H305 Secure Bulk File Transfer (BFX™) Utility

for Unisys OS2200 on Dorado Systems

Release 1.3.7

Software Reference Manual

Revision Record

Revision*	Description
1.0	Initial Extended Mode Secure BFX product release
1.1	Maintenance update aligned with Release 1.1
1.2	Maintenance update aligned with Release 1.2
1.3	Remote host substitution and maintenance
1.3.1	Enhancements to Tape Usermods to improve performance
1.3.3	Clarification of HOCHECK function in manual
1.3.5	Add messages TSB613/TSB614
1.3.7	Add new messages TSB607I, BFX415S, BFX416S, BFX944I and PAD parameter

You may submit written comments to:

Network Executive Software, Inc.
Publications Department
6450 Wedgwood Road N, Suite 103
Maple Grove, MN 55311
USA

Comments may also be submitted over the Internet by addressing e-mail to:

support@netex.com

or, by visiting our web site at:

http://www.netex.com

Always include the complete title of the document with your comments.

Preface

This manual describes the user interface to the Network Executive Software's H305 Extended Mode Secure Bulk File Transfer (BFX™) utility for Unisys OS2200 on Dorado systems. Secure BFX is used in conjunction with Network Executive Software's Secure NETwork EXecutive (NETEX ®) family of software products for use on IP networks. *The Secure Extended Mode products (H304/H305) do not interoperate with the existing or H300IPc/H301, or the H300e/H301e products, although they are user interface compatible.*

The first section of this manual, "Introduction" on page 1 is an introduction to Secure BFX. It includes a description of Secure BFX, network configurations that can support Secure BFX, and the three programs which compose Secure BFX and how they interact.

"ECL and Control Parameters for " on page 13 presents user control statements and parameters. This section also shows the ECL used when running Secure BFX, including simple examples.

The third section, "Applications" on page 31, provides detailed examples using Secure BFX.

The next section, "Installing " on page 35, describes the installation of Secure BFX. It includes defining default values for control parameters described in the second section.

"Appendix A. BFX Internal Summary" on page 39 describes the Secure BFX internal structure as it relates to writing user block (or record) modules.

"Appendix B. Secure BFX Error Messages" on page 53 contains Secure BFX error messages.

Secure BFX uses Secure NetEx/IP, but the reader does not need to understand Secure NetEx/IP to use the first four sections of this manual and Secure BFX.

Reference Material

The following manuals contain related information.

Number	Title and Description
MAN-REF-Hxx4	Hxx4 Secure NetEx/IP®

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 that are not described in this publication and 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™, PFX™, Secure NetEx/IP /IP, Secure BFX
Unisys Corp.	OS2200, Dorado

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

Comments may be submitted over the Internet by addressing email to:

support@netex.com

or, by visiting our web site at:

http://www.netex.com

Always include the complete title of the document with your comments.

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

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

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

code conversion: An optional feature in Secure NetEx/IP that dynamically converts the host data from one character set to another. Secure NetEx/IP with the code conversion has a special table that is used for code conversion and can be loaded with any type of code (for example, ASCII, EBCDIC, et cetera).

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.

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

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.

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

Revision Record ..... **ii**

Preface ..... **iii**

Reference Material ..... **v**

Notice to the Reader ..... **vii**
Notice to the Customer vii
Document Conventions viii
Glossary ix

Contents ..... **xi**
Figures xiv
Tables xv

Introduction **1**
Supported Configurations 1
Sample Network Configuration 2
Secure BFX Component Programs 2
Data Generating Modules 3

Using BFX **5**
Manual Job Submission 5
Automatic Job Submission 7
Secure Automatic Job Submission 9
Remote Job Submission 9
Remote Job Submission Example 9
Remote Hostname Substitution Table 10
Data Modes 12
Security 12
File Size 12

ECL and Control Parameters for Secure BFX **13**
ECL For Executing the BFXTI and BFXTR Programs 13
Secure BFX Execution Parameter Syntax and Placement 14
Using SECURE BFX 14
Additional checks SECURE job transmissions only 15
BFX Execution Parameters 15
General Information about Secure BFX Commands 15
BFX Component Commands and Parameters 16
Control Statements 19
Control Statement Parameters 19
COMMENTS 19
ID 19
BLOCK 20
BIGBLK 20

CONNDELAY/DELAYTIME 20
CONNMAX 20
CREOV 20
DSEC 20
DSEOV 20
DSLAB=n 20
FILE = *filename* 21
FILES=nnn|* 21
HOBCHECK 21
HOSTCHK | NOHOSTCHK 21
IDSTAMP 21
JBLOCK 22
JID 22
JOBFILE 22
JREPEATCONN 22
JRMAXL 22
JSAUTH 22
KEYIN 22
LETLABEL 22
MDSEC | NOMDSEC 22
MJSEC | NOMJSEC 23
MODE 23
MSGLVL 23
NEWHOST 24
NONSDF 24
NEWHOST 24
NODSEC 24
NOIDSTAMP 24
NOJSAUTH 24
NOJSEC 24
NOSUBMIT 25
NOSECURE 25
PAD 25
RECBUFF 25
REPEATCONN 25
RMAXL 25
RBM 25
RMOD 25
JSEC/SECURE 25
SOE | NOSOE 26
SPERRY 26
TRACKIO 26
LABELS 26
TIMEOF 26
TIMEOU 27
TIMESTAMP | NOTIMESTAMP 27
UNPACK 27
.EOT 27
.END 27
BFX Example 28
Special Considerations 29

Debugging Secure NetEx/IP problems 29
Record Formats and Record Type Conversion 29

Applications 31
Unisys OS2200 to Linux 32
Unisys OS2200 to IBM 33

Installing Secure BFX 35
Prerequisites for Installation 35
Release Distribution 35
Installation Process 36
Obtain the BFX distribution file. 36
FTP the distribution file to OS2200. 36
Check for required updates. 36
Upgrading H305 36
Removing H305 36
Software Installation 36
Modify BFX Authorized User file – BFXAUTHCFG 37
Execute the BFXJS Program 38
Verify the Secure BFX Installation (BFXJS needed) 38

Appendix A. BFX Internal Summary 39
Block Diagram 42
BFX Module Descriptions 44
BFX Receive File Control (RCV) 44
BFX Send File Control (SND) 47
Receiving Block Module (RBM) 50
Sending Block Module (SBM) 50
Receiving Record Module (RRM) 51
Sending Record Module (SRM) 51

Appendix B. Secure BFX Error Messages 53

Appendix C. SSL TRACING 79

Figures

Figure 1. Sample Secure NetEx/BFX Network 2
Figure 2. Host UNI1 manual job submission for initiating BFXTI job 6
Figure 3. Host UNI2 manual job submission for responding BFXTR job 6
Figure 4. BFXTI Automatic Job Submission 8
Figure 5. Remote Job Submission 10
Figure 6. Remote Hostname Substitution Table 11
Figure 7. BFXTI / BFXTR example ECL 13
Figure 8. BFX Command and Parameter List 18
Figure 9. Typical BFXTI Run Stream 28
Figure 10. Example Unisys initiated BFX file transfer from Linux to Unisys 32
Figure 11. Unisys initiated BFX file transfer from Unisys to OS2200 33
Figure 12. Sample BFXJS ECL 38
Figure 13. BFXTI Module Block Diagram 42
Figure 14. BFXTR Block Diagram 43
Figure 15. BFXJS Module Block Diagram 44
Figure 16. File Receive Data Flow 46
Figure 17. Send Receive Initialization Flow 48
Figure 18. File Send Data Flow 49

Tables

Table 1. BFX Default modules 39

Introduction

The Secure Bulk File Transfer (BFX™) utility is a software package designed to be used with Secure NetEx® software provided by Network Executive Software, Inc. Secure BFX and Secure NetEx/IP provide the software to rapidly move large amounts of sequential file data between processors. The processors may use different operating systems or be of a different manufacturer; provided they have an IP network and the proper Secure NetEx/IP products installed (H305 BFX requires H304).

Secure BFX uses batch processing facilities to perform file transfers. As a result, it has all the options of a batch program to access data base systems or other media supported by the OS2200 operating system. Also, as a batch utility, Secure BFX can be run as a batch job, from a terminal, or as a started job.

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

Supported Configurations

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

- Local and wide area IP networks.
- Intra-host processing allows application testing using just the local host computer, exercising the software capabilities of 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. Sample Secure NetEx/BFX Network on page 2. Secure BFX may also be used in a way that allows the user to place all of the commands in one of the hosts (Secure BFX must reside in both).

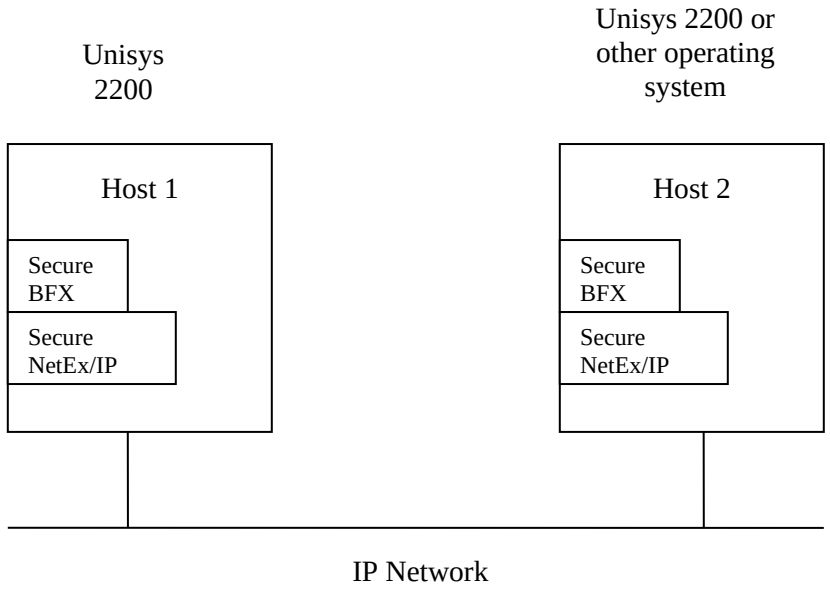

Figure 1. **Sample Secure NetEx/BFX Network**

Figure 1 is a simple example and demonstrates 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 Secure NetEx/IP at the same time can directly invoke it.

Secure BFX Component Programs

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

The BFXTI program runs in the processor that initiates the transfer request. BFXTI processes file transfer requests on the initiating system and will optionally send the job control statements for the BFXTR job to the responding system's BFXJS.

The BFXTR program runs in the processor that responds to the transfer request. BFXTR begins execution by connecting back to the BFXTI program on the initiating system.

The BFXJS program also runs in the processor that responds to the transfer request. BFXJS receives jobs transferred to the responding system and initiates them on its host system.

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

Data Generating Modules

Record Module BFXPERF contains two functions that send or receive data. The BFXGEN function generates data in records from 1 to 65535 bytes in length and transfers them to BFX. The BFXEAT function accepts records without performing I/O functions to the file subsystem. BFXPERF can be used to do data transfer timings between BFX's without the overhead of file I/O.

The RMAXL parameter is used to identify the size of records to send. See the RMAXL definition for rules on this parameter.

The PERFCNT parameter is used to identify the number of records to send.

BFXPERF job example to run BFXEAT

```
RECEIVE
FROM      = BFXGENHOST
ID        = BFXPERF
MODE      = CHAR
RMOD      = BFXPERF
```

BFXPERF job example to run BFXGEN

```
SEND
TO        = BFXEATHOST
ID        = BFXPERF
MODE      = CHAR
RMOD      = BFXPERF
PERFCNT   = 9999
RMAXL     = 2499
```

Using BFX

To use BFX, the programmer must write two applications. These applications will be referred to as the initiating procedure and the responding procedure. The initiating procedure is executed on one system, and the responding procedure is executed on the other system. Each procedure is written using BFX command statements for the BFX product used on that host. The structure and contents of the initiating and responding procedures vary according to the application. There are three basic BFX applications.

- **Manual job submission** - the users on the two systems each submit one BFXTI/BFXTR job.
- **Automatic job submission** - the user on the initiating system submits the BFXTI job which contains the job control statements for the responding BFXTR job. The responding job is sent over the network to BFXJS and automatically submitted on the responding system. The BFXTI/BFXTR jobs then connect and transfer files.
- **Remote job submission** - the user on the initiating system submits a BFXTI job which contains job control statements for a remote job. The remote job is sent over the network to BFXJS and is submitted for execution on the responding system. This remote job executes independently of BFX.

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

Manual Job Submission

Manual job submission is the most basic application of BFX. Manual job submission enables users in the initiating and responding systems to manually submit BFX jobs for processing.

When using manual job submission, the user must provide an initiating job that uses BFXTI, and a responding job that uses BFXTR. Both the initiating and responding jobs are written using the job control language and BFX commands for the host they will be executed on.

