining files.
User Response: Correct the error generated by or detected by the module and re-submit the jobs.

BFX116S Receiving Block Module aborted transfer.
Severity: 12 (Severe error)
Explanation: During the transfer of a file, the Receiving Block Module returned an Abort transfer indication (BUFLEV= 16.) The user module did not supply a message with the abort code, so this default message is printed. Transfer of this file is aborted; if batched execution is in use, Secure BFX will attempt to transfer the remaining files.
User Response: Correct the error generated by or detected by the module and re-submit the job

BFX118S Sending Block Module abort processed.
Severity: 12 (Severe Error)
Explanation: Due to a loss of communications or an error condition reported by the BFX program, the sending block module was called with the abort condition (BUFLEV= 16). Upon return, the user module did not supply a message to acknowledge the abort, so this default message is printed. Processing will continue as indicated by the messages that preceded this one.
User Response: Correct the condition indicated in the messages previous to this one.

BFX120S Receiving Block Module abort processed.
Severity: 12 (Severe Error)
Explanation: Due to a loss of communications or an error condition reported by the BFX program, the receiving block module was called with the abort condition (BUFLEV= 16). Upon return, the user module did not supply a message to acknowledge the abort, so this default message is printed. Processing will continue as indicated by the messages that preceded this one.
User Response: Correct the condition indicated in the messages previous to this one.

BFX121S MAXIMUM RECORD LENGTH EXCEEDS NEGOTIATED SIZE.
Severity: 12 (Severe Error)
Explanation: When the two Secure 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 Secure 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.

BFX123F FILE TRANSFER PROTOCOL SEQUENCE ERROR.
Severity: 15 (Fatal error)
Explanation: The file transfer information sent to the receiving Secure 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 Secure BFX or Secure NetEx/IP error. The current transfer is aborted, but the remaining transfer will be attempted.
User Response: Bring the error to the immediate attention of NetEx Customer Support.

BFX124S MODE= PARAMETERS NOT CONSISTENT FOR BOTH BFX JOBS.
Severity: 12 (Severe error) BFXSCN in BFXTI and BFXTR.
Explanation: In a Secure 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 Secure BFX buffers. The size of the buffers is determined when the Secure BFX programs are compiled and linked. Normally, these buffers are 64KB long (the largest block size allowed by Secure NetEx/IP). 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 64KB long, it cannot be transferred. If the longest record in the file is less than 64KB long, this error indicates that the Secure 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 Secure 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.

BFX203E FILE ffffffff TRANSFER ABORTED; nnnnnnnn RECORDS RECEIVED.
Severity: 10 (Error)
Explanation: This message is issued by the receiving Secure BFX when the file transfer process is aborted either because of the loss of Secure NetEx/IP communication or because of some other error detected by the Secure 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 Secure BFX messages.
User Response: Correct the error that caused the abort. Transfer the file again.

BFX206E FILE ffffffff TRANSFER ABORTED; nnnnnnnn RECORDS SENT.
Severity: 10 (Error)
Explanation: This message is issued by the sending Secure BFX when the file transfer process is aborted either because of the loss of Secure NetEx/IP communication or because of some other error detected by the Secure 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 Secure 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 Secure 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 Secure 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 fffffffffff.
Severity: 4 (Informational)
Explanation: The sending Secure 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. The name of the file is fffffffffff.
Response: None.

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

BFX301I OFFERING sssssss; BLOCK IN SIZE bbbb.
Severity: 2 (Diagnostic).
Explanation: The Secure BFX program has issued a Secure NetEx/IP 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 sssssss ON HOST hhhhhhhh; BLOCK IN SIZE nnnn.
Severity: 2 (Diagnostic).
Explanation: When BFXTR is connecting back to its starting BFXTI, it has issued a Secure NetEx/IP 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.

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

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

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

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

BFX310I CONNECT CONFIRM READ ISSUED.
Severity: 2 (Diagnostic).
Explanation: Following a successful SCONNECT request, the Secure 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 Secure BFX program has completed successfully. The Secure NetEx/IP 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 Secure BFX program when the session negotiation process is complete. "bbbbb" is the Secure NetEx/IP block size that will be used during file transfer. "dddd" is four hexadecimal digits that give the Secure NetEx/IP DATAMODE to be used based on the requirements of the two Secure 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.