Both the initiating and responding jobs contain matched sets of SEND and RECEIVE BFX commands. A SEND in one procedure must match a RECEIVE in the other procedure. The SEND command specifies the file to be read from the host that issued the SEND command. The RECEIVE command specifies the file to be written to the host that issued the RECEIVE command. The SEND and RECEIVE commands may also specify other information about the data being transferred. The first SEND or RECEIVE command in the BFXTI job must specify NOSUBMIT as a parameter to select manual job submission.

To begin the file transfer, a user on one host submits the BFXTI job, and a user on the other host submits the BFXTR job. When the jobs are executed, the BFXTR program connects to the BFXTI program (via NetEx/IP and the IP network) and the files are transferred.

```
@USE SENDFILE,FILE*BMD
@USE SENDFILE1,FILE*BMC
@USE RCVFILE,FILE*SEFOI
@ASG,A SENDFILE
@ASG,A SENDFILE1
@ASG,A RCVFILE             . BFX requires the files to be assigned
@XQT   BFX.BFXTI           . Start the Transfer initiator
SEND                       . Start of file transfer # 1
FILE = SENDFILE
TO   = UNI2
ID   = BFXJOB              . BFXJOB is the offer name on HOST UNI2
NOSUBMIT                   . Manual job submission example
.EOT                       . End of file transfer 1
SEND                       . Start of file transfer # 2
FILE = SENDFILE1
TO   = UNI2
ID   = BFXJOB              . BFXJOB is the offer name on HOST UNI2
.EOT                       . End of file transfer 2
RECEIVE                    . Start of file transfer #3 (Receive a file)
FILE = RCVFILE
ID   = BFXJOB              . BFXJOB is the offer name on HOST UNI2
FROM = UNI2
.END                       . END of BFX JOB
```

Figure 2. Host UNI1 manual job submission for initiating BFXTI job

```
@USE RCVFILE,FILE*BMD
@USE RCVFILE1,FILE*BMC
@USE SENDFILE,FILE*SEFOI
@ASG,A RCVFILE
@ASG,A RCVFILE1
@ASG,A SENDFILE            . BFX requires the files to be assigned
@XQT   BFX.BFXTR           . Start the Transfer Responder
RECEIVE                    . Start of file transfer # 1
FILE = RCVFILE
FROM = UNI1
ID   = BFXJOB              . BFXJOB is the offer name on HOST UNI2
.EOT                       . End of file transfer 1
RECEIVE                    . Start of file transfer # 2
FILE = RCVFILE1
FROM = UNI1
ID   = BFXJOB              . BFXJOB is the offer name on HOST UNI2
.EOT                       . End of file transfer 2
SEND                       . Read in file from UNI2
FILE = SENDFILE
TO   = UNI1
ID   = BFXJOB
.END                       . END of BFX JOB
```

Figure 3. Host UNI2 manual job submission for responding BFXTR job

The following text describes the events that occur when this BFX transfer takes place:

The user on UNI1 runs the INITIATE job, which invokes the BFXTI program. Immediately after this, the user on UNI2 runs the RESPOND job, which invokes the BFXTR program.

The OS2200 systems begin processing the jobs. (All of the ECL statements are described in "ECL and Control Parameters for " on page 13.) On UNI1, BFXTI is executed and the files to be transferred (identified by the USE names SENDILE, SENDFILE1 and RCVFILE) are allocated. Similarly on UNI2, BFXTR is executed and the files to be transferred (identified by the DD names RCVFILE, RCVFILE1 and SENDFILE) are allocated.

The SEND statement in the initiating procedure specifies that a file is to be sent from UNI1 to UNI2. The SEND statement also provides additional parameters related to the file transfer. Unspecified parameters are assigned default values. In this example, the NOSUBMIT parameter selects manual job submission. The .EOT statement signals the end of the parameters for this request.

The RECEIVE statement in the responding job specifies that a file is to be received from the sending host. Basically the same types of parameters are specified in the SEND and RECEIVE statements, but some parameters (the ones that are not changed) need only be specified in the first BFX statement.

Notice that a SEND in one job must match a RECEIVE in the other job.

The BFXTI program sends the file identified by the FILE parameter. In this case, the SENDFILE statement specifies that FILE FILE*BMD is to be sent. This file is sent to the RESPOND job, which receives it and calls it FILE*BMD, as specified in the FILE = RCVFILE and USE statements.

Notice that the pair of commands: SEND in the INITIATE procedure and RECEIVE in the RESPOND procedure, trigger the transfer.

The BFXTR and BFXTI programs continue processing. The BFXTI job then SENDs FILE*BMC over the network. BFXTR RECEIVEs it as FILE*BMC as specified by the FILE = RCVFILE1 and USE statements in the RESPOND job.

The BFXTR and BFXTI programs continue processing. The INITIATE job RECEIVEs file FILE*SEFOI over the network as FILE*SEFOI as specified by the FILE = RCVFILE and the USE statements in the BFXTI job.

BFXTR scans the command input and does not find any other BFX commands. BFXTR ends processing. The INITIATE BFXTI scans for commands and does not find any commands, then ends also. The .END statement may also be used to also signify the end of all BFX transfers.

There are several important points related to the preceding example.

- Either the BFXTI or the BFXTR side of the transfer can send or receive files.
- Each SEND statement must have a matching RECEIVE statement in the partner procedure.
- The IDs must match in the initiating and responding procedures.

Automatic Job Submission

Automatic job submission is the most commonly used method of transferring files using BFX. This method enables a user to transfer files to and/or from a responding system without user assistance on the responding system.

The user must first prepare an initiating job similar to the one used for manual job submission. This job identifies the responding host and defines the direction of the transfer. This job also describes the data and any special processing to be performed on the data.

The user must also prepare a set of job control information for the responding job and specifies the direction of the transfer. This responding job, written using the control language of the responding host, is physically encapsulated (or referenced) in the initiating job.

The basic idea of automatic job submission is that the initiating job first transfers the responding job across the IP network (using BFXTI). The BFX Job Submitter (BFXJS) program receives this responding job and

automatically submits it on its host system. The BFXTI and BFXTR programs then coordinate the transfer of the files as they did for manual job submission.

The initiating and responding hosts may be any host that has the appropriate NetEx/IP and BFX products installed, but to simplify the discussion we will assume both hosts are Unisys OS2200 systems.

```
@ASG,T    UNI2JOB
@ELT,IDQ  TPF$.UNI2JOB              . Create a temporary job file
@RUN,/A   BFXJOB,0/SECRET,BFX       . Start of job on the remote system
@CAT,P    FILE*NEWFILE.,///20
@USE      RCVFILE,FILE*NEWFILE
@ASG,A    RCVFILE
@XQT      BFX.BFXTR
RECEIVE
FROM = UNI1
FILE = RCVFILE
MODE = ASCII
ID   = BFXJOB
.END                        . End of BFX parameters for remote system
@END                        . END of JOB File to be sent to UNI2
@.
@USE SENDFILE,FILE*BMD      . Local system ECL
@ASG,A SENDFILE
@XQT   BFX.BFXTI            . Start the Transfer initiator
JOBFILE = UNI2JOB           . Point to the jobfile to run on the remote system
SEND                        . Start of file transfer # 1
FILE = SENDFILE
TO   = UNI2
ID   = BFXJOB               . BFXJOB is the offer name on HOST UNI2
.END                        . END of BFX JOB
```

Figure 4. BFXTI Automatic Job Submission

The following diagrams and text describe the events that occur when this BFX transfer takes place.

The user runs the INITIATE job, which invokes the BFXTI program. Running the INITIATE job triggers the following events:

Without the NOSUBMIT BFX parameter, BFXTI retrieves the job control language for the RESPOND job from its internal file (UNI2JOB), as specified by the JOBFILE = parameter. The BFXTI program then sends the RESPOND job to the BFXJS program on the remote host UNI2.

The BFXJS program submits the RESPOND job that was sent by BFXTI for processing on the responding host. The RESPOND job executes the BFXTR program that is resident on the UNI2 host and BFXTR begins execution. The INITIATE and RESPOND procedures interact in the same way that they did for manual job submission.

The BFXJS program is now ready to accept another job from any other user on the network.

The BFXTI program sends the file identified by the FILE = parameter and USE statements. In this case, the SENDFILE USE statement specifies that the file FILE*BMD is to be sent. This member is sent to the RESPOND job, which receives it as FILE*NEWFILE, as specified in the FILE = RCVFILE and USE statements in the RESPOND procedure.

Notice that the pair of commands, SEND in the INITIATE procedure and RECEIVE in the RESPOND procedure, trigger the transfer.

BFXTR then terminates when it does not find any other BFX commands or the .END statement. BFXTI also terminates when it does not find any commands or the .END statement.

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/sNetEx 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 data transfer is to be secure, a secure connection will be used, otherwise a non-secured connection will be used. The data will then be transferred.

The Authorization file is configured by the system administrator on each system. The location of the Authorization file is in the same location as the BFXJS program. The access rights to this file should be granted only to the userid that starts the BFXJS runstream. Each line of the file contains an initiating sNetEx Hostname and OS specific UserID separated by whitespace. (All trailing characters following the 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.

Remote Job Submission

Remote job submission is a special kind of job submission. Using remote job submission, a user can submit any job to the remote host (instead of a job 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.

The user must provide a local job and a remote job. The local job simply executes BFXTI and issues a SUBMIT command. The remote job executes independently from BFX.

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. BFXTI can then process other commands, or it terminates.

Remote Job Submission Example

Assume that a user on the local host wants to send a simple job that will erase a file on the remote host using remote job submission. The user writes the following local procedure:

```
@ASG,T    UNI2JOB
@ELT,IDQ  TPF$.UNI2JOB     . Create a temporary job file
@RUN,/A   ANYJOB,0/SECRET  . Start of job on the remote system
@. Enter the ECL statements for the job to run
@DELETE FILE*DELETEIT.
@.
@END                       . End of remote job statements
@.
@.                         . Start of local job ECL statements
@XQT   BFX.BFXTI           . Start the Transfer initiator
JOBSUBMIT
JOBFILE = UNI2JOB          . Point to the jobfile to run on the remote system
TO = UNI2
.END                       . END of BFX JOB
```

Figure 5. Remote Job Submission

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

The user runs the INITIATE procedure, which invokes the BFXTI program. Running the INITIATE procedure triggers the following events:

When BFXTI finds a JOBSUBMIT command, it establishes a NetEx/IP connection with BFXJS on the remote host. BFXTI then transfers the remote job to BFXJS on UNI2 and continues scanning for more control statements or terminates.

BFXJS receives the remote job from BFXTI, and submits the remote job for processing on the remote host. The remote job then executes independently from BFX.

Remote Hostname Substitution Table

This optional table is contained in bfxmapcfg 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 bfxmapcfg file under <your-bfx-loadlib>*secbfx 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 6. Remote Hostname Substitution Table on page 11.

```
#
#  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 6. Remote Hostname Substitution Table

Data Modes

The BFX utility transfers the data on a logical record basis using two 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 upon the character set used and the word size available to each host. Character mode allows most sequential source or text files to be moved between dissimilar hosts in a meaningful form.

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

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.

Security

Secure BFX provides the capability of transferring jobfiles (and associated Secure BFX protocol) privately/securely. Additionally, Secure BFX provides the capability of optionally transferring user data (and associated Secure BFX protocol) privately/securely. These controls are configuration parameters. System configurations can specify that ALL Secure BFX jobs be transferred utilizing an encrypted connection. The subsequent file transfers may optionally employ a secure connection. Existing host restrictions on utilization, validation, and accounting remain in effect.

File Size

Secure BFX is very efficient in transferring large files. Tape or disk data may be transferred. Record sizes up to 65,535 bytes can be transferred.

ECL and Control Parameters for Secure BFX

This section describes the ECL used to execute the BFXTR and BFXTI programs and the control parameters for these two programs. These control parameters, which may be control statements or parameters associated with a control statement, are used when executing the BFXTI and BFXTR programs.

ECL For Executing the BFXTI and BFXTR Programs

An example of the ECL needed to start and run the BFXTI and BFXTR programs is shown in Figure 7. Complete examples of BFXTI and BFXTR applications are contained in the "Applications" section on page 31.

```
@USE SENDFILE,FILE*BMD           . Attach a name to the file
@ASG,A SENDFILE                  . Assign the file to this job
@XQT   NETEX*BFX.BFXTI           . Start the Transfer initiator

If the partner system is also an OS2200 system, the contents of the RMTJOB file
would be:

@USE RCVFILE,FILE*NEWNAME        . Attach a name to the file
@ASG,A RCVFILE                   . Assign the file to this job
@XQT   NETEX*BFX.BFXTR           . Start the Transfer Responder
```

Figure 7. BFXTI / BFXTR example ECL

The following statements appear in Figure 7:

@USE

This parameter attaches a name to the file to be transferred.

@ASG,A

Assign the file to this job.

@XQT,*options* NETEX*BFX.BFXTI

This statement invokes the transfer initiate part of Secure BFX. The system will look for the program BFXTI in the NETEX*BFX program file.

The only recognized parameter is the ,f. This directs the BFX application to run in realtime mode. The RTLVL is from the TCPCFG element located in the execution file.

If the Secure BFX programs are copied to TPF$ before being executed, all the symbolic elements should also be copied. These elements are used by Secure NetEx/IP to set the COMAPI mode and specify a local IPv4 address to use to communicate with the SNXMAP component, as well as setting the global configuration parameters for Secure NetEx/IP.

Secure BFX Execution Parameter Syntax and Placement

All input to BFX is free form input. For example, the following two statements are equivalent:

```
FILE = SNDFILE

FILE=SNDFILE
```

Each statement should begin on a new line. The BFX request is terminated by the ".EOT" statement if you are doing multiple transmissions, a ".END" statement signifying the last transmission, or the end of the input file.

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.

Secure BFX processes the security file used by H301L and H304E products (@use configfile.,nnn*nnn.). Existing jobs will run without changes. The only parameter that has any meaning is the secure=yes, secure=no parameters. All the remaining parameters are deprecated. In new run streams, the @use configfile.,nnn*nnn. Could be completely discarded and a secure or nosecure parameter could be added with all the normal BFX parameters.

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 over-ridden 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 ECL run-streams. 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 NODSEC parameters. The SECHOST parameter would be used when the NOJSEC 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 NetEx system names and the userids that are allowed. The table consists of two fields and any additional comments you may wish to include. The first field is the NetEx hostname (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. The userid “user1: does NOT match the 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.

BFX Execution Parameters

The BFX execution parameters consist of three control statements, SEND, RECEIVE, and JOBSUBMIT, with a set of parameters. All three control statements use the same parameters, as shown in Figure 8 on page 18.

The SEND and RECEIVE control statements specify the direction of the file transfer. The JOBSUBMIT control statement specifies that a remote job shall be simply submitted on the remote host, and then BFXTI will end activity.

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.

BFX Component Commands and Parameters

Command	Default	Used By
.END		BFXTI, BFXTR
.EOT		BFXTI, BFXTR
ARCHIVE		*Unsupported – ERROR*
BLKMX		*Unsupported – ERROR*
BLOCK=	8191Words	BFXTI, BFXTR
BIGBLK		BFXTI, BFXTR
BMOD		*Unsupported –IGNORED*
BPARM		*Unsupported –IGNORED*
CHKSUM		*Unsupported – ERROR*
CLOSE		*Deprecated – IGNORED*
CONNDELAY=	5	BFXTI, BFXTR
CONNMAX=	20	BFXTR
CREOV		BFXTI, BFXTR
DEBUG=	NO	BFXTI, BFXTR
DELAYTIME=	5	BFXTI, BFXTR
DSEC	DSEC	BFXTI, BFXTR
DSEOV		BFXTI, BFXTR
DSLAB		BFXTI, BFXTR
FILE=	BFXFILE	BFXTI, BFXTR
FILES=	number or *	BFXTI, BFXTR
FROM=	NTXLCL	BFXTI, BFXTR
HOBCHECK=	OFF	BFXTI, BFXTR
HOBOPT		*Unsupported – ERROR*
HOSTCHK	HOSTCHK	BFXTI
ID=	“????????”	BFXTI, BFXTR
IDSTAMP	NOIDSTAMP	BFXTI, BFXTR, BFXJS
JBLOCK=	8191Words	BFXTI, BFXJS

Command	Default	Used By
JID=	BFXJS	BFXTI, BFXJS
JFILE		*Unsupported – ERROR*
JFLAG		*Unsupported – ERROR*
JOBFILE=	JOBXXXX	BFXTI
JOBSUBMIT	JOBSUBMIT	BFXTI
JREPEATCONN=	5	BFXTI
JRMAXL=	240	BFXTI, BFXJS
JSAUTH	JSAUTH	BFXTI, BFXJS
JSEC	JSEC	BFXTI, BFXTR
KEYIN	BFXJS	BFXJS
LETLABEL	LETLABEL	BFXTI, BFXTR
MDSEC	MDSEC	BFXTI, BFXTR
MJSEC	MJSEC	BFXTI
MODE=	ASCII	BFXTI, BFXTR
MSGLEVEL=	4	BFXTI, BFXTR, BFXJS
MSGLVL=	4	BFXTI, BFXTR, BFXJS
NEWHOST=	None	BFXTI
NOCLOSE		*Deprecated – IGNORED*
NOCR		*Silently ignored*
NOCRLF		*Silently ignored*
NODSEC	DSEC	BFXTI, BFXTR
NOHOSTCHK	HOSTCHK	BFXTI
NOIDSTAMP	NOIDSTAMP	BFXTI, BFXTR, BFXJS
NOJSAUTH	JSAUTH	BFXTI, BFXJS
NOJSEC	JSEC	BFXTI, BFXTR
NOLF		*Silently ignored*
NOMDSEC	MDSEC	BFXTI, BFXTR
NOMJSEC	MJSEC	BFXTI, BFXJS
NONSDF		BFXTI, BFXTR
NOSECURE	SECURE	BFXTI, BFXJS
NOSOE	NOSOE	BFXTI, BFXTR
NOSUBMIT	JOBSUBMIT	BFXTI

Command	Default	Used By
NOTIMESTAMP	NOTIMESTAMP	BFXTI, BFXTR, BFXJS
PAD	none	BFXTI, BFXTR
RATE		*Deprecated – IGNORED*
RBM		BFXTI, BFXTR
RECBUFF	0	BFXTI
RECEIVE	SEND	BFXTI, BFXTR
RECSIZE=	(must be specified)	BFXTI, BFXTR
REPEATCONN=	20	BFXTR
RESTORE		*Unsupported – ERROR*
RMAXL=	1024 Words	BFXTI, BFXTR, BFXJS
RMOD		BFXTI, BFXTR
RPARM	(blanks)	*Silently ignored*
LABELS		BFXTI, BFXTR
SECURE	SECURE	BFXTI,BFXTR
SEND	SEND	BFXTI, BFXTR
SENDJB		*Unsupported – ERROR*
SOE	NOSOE	BFXTI, BFXTR
SPERRY		BFXTI, BFXTR
TIMEFL		*Unsupported – ERROR*
TIMEOFFER=	240	BFXTI, BFXJS
TIMEOUT=	60	BFXTI, BFXTR, BFXJS
TIMESTAMP	NOTIMESTAMP	BFXTI, BFXTR, BFXJS
TO=	NTXLCL	BFXTI, BFXTR
TRACKIO		BFXTI, BFXTR
UNPACK		BFXTI, BFXTR
USE_NAME		*Unsupported – ERROR*

Figure 8. BFX Command and Parameter List

Control Statements

SEND

This statement indicates that the file transfer will be from the local host to the remote host. The TO= statement names the NetEx/IP host that the file will be transferred to. This name is configured by the systems programmer when the NetEx/IP network configuration is generated.

For any given file transfer, a SEND statement on one end must match a RECEIVE statement on the other end.

RECEIVE

This statement indicates that the file transfer will be from the remote host to the local host. The FROM= statement names the NetEx/IP host that will send the file. This name is configured by the systems programmer when the NetEx/IP network configuration is generated.

For any given file transfer, a RECEIVE statement on one end must match a SEND statement on the other end. If both sides specify either SEND or RECEIVE, an error occurs.

JOBSUBMIT

This statement indicates that a remote job is to be submitted for execution on the responding host. The TO= statement names the NetEx/IP host that the JOB will be transferred to. This name is configured by the systems programmer when the NetEx/IP network configuration is generated. The remote job is included in the BFXTI execution statements where the BFXTR job is otherwise included. When the JOBSUBMIT statement is specified, the BFXTI program does not wait for the other host to connect back, but simply submits the job to BFXJS and terminates.

Control Statement Parameters

The SEND, RECEIVE, and JOBSUBMIT statements use the following parameters (shown in Figure 8 on page 18). Defaults for many of these parameters may be defined in the BFX CFG files. Refer to “Installing ” on page 35 for more information.

COMMENTS

All control statement parameters must be specified on a separate line. Only one parameter statement is allowed per line. All other characters after the parameter and the associated value will be treated as comments. COMMENTS ARE NOT PERMITTED ON THE STATEMENTS FOR RPARM, TRACKIO, LABELS, FILES, RECSIZE, BIGBLK, DSLAB, DSEOV, and CREOV. These were once all specified on the rparm statement, but may now be entered on separate lines. If comments are entered on these lines, it may cause the BFX transfer to function improperly. The external symptom will be multiple BFX transfer statements parameters for different control statements (SEND, RECEIVE, JOBSUBMIT) listed, without the BFX transfer messages. The last value specified will be the value used.

ID

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

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

BLOCK

This optional parameter specifies the size in words of the buffers of data to be sent through Secure NetEx/IP to the other host. The size in BLOCK must be at least large enough to accommodate the largest logical record specficied in OCTETS to be sent from the file. If specified, it should be included on both the BFXTI and BFXTR execution parameters; 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.

If this parameter is not supplied, an installation-defined default is used. Valid range is 1-16383 words (4-65536 bytes).

BIGBLK

The BIGBLK parameter can be used to send large blocks of tape data across the network and write this out to a tape on the receiving side. It uses a special protocol. Both sides must use the BIGBLK protocol. This can be automatically invoked by specifying the SPERRY parameter, NOT using the RBM parameter and using a tape file. It does not need special authorization to process the tape labels. This is only supported on Unisys. The tapes may be written in either C-FORMAT or A-FORMAT(QUARTER WORD).

CONNDELAY/DELAYTIME

This optional parameter specifies the number of seconds to delay between connect attempts when a connect attempt gets a 3501 or 3502 error. (Same as DELAYTIME)

The default is 10 seconds. Valid range is from 0 to 32767.

CONNMAX

This optional parameter specifies the number of times to try connect when a connect attempt gets a 3501 or 3502 error. Valid range is from 0 to 32767.

CREOV

When transferring tape files, CREOV allows the creation of end-of-volume records on the receiving tape. This is only meaningful if the RBM parameter is specified.

DSEC

The DSEC command specifies that for this file transfer, the data and associated Secure BFX protocol will transfer securely.

DSEOV

When transferring tape files, DSEOV specifies that the tape end of volume records (EOV1 & EOV2) should not be sent across the network to the receiving side. The tapes involved in the transfer will not be mirror images. The output tape may not even be a multivolume dataset. All user data will still be sent to the remote side.

DSLAB=n

When transferring tape files, DSLAB specifies that the tape label records (VOL*n*, HDR*n*, UHL*n*, EOF*n*, EOV*n*, UTL*n*) should not be sent across the network to the receiving side. The tapes involved in the transfer will not be mirror images. All user data will still be sent to the remote side. This defaults

to false. If the labels parameter is also false, dslab becomes true for the bigblk record module and false for the BFXRBM record module.

FILE = *filename*

This parameter specifies an assigned file name for the input or output file. The name is either an internal or external name with a maximum of 12 alphanumeric characters.

FILES=nnn|*

This statement adds the FILES=nnn|* clause to the LABELS (see preceding discussion of LABELS) statement and is used only if transferring tape files from a UNISYS 2200 system to another UNISYS 2200 system. The FILES=nnn or the FILES=* clause causes multiple files to be sent within one BFX session. The collective nnn files are viewed as one transfer. The receiving UNISYS BFX writes the required EOF records on the tape to separate the files.

HOBCHECK

This parameter is the "High Order Bit Check." When sent to OS 2200 systems, character mode transfers are done in A-format. A-format regards each character as nine bits, with the low order eight bits as data and the high order bit as a "zero."

There are four HOBCHECK options, as follows. In H305 the hob bit will always be cleared in a character mode transfer. This is done by TCP on the system.

HOBCHECK ABORT

This will cause BFX to abort the file transfer if the ninth bit is on in any byte.

HOBCHECK CHECKONLY

This will cause BFX to issue a warning message if the ninth bit is on in any byte.

HOBCHECK CLEAR

This will cause BFX to set the ninth bit in any byte to zero if the bit is on. Use this feature with caution.

HOBCHECK OFF

This will cause BFX not to interrogate the ninth bit and let the data pass through. HOBCHECK off is a default.

HOSTCHK | NOHOSTCHK

This optional parameter 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 H305 product.

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

IDSTAMP

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.

JBLOCK

This command specifies the block size in words 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

This optional parameter specifies an alternate name for the BFXJS job submission program on the remote host program. This parameter is ignored if BFXTR was invoked; it is also ignored on any control parameter statement other than the first one.

If this parameter is not supplied, an installation-defined default is used (BFXJS).

JOBFILE

This optional parameter specifies an assigned file name for the ASCII card image file. The name is either an internal or external name with a maximum of 12 alphanumeric characters.

JREPEATCONN

This parameter 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/CONNDELAY secs in between connect attempts. Valid range is 0-32767.

JRMAXL

This parameter 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 32,767.

The JRMAXL command is legal in BFXTI and BFXJS.

JSAUTH

If JSAUTH is set in bfxjs.cfg, a SECURE job submission is checked for authorized user. (Default is JSAUTH.)

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

KEYIN

Specifies the console keyin used to communicate with BFXJS. Currently “TERM” is the only valid command, and requests BFXJS terminate.

LETLABEL

This parameter specifies the accounting code for processing tape labels, used by the sender / receiver record module.

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

This optional parameter specifies the nature of the information in the file. The ASCII parameter implies that the entire contents of the file consist of human-readable ASCII text (character mode). The BIT parameter implies that the file contains at least some information that is binary in nature: binary integers, floating point numbers, packed decimal numbers, or others (bit mode).

If the file being transferred is going to another Unisys 2200 processor, then the file transfer processing will be similar for any of the MODE parameters selected. However, if the destination processor is not a Unisys 2200 host, the processing differs according to the MODE specified.

If MODE=ASCII is specified, the ASCII text will be converted to the character set of the receiving machine. The logical records of character information will be converted to the corresponding logical record structure of the non-Unisys 2200 host system.

If MODE=BIT is specified, then the exact pattern of bits in the logical record will be sent to the other host and stored as a binary logical record. This record would presumably be converted to a useful form at some later time. Record sizes (set by RMAXL) in binary transfers must be multiples of a 2200 mass storage sector (28 words). If the data exchange is between a Unisys system and a zOS system, you should use an lrecl of 2016 for the IBM dataset and a RMAXL of 448 on the Unisys system, or a multiple of these numbers. 2016 is the least common multiple of the 32 bit IBM system, the 36 bit Unisys system, and the sector size of 28 words on the Unisys disk drive. These parameters should avoid record truncated messages on the IBM side.

If MODE=OCTET is specified, binary data are received or sent in multiples of 8 bits, but do not need to be 28-word multiples. THE INPUT TAPE MUST BE WRITTEN IN A-FORMAT (QUARTER WORD). The RMAXL parameter on the receiving side specifies the blocking factor, on the sending side the recsize parameter MUST be specified.

If MODE=EBCDIC, the NetEx/IP EBCDIC mode (3) will be used. This is for applications that want to "store and forward" the EBCDIC data without translation.

Both sides should consistently specify the MODE. If the two specifications are inconsistent between bit and text mode, the transfer will be aborted and BFX error message BFX124S will be issued. (See Appendix B for a description of BFX error messages.)

MSGLVL

This optional parameter specifies the type of messages that the user wants to see in the PRINT$ file. The parameter must be specified as a decimal number in the range of 0-16. The meaning of the various message levels is shown below:

0-3 This will generate messages that are meant for diagnostic purposes. These messages will trace the flow of events in BFX in detail, and are intended only for use in diagnosing problems with BFX or NetEx/IP.

4-7 This will generate messages indicating the status of job submission, starting of the remote BFXTR job if applicable, and statistics on the file transferred.

8-11 If the file transfer is normal, this will generate only the transfer complete message. If an error is encountered that causes the transfer to fail, 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.

If this parameter is not supplied, an installation-defined default is used.

NEWHOST

This optional parameter specifies a new host name of the NetEx/IP program to be used for file transfer. A job file will be sent to the specified host and all further file transfers will take place between these two hosts.

This parameter is valid only when used with the BFXTI program.

NONSDF

This optional parameter indicates to BFX that the file being received or sent is to be accessed in binary mode. The file may or may not be stored in SDF format. All bits in the file will be transmitted. This may include any SDF records if the file is in SDF format. If the mode is bit, all bits are sent across the network. If the mode is ASCII, all bits will again be transferred, but the ASCII characters will be translated according to the mode specified on the receiving job.

As with all binary mode reads or writes, the RMAXL parameter determines how many words will be read or written with each record. For disk files, the RMAXL must be set as a multiple of a mass storage sector (28 words). For tape files, RMAXL need only be larger (in words) than the largest record on the tape.

NEWHOST

This optional parameter specifies a new host name of the NetEx/IP program to be used for file transfer. A job file will be sent to the specified host and all further file transfers will take place between these two hosts.

This parameter is valid only when used with the BFXTI program.

NODSEC

The NODSEC command specifies that for this file transfer, the data and associated Secure BFX protocol will NOT transfer securely. This cannot override the setting in a configuration file.

NOIDSTAMP

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

If NOJSAUTH is set in bfxjs.cfg, a SECURE job submission is not checked for authorized user. (Default is JSAUTH.)

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

NOJSEC

The NOJSEC command specifies that for this file transfer, the jobfile and the associated Secure BFX protocol will NOT transfer securely. This cannot override the setting in a configuration file.

NOSUBMIT

This optional parameter specifies that the BFXTI job does not contain a BFXTR job for submission on the remote host. (NOSUBMIT selects manual job submission as described in the introduction.) If NOSUBMIT is specified, the user on the local host and the user on the remote host must manually submit the BFXTI and BFXTR jobs on the two hosts. The users must coordinate the submitting of the jobs so that the BFXTI job is started just before the BFXTR job.

NOSECURE

This command specifies the job file is sent on an unencrypted connection. BFXJS must be offered in an unsecured mode. See "Using SECURE BFX" for restrictions. (Same as NOJSEC.)

PAD

The only valid parameter is SPACE. For an ascii transfer where the file resides on a disk, odd length records will be padded with spaces up to the next full word boundary. Using the PAD parameter in BFXTxCFG files will cause BIT mode transfers to fail with BFX416S error messages.

RECBUFF

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

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. Valid range is 0-32767.

RMAXL

This optional parameter specifies the maximum length of any record in the file. This value must be less than the block size. It must also be large enough to accommodate the largest record in the file; otherwise that record will be truncated. In BIT mode, RMAXL must be a sector multiple (28 words). It is used to determine whether each new record will fit into the current block so too large a value will cause part of each block to be unused. **In an octet mode transfer, this field contains the count of records to be written in one physical block on the output tape.** RMAXL must not exceed 32,767.

RBM

This optional parameter is used in sending and receiving tape data across the network. It requires the SPERRY parameter to be specified. If the tape labels are to be sent, the job must be allowed access to the tape labels. THE INPUT TAPE MUST BE WRITTEN IN A-FORMAT (QUARTER WORD).

RMOD

This parameter is the record module name.

JSEC/SECURE

Specifies the JOBFILE should use an encrypted connection. See "Using SECURE BFX" for restrictions.

SOE | NOSOE

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

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

NOSOE is the default setting.

SPERRY

This parameter invokes specific I/O and file-handling capabilities which are meaningful only for 2200 to 2200 transfers.

TRACKIO

This statement is used only if the transfer is between two 2200 systems and is a disk-to-disk transfer. The SPERRY and TRACKIO options must be specified for each BFX. Specifying these options for both BFXs causes the data to be transferred as records of 1794 words (1792 data + 1 address + 1 checksum). The checksum field is always 0, so it is not validated. Each record has the mass storage address (sector) of that track. This permits program files to be transferred. BFX skips over the "hole" between the directory and the data. TRACKIO requires at least 2000 word records.

LABELS

This statement is used primarily to send labeled tapes (with possibly multiple tape files) between two 2200 systems. RPARM = LABELS can be used to receive tape files from a non-2200 system. In that case, the SPERRY and RPARM = LABELS statements would appear in the UNISYS BFX input along with the RECEIVE verb. The SEND verb would be on the non-2200 side. Sending tape files from a UNISYS host to a non-UNISYS host does not involve the SPERRY and RPARM= LABELS directives. When these directives are specified on a SEND, H305e adds additional "non-data" records to the transfer. The non-data records inform the receiving BFX (also with SPERRY and LABELS) how to write the destination tape. Without the SPERRY and LABELS directives, sending from a tape file will only send the data records.

Multi-volume tape files may be read correctly with or without the SPERRY and LABELS options. The NetEx/IP parameter READTO should be increased to ·allow for the delay. If BFX jobs abort with a 2306 error, and the data set is a multi-volume tape set, then increasing READTO on both hosts will probably fix the problem.

TIMEOF

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

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

The default setting of this parameter is 240. It currently only has the effect in BFXTI. BFXJS will always use a TIMEOF = 0. Valid range is from 0 to 32767.

TIMEOU

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

Note: BFX now retries all read timeouts so this essentially will do nothing to the operation of the job. However it can still be set for compatibility.

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

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

The default value for TIMEOU is 60. Valid range is 0-32767.

TIMESTAMP | NOTIMESTAMP

These optional parameters specify if a time of day timestamp is to precede all BFX generated messages (TIMESTAMP) or not (NOTIMESTAMP).

If present, the timestamp will take the form *hh.mm.ss* (where *hh* = hours (24-hour clock), *mm* = minutes, and *ss* = seconds).

If this parameter is not supplied, an installation-defined default is used.

UNPACK

The data written on the tape should be written 8 bits to the byte. This is the default when the receiving tape is assigned in Q-Format.

.EOT

The .EOT control statement specifies that all the control parameters for this transfer have been specified and the transfer is to take place. This statement is used when more transfers are to follow; otherwise the .END statement is used.

.END

The .END control statement specifies that all the control parameters for this transfer have been specified and the transfer is to take place. This statement also marks the end of any further transfers. The physical end-of-file or @EOF performs the same function as this statement.

BFX Example

Figure 9 contains typical batch control statements used to transfer a cataloged file from HOSTA to HOSTB. The file is an ASCII file with a maximum record length of 22 words.