BFX319I Hostname $$$$$$$$ mapped to $$$$$$$$
Severity: 9 (Informative)
Explanation: A BFX application ID (or ID mask) was specified on the SEND/RECEIVE control statement. A match was found in the Remote Hostname Substitution Table, and the "mapped to" hostname was used for the connection.
User Response: None.

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 error number returned from the READ statement that attempted to decode the value will follow the message. Secure 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 re-submit 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). Secure 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 re-submit 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. Secure 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 re-submit the job.

BFX407E Parameter value xxxxxxxx too long.
Severity: 9 (Error).
Explanation: A string value specified in an input statement to be assigned to a parameter was longer than the parameter field itself. The string is truncated to fit in the field. Processing will continue normally.
User Response: Unexpected results may occur because of this truncation. If so, supply a valid length string and re-submit 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.

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

BFX413I CLOSE/NOCLOSE parameter has been deprecated – Ignored.
Severity: 15 (Informational).
Explanation: The input parameters included a CLOSE or NOCLOSE statement. These parameters are no longer supported. They are ignored.
User Response: Remove the parameters from the run stream.

BFX414I RATE parameter has been deprecated -- Ignored.
Severity: 15 (Informational).
Explanation: The input parameters included a RATE statement. This parameter is no longer supported. The parameter is ignored.
User Response: Remove the parameters from the run stream.

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

BFX900I Hxx1 b.r dddd.
Severity: 15 (Informational – always issued).
Explanation: Secure BFX sign-on line. Indicates the Secure BFX product's release level and build date.
User Response: None.

BFX902S COMMAND INPUT FILE 'filename' CANNOT BE FOUND
Severity: 15 (Severe Error)
Explanation: Secure 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: Secure BFX failed to allocate physical memory for data or message buffer space.
User Response: Retry the operation.

BFX904S Error: CHAR mode maximum record length exceeds xxxxx bytes.
Severity: 15 (Severe error)
Explanation: The maximum value of RMAXL is either 9999 or 65535 in CHARACTER mode. A record was read that exceeded this during a CHARACTER mode send. This will be 9999 or 65535 dependent on BFX support of hex record length format.
User Response: Change the mode to BIT or modify the file to have shorter records.

BFX905I Defaults input file cannot be found.
Severity: 4 (Informational)
Explanation: The defaults file for BFXTI (bfxti.cfg), BFXTR (bfxtr.cfg), or BFXJS (bfxjs.cfg) cannot be found. These files may be created in order to override the default programmed parameters or substitute for command line parameters.
User Response: This is an informational message. A default parameter file is not necessary.

BFX922I Calling File Open.
Severity: 4 (Informational)
Explanation: A system call to open a file will be issued. This can be used to track possible issues with filesystem access.
User Response: This is an informational message.

BFX923I Returned from File Open – elapsed time ss.nnnnnn.
Severity: 4 (Informational)
Explanation: A system call to open a file has returned. The elapsed time for the open to complete is given in seconds (ss) and nanoseconds (nnnnnn) . This can be used to track possible issues with filesystem access.
User Response: This is an informational message.

BFX942I Using the record module: *nnnnnnn*.
Severity: 15 (Informational)
Explanation: ***nnnnnnnn*** is the name of the record module being used to process the records.
User Response: None.

Appendix B. H805 (Linux) and H625 (AIX) Installation

Prerequisites

The following are prerequisites for installing H805 and H625 Secure 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 Hxx4 Secure NetEx/IP product for the platform.

Pre-Installation

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

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

Accessing the UNIX Secure BFX software distribution

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

Getting the NetEx Public Key

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

You can download the NetEx public key from the H805 document page on the http://www.netex.com/support website.

Importing the NetEx 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 Hxx5-1.0. . ..rpm
```

Installation

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

A link to the distribution can be obtained from NetEx 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 Hxx5-1.0...rpm
```

or

```
# rpm –i Hxx5-1.0...rpm
```

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

```
# yum –-nogpgcheck install Hxx5-1.0...rpm
```

or

```
# rpm –i –-nosignature Hxx5-1.0...rpm
```

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

Customers do NOT have the option of installing Hxx5 into a user-defined directory location.

NOTE: If Secure 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 could add symbolic links from the current location to the path specified in your jobs.