```
@RUN      BFXTI,acct,proj,...
@ASG,A    FILE2SEND.
@ASG,T    UNI2JOB
@DATA,IQ  TPF$.UNI2JOB         . Create a temporary job file
@RUN,/A   BFXTR,acct,proj,...  . Start of job on the remote system
@PASSWD   SECURI
@DELETE,C FILE2RECV.
@CAT,P    FILE2RECV.,///200
@ASG,UP   FILE2RECV,F
@XQT      file.BFXTR
RECEIVE
MODE=ASCII
ID=BFXTI
FROM=HOSTA
FILE=FILE2RECV
BLOCK=2000
RMAXL=22
.END
@END                           . End of remote job
@.                             . Start of local job
@XQT file.BFXTI
SEND
JOBFILE = UNI2JOB
MODE=ASCII
ID=BFXTI
TO=HOSTB
FILE=FILE2SEND
BLOCK=2000
RMAXL=22
.END                           .End of local of BFX JOB
@FIN
```

Figure 9. Typical BFXTI Run Stream

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/IP 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.

Record Formats and Record Type Conversion

The supplied Record Module uses the standard SDF library routines (SDFI$/SDFO$) to read and write ASCII character files. Bit-string mode files are read and written with the normal IOW$ routine.

The RMAXL specified for a character file should be at least as large as the longest SDF image in the file. Bit string files may be transferred to or from the UNISYS host using either MODE= BIT or MODE= OCTET. Both modes process the data just as it appears in the source file. The major differences are as follows.

- MODE=BIT

  This mode assumes that the BFX bit-mode records are multiples of UNISYS 2200 mass storage sectors (28 words or 126 8-bit bytes). If a record's length is different from this, the transfer is aborted. This method of transfer is efficient for transfers between 2200 systems and for non-2200 systems where BFX record sizes accommodate the multiple sector rules.

- MODE=OCTET

  This feature is also a bit-mode transfer, but it does not require that records, either receiving or sending, be a multiple of a disk sector. The bit-length of each record must only be a multiple of eight bits. On input, the data is moved from each record into a buffer (1792 words). Records are packed together, the end of the previous record immediately preceding the first byte of the next, with no intervening record separator. When the buffer is full, the records are written to the file. This sequence is repeated until all incoming records are processed. If the data is to be later returned to the sending host, all records must be the same length.

  Because bit mode data has no record structure, MODE= OCTET requires that the record length be stated in a RPARM = RECSIZE = nnnn statement. This statement is used only for SEND operations. The nnnn value is the length of each record in 8-bit bytes (octets). BFX will then logically partition the file into records of this length and send them to the remote host in bit mode.

BFX will block the records for transfer using as many records as can fit in the negotiated block size, and de-block on the receiving side. The supplied code is designed to support transfer of file data between "incompatible" computers in two ways:

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

Applications

The following examples are provided in this section.

- Unisys OS2200 to Linux
- Unisys OS2200 to IBM OS2200

Unisys OS2200 to Linux

The example in Figure 10 shows the Unisys OS2200 job used to perform a Secure BFX transfer from an OS2200 initiating host to a Linux remote host. The NetEx/IPhost name of the OS2200 initiating host is CHICA, and the NetEx/IP hostname of the Linux remote host is WASHC. The user wants to send a single file from CHICA to WASHC.

```
@RUN      BFXTI,acct,proj,...
@ASG,T    UNI2JOB
@DATA,IQ  TPF$.JOBFILE         .Create a temporary job file
#nesi
#nesi
cd /home/nesi
rm DATA5MR2
/usr/share/nesi/sbfx/bin/bfxtr<<EOF
receive from CHICA file DATA5MR2 id A2 mode char msglvl 0 timestamp block 32000
EOF
@END
@.
@USE FILE2SEND.,FILE*SENDFILE.
@ASG,A    FILE2SEND.
@USE BFX,H305*RELEASE100.
@XQT BFX.BFXTI
SEND
MODE=ASCII
ID=A2
TO=WASHC
FILE=FILE2SEND
BLOCK=3000
RMAXL=22
.END of BFX JOB
@FIN
```

Figure 10. Example Unisys initiated BFX file transfer from Linux to Unisys

Unisys OS2200 to IBM

The example in Figure 11 shows the Unisys OS2200 job used to perform a Secure BFX transfer from Unisys initiating host to an IBM remote host. The Secure NetEx/IP name of the Unisys initiating host is CHICA, and the Secure NetEx/IP name of the IBM remote host is WASHC. The user wants to send a single file from CHICA to WASHC.

```
@RUN      BFXTI,acct,proj,…
@ASG,T    UNI2JOB
@DATA,IQ   TPF$.UNI2JOB
//BETAUS1  JOB ,CLASS=A,MSGCLASS=A,NOTIFY=&SYSUID
//SEND    EXEC PGM=BFXTR,
// PARM='RECEIVE FROM=CHICA  ID=BFXU MODE=CHAR TIME HOSTCHK MSGLVL 0’
//STEPLIB  DD  DSN=NETEX.SBFX.BFXLOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*,DCB=BUFNO=01
//FILEOUT  DD  DSN=BETATST.UNISYS.DATA1,DISP=SH
//RMTJOB   DD  DATA,DLM=ZZ
@END
@USE SENDFILE,FILE*SEND.
@ASG,T SENDFILE,F///99999
@USE BFX,H305*RELEASE100.
@XQT    BFX.BFXTI
SEND
TO = WASHC
FILE = SENDFILE
BLOCK = 2000
MODE = ASCII
ID = BFXU
TIMESTAMP
@END
```

Figure 11. Unisys initiated BFX file transfer from Unisys to OS2200

Installing Secure BFX

This section describes the installation of Secure BFX in step-by-step procedures.

Prerequisites for Installation

H305 Secure BFX needs the following prerequisites for successful operation:

- A Unisys OS2200 system that is running the H304 Secure NetEx/IP software product.
- At least one other host with its own Secure NetEx/IP and Secure BFX software. Secure NetEx/IP is not network compatible with the older releases of NetEx and BFX.
- Before installing H305 Secure BFX fox OS2200, the H304 Secure NetEx/IP installation must be completed.

Release Distribution

H305 Secure BFX for OS2200 is distributed as a downloadable file.

Installation Process

This section describes the installation procedure for the H305 Secure BFX Release.

The following steps outline the installation process. Before proceeding with the installation, please read the Memo to Users accompanying the distribution for any additions or changes to the installation instructions.

Obtain the BFX distribution file.

The H305 distribution comprises one file which may be FTPed to your Unisys host file by using the ftp parameters "bin" and "quote site cfmt". To download the distribution file, contact support at support@netex.com for download instructions.

FTP the distribution file to OS2200.

FTP the distribution file to the OS2200 system as follows:

1) Connect via FTP to your OS2200 system.
2) If necessary, change the location of your local directory to the location of the distribution file:

   lcd 'directory-name'
3) Set the required attributes for the file:

   bin

   quote site cfmt
4) Transfer the distribution file, e.g.:

   put H305_RELEASE_nnn_CFMT H305*RELEASE-nnn.
5) Quit your FTP client

Using the above names results in the distribution file residing on OS2200 as the program file:

H305*RELEASE-nnn

Check for required updates.

Check if there are any updates by going to www.netex.com, and under the Support tab select products – follow the link to Unisys Clearpath/Dorado (H30x) platform and then select 'Updates' for the appropriate H30x product. If there are any, download them and follow their installation instructions.

Upgrading H305

This is a new product. You must follow the "Software Installation" instructions.

Removing H305

The product can be removed from your system, by deleting the appropriate files. There are no solar considerations.

Software Installation

1. Edit the member in the release file link-secbfx.
2. In lines 11 and 12 change netex*secbfx to the name of your new load library for Secure BFX.
3. In lines 15 and 16 change netex*secbfxjs to the name of your new load library for Secure BFXJS.

4. In line 20 change H305*release000 to the name of the file you ftped to your system,
5. In line 21 change H304*release000 to the name of the release file for the Secure NetEx/IP product.
6. In line 24 change SYSLIB*COMAPI$TOOLS to the comapi tools library installed by the OS2200. This is to access the comapi router, The comapi used is controlled by the TCPCFG member below,
7. Save and @start this ecl. This run is currently not break pointed to a file.
8. Update the symbolic members in your secbfx load libraries.
    a. TCPCFG
        i. COMAPI should specify the COMAPI mode you wish to use.
        ii. INTV4ADDR should specify a 127.0.0.n address assigned to the correct COMAPI.
        iii. PTMPORT should specify the TCP port used by the SNXMAP component of Secure NetEx/IP. This is normally 3919. This must match on all your Secure NetEx/IP hosts.
        iv. RTLVL specifies the realtime level netex and optionally BFX is run at, the range is 5-35. 0 disables realtime.
    b. BFXTICFG
        i. Should specify the Secure BFX global parameters for BXTI to use.
        ii. Verify the security parameters you wish to use. See the section "Using SECURE BFX" for a discussion of parameters.
    c. BFXTRCFG
        i. Should specify the Secure BFX global parameters for BXTR to use.
        ii. Verify the security parameters you wish to use. See the section "Using SECURE BFX" for a discussion of parameters.

9. Update the symbolic members in your secbfxjs load library.
    a. TCPCFG
        i. COMAPI should specify the COMAPI mode you wish to use.
        ii. INTV4ADDR should specify a 127.0.0.n address assigned to the correct COMAPI.
        iii. PTMPORT should specify the TCP port used by the SNXMAP component of Secure NetEx/IP. This is normally 3919. This must match on all your Secure NetEx/IP hosts.
    b. BFXJSCFG
        i. Should specify the Secure BFX global parameters for BXJS to use.
        ii. Verify the security parameters you wish to use. See the section "Using SECURE BFX" for a discussion of parameters.
    c. BFXAUTHCFG
        i. See the section "Using SECURE BFX" for a discussion of parameters

Modify BFX Authorized User file – BFXAUTHCFG

The Authorization file is configured by the system administrator and secured by file access rights to the user that starts BFXJS. The Authority file is in your secure BFXJS load library. 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.)

Execute the BFXJS Program

INSTALLATION NOTE:

@USE JOBSTART,NETEX*JOBD. points to a dataset that is used to submit jobs to the OS. If this dataset does not exit, you should execute the command @cat,p NETEX*JOBD.,///50. If this is not done, the bfxjs run will not be able to submit jobs to the OS. The BFXJS log will should a Start failure error =06. To run multiple copies of BFXJS, the JOBSTART @USE statement must point to different cataloged files.

It is the responsibility of the installation systems programmer to provide the ECL needed to start and run the resident BFXJS program. Sample ECL to do this is shown in Figure 12 on page 38. This sample ECL is contained in the release file element: secbfxjs.

The run breakpoints the print file to NETEX*secjs. The use statement points to your load library for Secure BFX. BFXJS will remain in execution until (program component of Secure NetEx/IP) terminates. BFXJS will remain in execution and print a message every 30 seconds until SNXMAP is started again.