Upgrading H805 (Linux) and H625 (AIX)

If you are upgrading an existing installation of Secure BFX, it is strongly recommended that any running Secure NetEx/IP/IP and Secure 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 Hxx5-1.0...rpm
```

or

```
# rpm –U Hxx5-1.0...rpm
```

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

```
# yum –-nogpgcheck upgrade Hxx5-1.0...rpm
```

or

```
# rpm –U –-nosignature Hxx5-1.0...rpm
```

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

Customers do NOT have the option of installing Hxx5 into a user-defined directory location.

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

Removing H805 (Linux) and H625 (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 Hxx5
```

or

```
# rpm –e Hxx5
```

Removing the NetEx Public Key

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

rpm –e gpg-pubkey-3d6b35d3-51bb5907

Post Installation of H805 (Linux) and H625 (AIX)

It is recommended that Secure BFX executables be in the PATH:

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

```
ln -sf /usr/share/nesi/sbfx/bin/bfxti     /usr/bin/bfxti
ln -sf /usr/share/nesi/sbfx/bin/bfxtr     /usr/bin/bfxtr
```

Optional:

```
ln -sf /usr/share/nesi/sbfx/bin/bfxrecv   /usr/bin/bfxrecv
ln -sf /usr/share/nesi/sbfx/bin/bfxsend   /usr/bin/bfxsend
ln -sf /usr/share/nesi/sbfx/bin/bfxtest   /usr/bin/bfxtest
```

Here is an example of a technique to accomplish this:

```
for id in bfxrecv bfxsend bfxtest bfxti bfxtr; do
    ln -s /usr/share/nesi/sbfx/bin/$id /usr/bin/$id;
done
```

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

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

```
ln -sf /usr/share/nesi/sbfx/bin/bfxti     /usr/nesi/bfx/bin/bfxti
ln -sf /usr/share/nesi/sbfx/bin/bfxtr     /usr/nesi/bfx/bin/bfxtr
```

Optional:

```
ln -sf /usr/share/nesi/sbfx/bin/bfxrecv   /usr/nesi/bfx/bin/bfxrecv
ln -sf /usr/share/nesi/sbfx/bin/bfxsend   /usr/nesi/bfx/bin/bfxsend
ln -sf /usr/share/nesi/sbfx/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 bfxrecv bfxsend bfxtest bfxti bfxtr; do
    ln -s /usr/share/nesi/sbfx/bin/$id /usr/nesi/bfx/bin/$id;
done
```

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

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

Modify BFX Authorized User file – bfxauth.cfg

The Authorization file is configured by the system administrator and secured by file access rights to the user that starts BFXJS. The location of the Authority file is /usr/share/nesi/sbfx/bfxauth.cfg . Each line of the file contains an initiating Secure NetEx/IP Hostname and OS specific UserID separated by whitespace. (All trailing characters following the UserID are ignored.) The file is processed from the beginning to the end looking for a match of the initiating Hostname and the UserID submitting the job. If the Hostname or UserID specified in the file is an asterisk (*), it will match any string. Once a match for the Hostname and UserID is met, no other lines are inspected in the file and the job will be submitted. If no match is met, the job will not be submitted and an error is returned to BFXTI. (The Authority file is only inspected for Secure Automatic Job submission and when the JSAUTH is set in the configuration file.)

Starting, Stopping & Verifying the Secure BFX Installation

For Unix/Linux systems (SysV Init):

The service command should be used to stop, start or restart sbfxjs:

```
# service sbfxjs stop
# service sbfxjs start
# service sbfxjs restart
```

The chkconfig command should be used to verify installation:

```
# chkconfig  --list  sbfxjs
```

```
The BFXJS log can be found in /usr/share/nesi/sbfx/bfxjslog
```

For Unix/Linux systems (Systemd):

The systemctl command should be used to stop, start, restart and verify SBFX:

```
# systemctl stop sbfxjs.service
# systemctl start sbfxjs.service
# systemctl restart sbfxjs.service
# systemctl status sbfxjs.service
```

The sbfxjs log is managed by the systemd journal. To view the log use:

```
# journalctl -u sbfxjs
```

For AIX systems:

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

```
# startsrc  -s  sbfxjs
```

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

```
# stopsrc  -s  sbfxjs
```

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

```
# lssrc -S -s sbfxjs

# lsitab sbfxjs
```

Completion and Testing of Installation

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

Now invoke the Secure BFX test script:

```
Secure BFX in the PATH:
# bfxtest

Secure BFX not in the PATH:
# /usr/share/nesi/sbfx/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/sbfx/bfxjslog or journalctl -u sbfxjs) and look for the failure there.