```
@RUN,/A    SECJS,NETEX/DON,NETEX
@.
@use      BFX.,yourLibrary
@.
@CAT,PG   netex*secjs(+1).
@cycle      netex*secjs.,5
@USE       OUTPUT,netex*secjs.
@SYM,DF   PRINT$
@BRKPT    PRINT$/OUTPUT
@FREE     TPF$.
@ASG,T    TPF$.,///2000
@.
@COPY,A    BFX.BFXJS
@COPY,s    BFX.
@USE      JOBSTART,NETEX*JOBD.
@USE      PADS$PF.,tpf$.
@.
@XQT       BFXJS
JOBFILE     = JOBSTART
.end
```

Figure 12. Sample BFXJS ECL

Verify the Secure BFX Installation (BFXJS needed)

The installation of Secure BFX can be verified by signing on to the OS 2200 system and issuing:

- @USE BFX,<your-bfx-loadlib>
- @ADD BFX.VERIFY-TEST

Follow the on screen prompts as directed.

Appendix A. BFX Internal Summary

The following paragraphs briefly summarize the internal structure of BFX. Table 1 lists and describes the BFX default modules. All modules have 6-character names with the first 3 characters being BFX. Some references to the modules refer only to the last 3 characters of the name.

Immediately following Table 1 are three block diagrams showing the interaction of these modules in each of the three BFX programs.

Table 1. BFX Default modules

Module	Description
BFXTI	BFXTIB entry point and control module. BFXTIB scans control parameters, and sets them. Calls the SND module to transfer the job file to the remote BFXJS program, then calls the SND or RCV module to transfer the file according to the user's parameters. If additional transfers are provided, it continues the cycle until all files have been transferred.
BFXTR	BFXTR entry point and control module. BFXTR scans control parameters, and sets them. Calls the SND or RCV module to transfer the file according to the user's parameters. If additional transfers are provided, it continues the cycle until all files have been transferred.
BFXJS	BFXJS entry point and control module. BFXJS scans control parameters, and sets them. Calls the RCV module to receive the job file and submits it to the batch internal reader. BFXJS will continue to call RCV until canceled by the operator.
BFXRCV	This module directs the process of receiving a file from a remote BFX program. If invoked by TI, it will call SOF to complete session negotiation with the CONNECTing side. If invoked by TR, it will call SCN to connect to TI and negotiate parameters. When the connection is successfully established, it will call the receiving Block Module to allow the data to be written to the file. During transfer, it processes all generated informational and error messages, and detects and handles error and EOF conditions.
BFXSND	This module directs the process of sending a file from a remote B FX program. If invoked by TI, it will call SOF to complete session negotiation with the CONNECTing side. If invoked by TR, it will call SCN to connect to TI and negotiate parameters. When the connection is successfully established, it will call the transmitting Block Module to obtain the data from the file. During transfer, it processes all generated informational and error messages, and detects and handles error and EOF conditions.
BFXSOF	This module issues an SOFFR request to allow another BFX program to connect to it. When the connection completes, the module negotiates the file transfer parameters.

Table 1. BFX Default modules	
Module	**Description**
BFXSCN	This module issues a SCONNect request to connect to an offered BFX program. When the connection completes, the module negotiates the file transfer parameters.
BFXRBM	This module is the Receiving Block Module. It accepts buffers of blocked file data delivered to it by the calling RCV module. It breaks the block into logical records and calls the requested Record Module to write the record on the file.
BFXSBM	This module is the Sending ·Block Module. It calls the Sending Record Module to get logical records of data from the file and returns to SND to have the data transmitted over the network
BFXRRM	The Receiving Record Module opens the sequential file for output, and accepts data to the calling RBM module on a logical record basis. It handles record type conversions and closes the file when EOF or error conditions are sent.
BFXCRM	This module is the receiving record module for receiving submitted job files. BFXCRM uses a remote symbiont interface (RSI) station. The received job is written to the RSI job station, rather than to mass storage. (The RSI job station appears as a card reader to the 2200 operating system.)

Table 1. BFX Default modules	
Module	**Description**
BFXSRM	The Sending Record Module opens the sequential file for input, and provides data to the calling SBM module on a logical record basis. It detects EOF and generates any file specific error or warning messages.
BFXUIM	BFXUIM will test and call (if present) any site installed Block and Record Modules.

Block Diagram

Many modules are used in several or all of the BFX programs. The conceptual flow of control is different for the three BFX programs. Figure 13 is a block diagram for the BFXTI program. BFX Module Descriptions on page 44 describes each component of the block diagram.

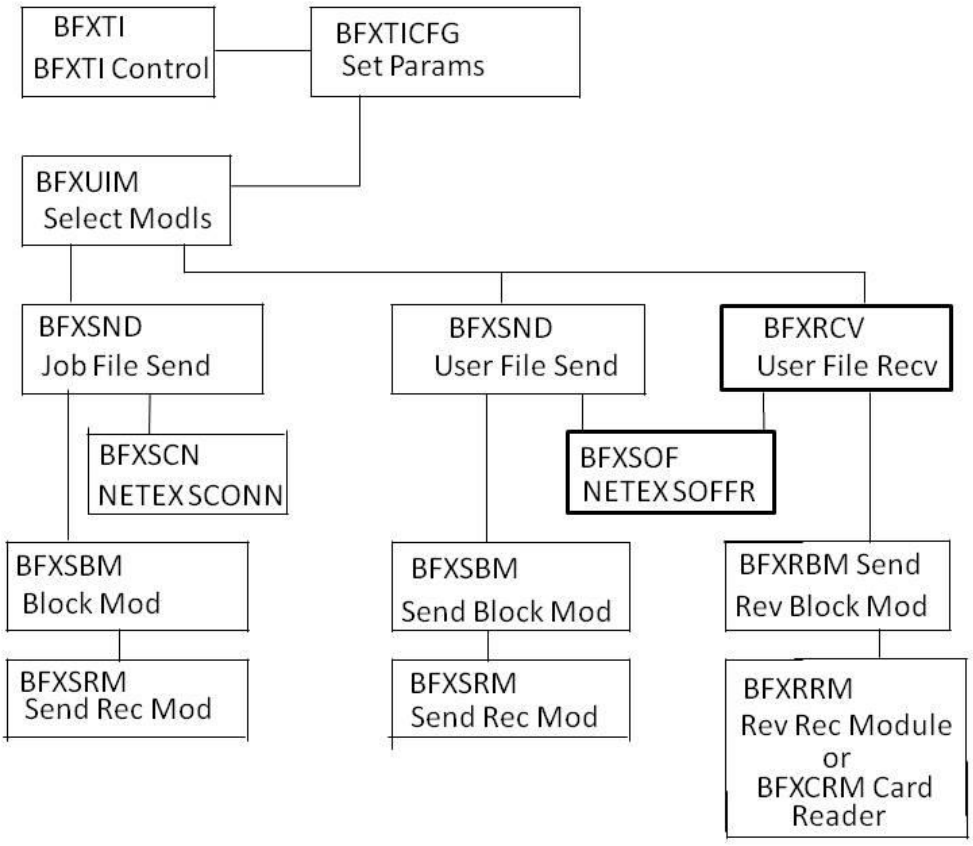

Figure 13. BFXTI Module Block Diagram

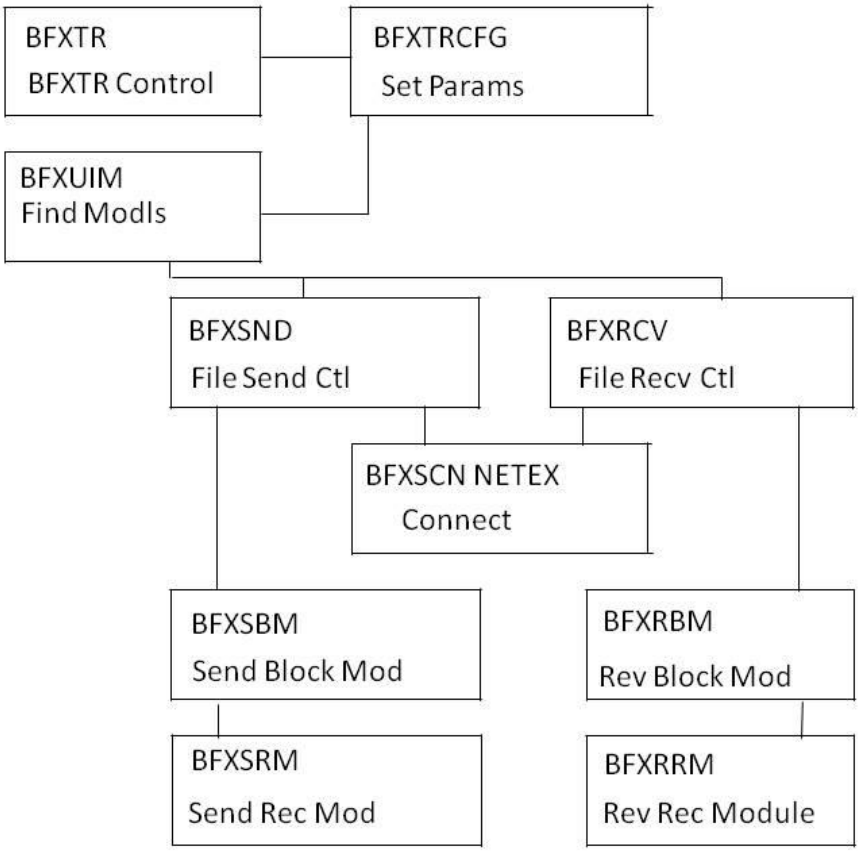

Figure 14. BFXTR Block Diagram

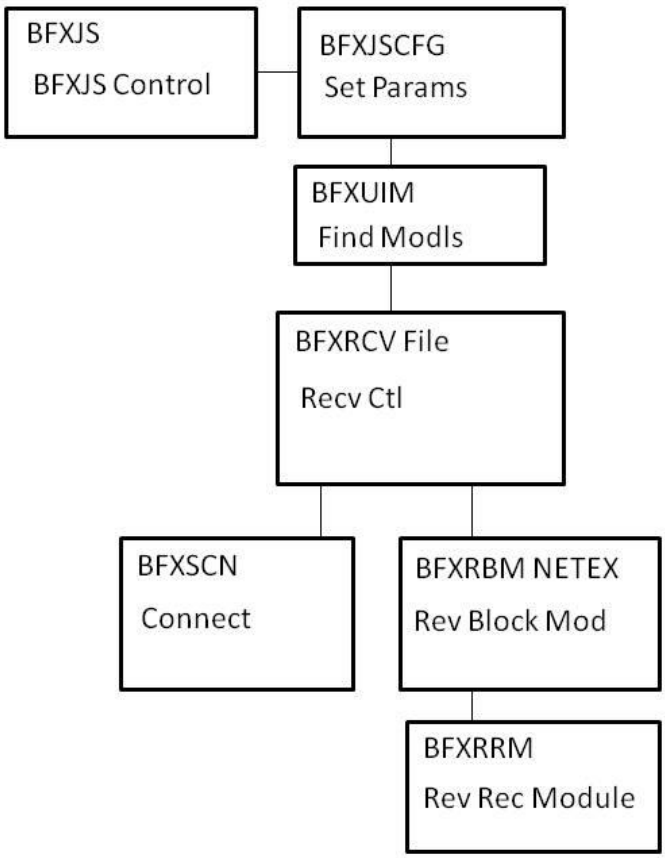

Figure 15. BFXJS Module Block Diagram

BFX Module Descriptions

The following paragraphs describe seven of the default BFX modules. The following modules (RCV, SND, RBM, SBM, RRM, SRM, and SUB) were introduced earlier in the block diagrams. Refer back to the block diagrams as necessary while reading the module descriptions.

BFX Receive File Control (RCV)

This module receives control from the TI, TR, or JS main modules when a file is to be transferred. The parameters to be ·used for the transfer are specified in the call sequence to the RCV module. Its flow of control is as follows:

1. The RCV module determines, from the passed parameters, whether to call the SOF or SCN module to OFFER or CONNECT to the other BFX program. These modules will resolve the buffer size, Least Common Multiple, Minimum Byte Count, and file Mode negotiations as detailed in the BFX General Design Specification.
2. The receiving Block Module is called for the first time. On a first time call, the Block Module will call the Record Module to open the file for output and verify and/or override the RMAXL record size parameter

3. Upon completion of the Block Module initialization, RCV begins the transfer process. It uses a multiple buffering technique to overlap NetEx/IP processing of the incoming record with the file writes performed by the Block Module. Figure 16. File Receive Data Flow on page 46 shows the normal flow of data transfer.
4. Whenever the Block Module is called to write a block received from NetEx/IP , the Block Module may return with a message. The message will be sent to the SND control module in the other application, and its contents will be placed on the system output of both applications. If the message indicates an abnormal end of transfer, RCV will send the message and wait for a Disconnect Indication from the other BFX to indicate acknowledgement of the error.
5. When an End of Information delimiter is received from the remote BFX, RCV will call the Block and Record Modules with the EOI indication.

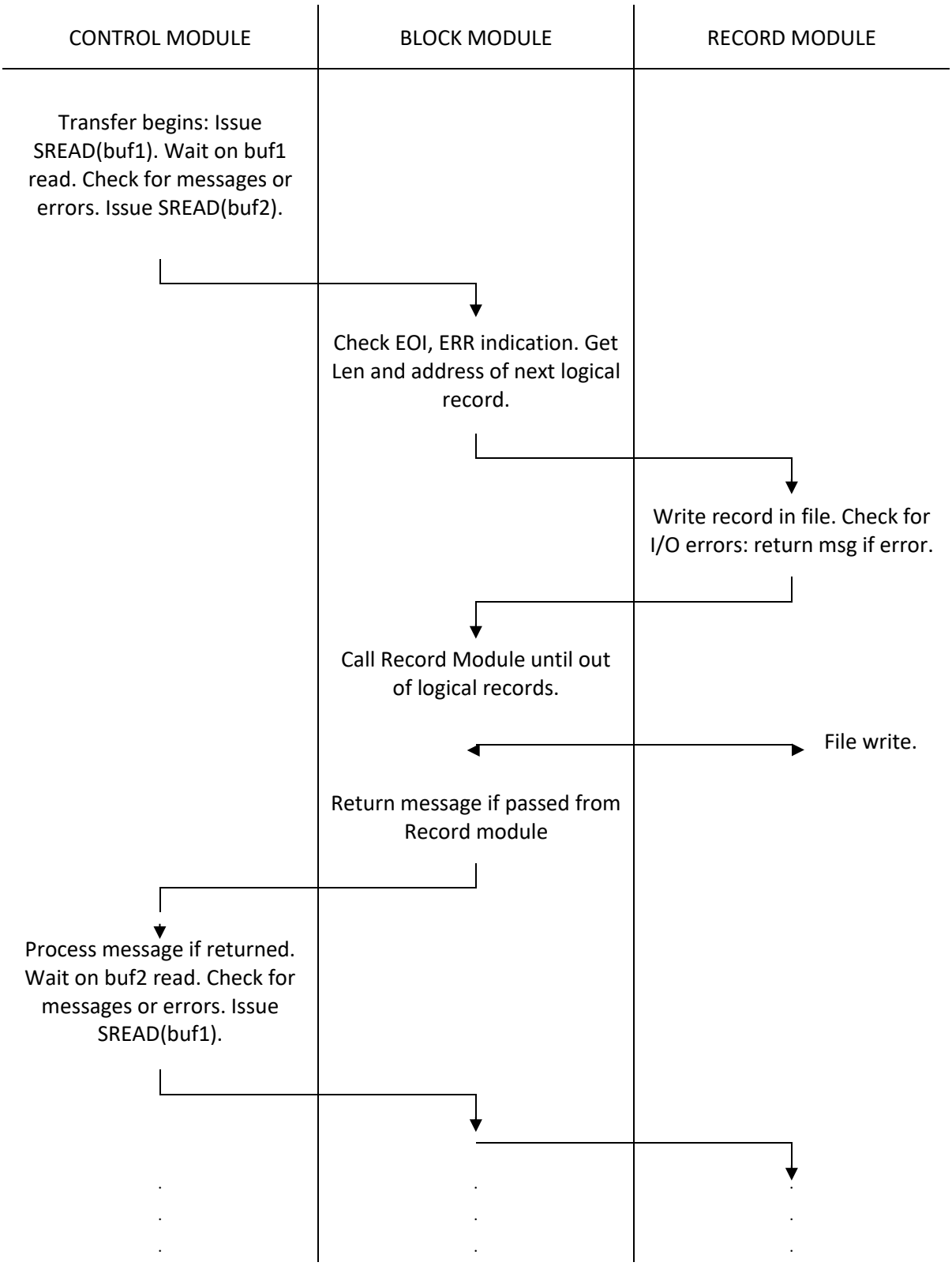

Figure 16. File Receive Data Flow

BFX Send File Control (SND)

This module receives control from the TI, or TR main modules when a file is to be transferred. The parameters to be used for the transfer are specified in the call sequence to the SND module. Its flow of control is as follows:

1. The SND module determines, from the passed parameters, whether to call the SOF or SCN module to OFFER or CONNECT to the other BFX program. These modules will resolve the buffer size, Least Common Multiple, Minimum Byte Count, and file Mode negotiations as detailed in the BFX General Design Specification. This initialization logic is illustrated in Figure 17. Send Receive Initialization Flow on page 48.
2. The receiving Block Module is called for the first time. On a first time call, the Block Module will call the Record Module to open the file for output and verify and/or ·override the RMAXL record size parameter.
3. Upon completion of connection negotiation, SND begins the transfer process. It uses a multiple buffering technique to overlap NetEx/IP processing of the incoming record with the file writes performed by the Block Module. Figure 18. File Send Data Flow on page 49 shows the normal flow of data transfer.
4. Whenever the Block Module is called to provide a block to be sent to NetEx/IP, the Block Module may return with a message. The message will be sent to the RCV control module in the other application, and its contents will be placed on the SYSPRINT log files of both applications. If the message indicates a normal or abnormal end of transfer, SND will send the message and wait for a Disconnect Indication from the other BFX to indicate acknowledgement of the ending indication.
5. If an abnormal end indication is received from the receiving remote BFX, SND will call the Block and Record Modules with an error indication. In that case, the Block and Record modules are to close the file and return without any additional data. SND will return to its calling module.
6. If the called Block Module returns an informational message, it will be forwarded to the opposite RCV module for logging. In addition, the message will be recorded locally on the system output. If an error or end of transfer message is returned from the Record Module, SND will forward the message and wait for a Disconnect Indication from the other side to acknowledge end of transfer.

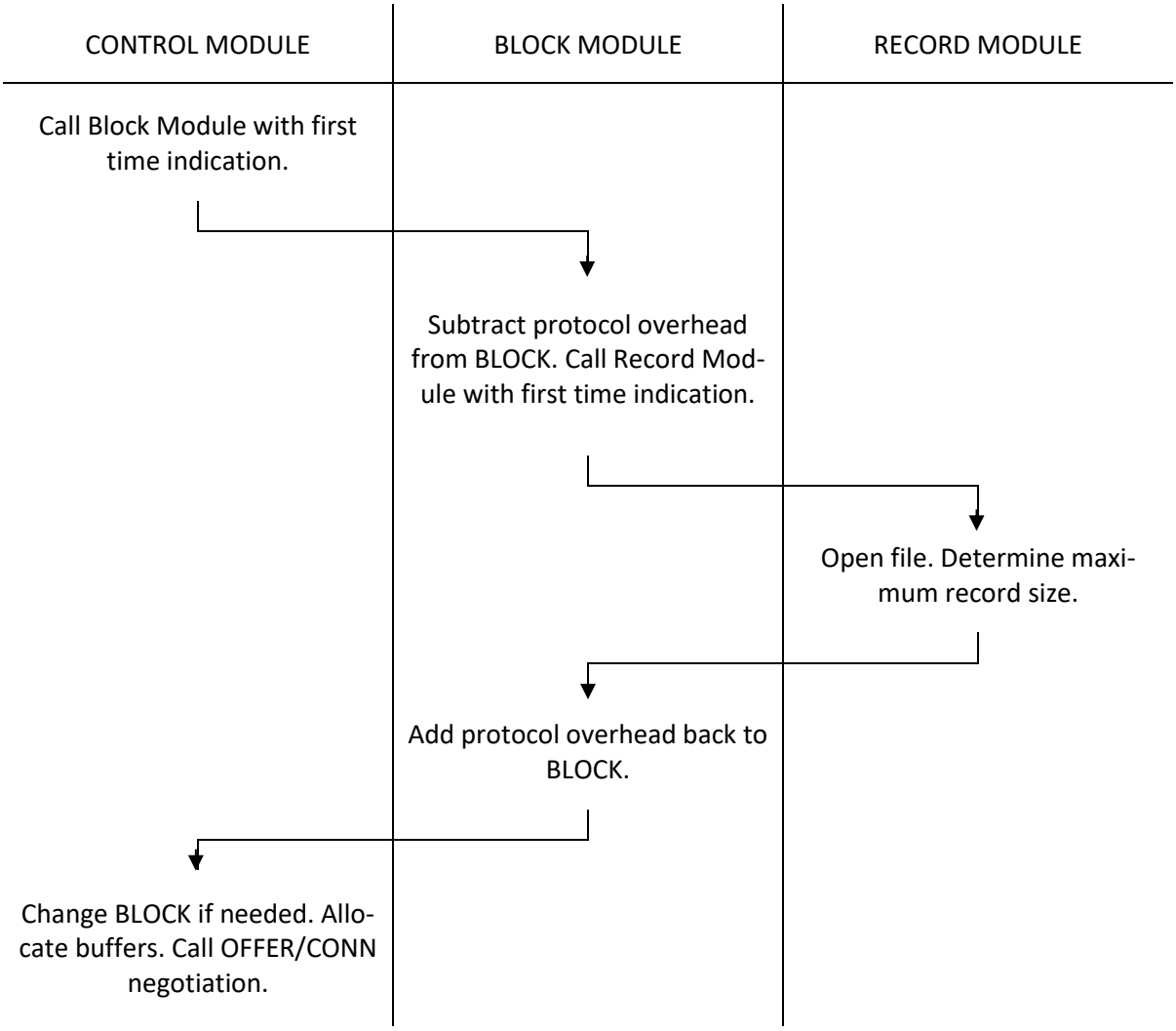

Figure 17. Send Receive Initialization Flow

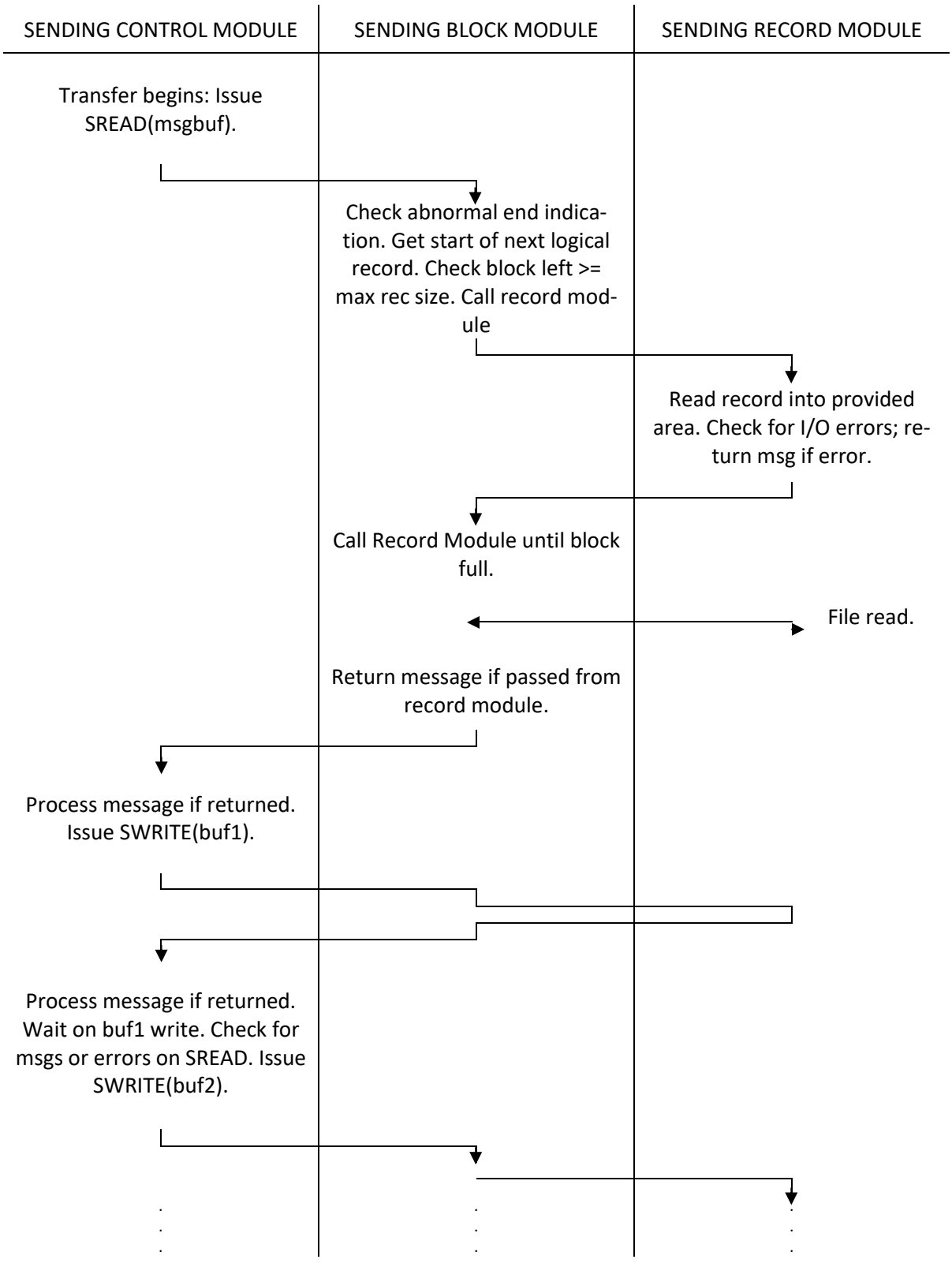

Figure 18. File Send Data Flow

Receiving Block Module (RBM)

This module is called once to open the file and establish communications. Afterwards, it is called each time a block of file data arrives from NetEx/IP. It is the responsibility of the receiving block module to decode all file transfer protocol information contained within the NetEx/IP data block. The block module will then call the receiving record module once for each logical record contained within the file. Overall logic flow for this module is as follows:

1. When entered, it checks the entry the first time. It examines the file MODE specified by the user and subtracts the appropriate header length from the BLOCK = value passed by the control module. It then calls the receiving record module for the first time providing the passed file parameters and the adjusted BLOCK= size.
2. On successful return from the receiving record module, the file will have been opened. Also returned by the record module is the maximum logical record length that will be encountered in the file. The block module will add the record header length to the maximum record length and return the value to the RCV module. At this point, the block module returns to the control module.
3. On the next entry to the block module, the first buffer of information from the sending block module will be provided. Based on the file MODE, the block module will decode the logical record information and call the record module once for each logical record. When the block is exhausted, the block module will return to receive another buffer.
4. If an end of information or terminal error indication is encountered in the incoming data stream, the block module will call the record module for the last time with an error or EOI indication. On return from the block module, the file should be closed and a termination message will normally be produced by the record module.
5. If the record module returns a message or abnormal end error, the block module will forward the message to the control module. If abnormal end was indicated it will free any allocated data areas since the module was called for the last time.

Sending Block Module (SBM)

The sending block module is responsible for providing blocks of file information to be sent to a receiving remote BFX application. When called the first time, it will call the sending record module to open the file and determine record and record size requirements. Its logic proceeds as follows:

1. When entered, it checks the entry the first time. It examines the file MODE specified by the user and subtracts the appropriate header length from the BLOCK = value passed by the control module. It then calls the sending record module for the first time providing the passed file parameters and the adjusted BLOCK= size.
2. On successful return from the sending record module, the file will have been opened. Also returned by the record module is the maximum logical record length that will be encountered in the file. The block module will add the record header length to the maximum and return the value to the SND module. At this point, the block module returns to the control module.
3. On the next entry to the block module, a buffer will be passed that is to be filled by the block module. The block module will insert block protocol information based on the file MODE and repeatedly call the record module until the remaining space in the block is too small to accommodate the maximum record size first declared by the record module. Record protocol information is added based on the negotiated LCM and the file MODE.
4. The record module may return an informational message, an abnormal end message, or an end of information message. If the message indicates an end of transfer, the record module will have already

closed the file. Forward the message to the control module and free any storage if an end message was passed up.

5. The control module may call with an abnormal end indication based on a loss of communications or an abort issued by the receiving BFX module. In that case, call the record module with an abnormal end indication. On return, the file should have been closed. Free any allocated storage areas and return to the control module for the last time.

Receiving Record Module (RRM)

The receiving record module is actually responsible for the QSAM I/O that will put the logical records in the file designated by the user. It opens the file, determines record format and logical record formatting requirements, and issues PUT macros to write the data when it is delivered by the block module. If I/O errors occur, it is responsible for analyzing the errors, determining if the error is fatal or not, and returning a comprehensible error message to the block module in the event of an I/O error.

The BFX default receiving record module operates as follows:

1. When the receiving record module receives control for the first time, it opens the file whose DDNAME is passed to it from the block module. Using the LRECL parameter produced as a result of the OPEN, it returns the maximum logical record length to the block module. If the specified BLOCK= parameter is not large enough to accommodate the largest logical record, then the RCV module will adjust the value of BLOCK upwards. If the open for the file does not succeed, then the record module will return an open failure message to the block module for logging.
2. Subsequent calls to the record module will provide the address and length of the logical record. The record module will PUT (WRITE for VBS format) this record to the file, adding V-format header information if needed. If record type conversion is called for, then it will truncate or pad the record before it is written.
3. If the record module is called with an error or EOF indication, then the record module will close the file and return to the block module.

Sending Record Module (SRM)

The sending record module is responsible for reading logical records from the file to be sent over the network. On the first call, it will open the file for input. Subsequently, it will provide a logical record every time it is called. When end of file is reached or a permanent error is detected in reading the file, then it will close the file, generate a termination message, and return.

SRM flow of control proceeds as follows:

1. When the sending record module receives control for the first time, it opens the file whose DDNAME is passed to it from the block module. Using the LRECL parameter produced as a result of the OPEN, it returns the maximum logical record length to the block module. If the specified BLOCK= parameter is not large enough to accommodate the largest logical record, then the SND module will adjust the value of BLOCK upwards. If the open for the file does not succeed, then the record module will return an open failure message to the block module for logging.
2. Subsequent calls to the record module will provide the address and length in which the logical record is to be placed. The record module will GET the record into this block module provided area.
3. If the record module is called with an error or EOF indication, then the record module will close the file and return to the block module.

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

BFX006S "xxxxxxxx" NOT RECOGNIZED IN CONTROL STATEMENT.
Severity: 15 (Severe 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 re-run 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.

BFX017IC JOB: nnnnnnnnnnnnn
Severity: Continued informational)
Explanation: When sending a secured jobfile, the job card is displayed.
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: 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 BFX job 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.

BFX025F REMOTE BFX ABORTED EXECUTION.
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 that the Secure BFX program on the remote host terminated abnormally. Processing is terminated, as no further data transfer is possible.
User Response: Examine the output from the job on the remote host to determine the cause of failure. Correct the error and re-submit the job.

BFX026F REMOTE HOST NetEx NOT PRESENT.
Severity: 15 (Fatal Error)
Explanation: When an attempt was made to connect to the remote Secure BFX program, the Secure NetEx/IP on the local machine reported that no Secure NetEx/IP (SNXMAP component) was present on the remote host.
User Response: Consult with operations to determine whether Secure NetEx (SNXMAP component) should have been present on the remote host. Re-submit the job when Secure NetEx is available on both hosts.

BFX027F SPECIFIED HOST IS NOT ON THE NETWORK.
Severity: 15 (Fatal Error)
Explanation: When BFXTI was attempting to connect to BFXJS, or when BFXTR was attempting to connect back to the initiating BFXTI, the 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/CONNDELAY in between each attempt. TI connecting to JS will try JREPEATCONN # of times and delay DELAYTIME/CONNDELAY 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/CONNDELAY and re-submit the job.

BFX033S NO RESPONSE FROM REMOTE BFX PROGRAM. – *Note: Now Deprecated.*
Severity: 12 (Severe Error)
Explanation: The local BFX program expected to receive data or a message from the remote BFX program, but none was received within a reasonable time. The probable cause of this error is that the remote BFX program has become "hung", either because of a programming error, or because of very slow response times on the remote host.
User Response: If response times are slow on the remote host, it may be necessary to specify a higher value for READTIMEOUT and re-submit the job. If a programming error is suspected and user-written modules are in use, ensure that they are not at fault.

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

BFX036W Connection refused, [retrying after 5 secs] [retrying next address].
Severity 4 (Warning)
Explanation: Secure BFX attempted a TCP connection for a secure transfer. The remote system was not able to process the connection request. The system will attempt a retry after 5 seconds. If repeated attempts fail the request will try the next IP address if specified. If all addresses have been tried the request will fail.
User Response: None. Determine if BFXSJS is processing on the remote systems. If not, start bfxsjs.

BFX037I Deprecated nnnnnnn
Severity: 9 (Informational)
Explanation: The keyword listed has been deprecated. The job continues to process, ignoring the keyword parameter.
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.

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

BFX043F Secure 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 Secure 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 Secure 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.

ERRTAB	Status Code (Octal)	Description
	0	Request processed normally
IMPMSG	1	Improper run-stream in file
NOTASG	2	File access denied
NOTPFS	3	Element unobtainable
NOFILE	4	No file specified
IOMSG	5	I/O error encountered
TAPTMP	6	File not cataloged on mass storage
FILFRE	7	File freed while processing @START
XOPTVIOL	10	X option security violation
UOPTVIOL	11	U option requires SENTRY configured
MISSFLD	12	User-id and account needed on @RUN statement
ACCDENIED	13	Started denied access to target user-id
OWNERNOT	14	Target user does not own @START file
CPYIOMSG	15	Copy I/O error (plus err code appended to end)
CPYFIMSG	16	Copy file unobtainable (plus fac status appended to end)
CPYBFMSG	17	PCT full. Cannot copy runstream
ZOPTPRIV	20	Z option requires starter to have Z-option privilege
ZOPTXHGH	21	Z-option requires user-id to have system-high capability
SUMODMSG	22	Contact site administrator and report @START command internal error 01
FMMSW5	23	STD mass storage tight. Cannot start run
IUCSF	24	The user-id is invalid for an Exec-initiated @START
NUCSF	25	The user-id does not exist

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

BFX055S SecureHost $27 ignored -- length > 8 -- number remains at $30.
Severity: 15 (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.

BFX056S SecureHost $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 secure and using nosechost parameter to exclude the system(s) you do not want secure transfers to occur to.

BFX057S NoSecureHost $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.

BFX058S NoSecureHost $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 secure and using sechost parameter to include the system(s) you want secure transfers to occur to.

BFX059W Nosecure cannot be specified, once secure has been set – sent securely
Severity: 12 (Warning)
Explanation: Once a secure transfer has been determined, the user cannot specify nosecure. 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 secured.
Severity: 8 (Informational)
Explanation: A secured connection will be used to transfer the file.
User Response: None

BFX062I Connection will be non-secured.
Severity: 8 (Informational)
Explanation: A non-secured connection will be used to transfer the file.
User Response: None

BFX063F Secure job submission failed because BFXJS could not access the authorized users file.
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.

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.

BFX084S PROTOCOL ERROR – DATA RECEIVED WHEN SENDING.
Severity: 15 (Severe 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.

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 incorrect user-written block module. Correct the block module and re-submit the job.

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 fffffffffff PERMANENT I/O ERROR. RC = rr.
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: Errors were detected during the transfer of the file. Secure BFX will stop the batched transfer of files.
User Response: Correct the error encountered as described by the previous log messages. Re-run the Secure BFX jobs.

BFX110S Cannot open input file nnnnnnnnnn
Severity 12 (Severe error)
Explanation: System error code is shown in message.
User Response: Correct the file the error reported by the system and re-run the job.

BFX111S RECORD MODULE INITIALIZATION FAILED.
Severity: 12 (Severe error)
Explanation: A user-written 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 user-written Record Module.

BFX113S SENDING RECORD MODULE ABORTED TRANSFER.
Severity: 12 (Severe error)
Explanation: During the transfer of a file, a user-written 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 user-written module and re-submit the jobs.

BFX114S RECEIVING RECORD MODULE ABORTED TRANSFER.
Severity: 12 (Severe error)
Explanation: During the transfer of a file, a user-written 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 user-written module and re-submit the jobs.

BFX115S SENDING BLOCK MODULE ABORTED TRANSFER.
Severity: 12 (Severe error)
Explanation: During the transfer of a file, a user-written 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 user-written module and re-submit the jobs.

BFX116S RECEIVING BLOCK MODULE ABORTED TRANSFER.
Severity: 12 (Severe error)
Explanation: During the transfer of a file, a user-written 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 user-written module and re-submit the job

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 BFX programs so it is sufficient to transfer the file. Note that a header of 6 bytes (bit mode) or 8 bytes (character mode) is prefixed to each record transferred.

BFX122F INVALID FILE TRANSFER PROTOCOL INFORMATION.
Severity: 15 (Terminal error; ABEND will follow)
Explanation: The file transfer information sent to the receiving Secure BFX was found to be incorrect. This may be because of an internal Secure BFX error, internal Secure NetEx/IP API error, or because of a user-written Block Module that is incorrectly sending data to the standard Receiving Block Module. As this error could be very serious, the Secure BFX program will unilaterally ABEND.
User Response: If the error was caused by a user-written Block Module, correct the coding error that caused incorrect data to be sent. If no user-written Block Module was used, bring the error to the immediate attention of Network Executive Software Customer Support.

BFX123F FILE TRANSFER PROTOCOL SEQUENCE ERROR.
Severity: 15 (Fatal error)
Explanation: The file transfer information sent to the receiving BFX was found to be incorrect. A record numbering check indicated that records are missing, duplicated, or out of sequence. This may be due to an internal Secure BFX error, an internal Secure NetEx/IP API error, or to a user-written Block Module that is incorrectly sending data to the standard Receiving Block Module. The current transfer is aborted, but the remaining transfer will be attempted.
User Response: If the error was caused by a user-written Block Module, correct the coding error that caused incorrect data to be sent. If no user-written Block Module was used, bring the error to the immediate attention of Network Executive Software Customer Support.

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

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

BFX128S Rmaxl too small for record length.
Severity: 12 (Severe)
Explanation: The record to be transferred is larger than the specified RMAXL parameter.
User Response: Increase the RMAXL if the record size is larger than RMAXL

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. See EXEC SYSTEM SOFTWARE EXECUTIVE REQUESTS PROGRAMING REFERENCE MANUAL @START for a complete list of errors and their meanings. Some are:

1. Improper runstream file
2. File access denied
3. Element unobtainable
4. No file specified
5. I/O error

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 [SECURED] FILE fffffffffff. [TO nnnnnnnnnn]
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. If this is a secured file transfer SECURED is added to the message, along with the remote host name.
Response: None.

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

BFX222F IBMTAP Record module requires BIT mode.
Severity: 15 (FATAL)
Explanation: When transferring an IBM tape, using the IBMTAPE record module, the file must be done in MODE=BIT
User Response: Correct the job and re-run.

BFX223F IBMTAP file fffffffff needs to be a tape file.
Severity: 15 (Fatal)
Explanation: When transferring file fffffffff, using the IBMTAPE record module, the file must reside on tape.
User Response: Correct the job and re-run.

BFX224F IBMTAP file ffffff cannot have Q flag set.
Severity: 15 (Fatal)
Explanation: The tape must be assigned without the Q flag (",T//////Q") and must use MODE=BIT. This will allow tape blocks greater than 9,999 characters. If the Q flag were set and MODE=EBCDIC, tape blocks would be limited to 9,999 characters.
User Response: Correct the job and re-run.

BFX225F BIGBLK encountered an invalid state ssssssss.
Severity: 15 (Fatal)
Explanation: An unexpected error occurred when processing the tape.
User Response: Notify Network Executive Software support.

BFX226F IBMTAP - Tape not positioned at label record.
Severity: 15 (Fatal)
Explanation: When transferring file fffffffff, using the IBMTAP record module, the program should have encountered a tape label. The label was not found.
User Response: Verify the tape is a properly labeled tape and re-run the job.

BFX227I IBMTAP - Tape vvvvvv read..Swapping to next tape ttttttt.
Severity: 4 (Informational)
Explanation: Secure BFX has completed reading the tape with volume-id vvvvvv and needs to switch to the next tape specified for a multi-volume transfer.
User Response: Mount the next tape to allow the job to continue processing.

BFX228I IBMTAP - Tape vvvvvv completes file ffffff.
Severity: 4 (Informational)
Explanation: The end of file marks were encountered on the tape. The file transfer has been completed.
User Response: No response required.

BFX230I HOB CHECK REC nnn WORD nn: ddddddddd.
Severity: 15 (Informational)
Explanation: When HOBCHECK CHECKONLY is used and words with bit 9 set are deleted, the bad word, as well as the record and word number are displayed. See the HOBCHECK discussion for details on this feature.
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 “sssssss”, which is the JID or ID parameter specified in an input statement. The block size that the offering program would like use is “bbbb”.
User Response: None.

BFX302I CONNECTING TO ssssssss ON HOST hhhhhhhh; BLOCK IN SIZE nnnn.
Severity: 2 (Diagnostic).
Explanation: When BFXTR is connecting back to its starting BFXTI, it has issued a 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.

BFX303I CONNECTING TO hhhhhhhh; at pppp.
Severity: 2 (Diagnostic).
Explanation: The Secure BFX program has issued a Secure NetEx/IP SCONNECT with a previously offered Secure BFX program. "hhhhhhhh" is the host name specified in a HOST parameter statement. "pppp" is the IP address/port.
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 NetEx/IP block size that will be used during file transfer. "dddd" is four hexadecimal digits that give the 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.

BFX320F BAD CONNECT DATA RECEIVED.
Severity: 15 (Informational)
Explanation: Secure BFX connect/confirm protocol from remote job was not correct.
User Response: Trace data for reason. Verify the MODE = parameter is set to the same value in the sending and receiving jobs.
Response: Contact Network Executive Software Customer Support for assistance if necessary.

BFX399W WARNING: TAPE ASSIGNED FORMAT-A.
Severity: 15 (Informational)
Explanation: The tape was assigned in the A-Format mode for the tape drive.
User Response: Verify this is the mode the tape was written in.

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 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 filed itself. The string is truncated to fit in the field. The maximum length of the filed in question is printed following the message. Processing will continue normally.
User Response: Unexpected results may occur because of this truncation. If so, supply a valid length string and 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.

BFX411F SECURE BFX IMPROPERLY BUILT.
Severity: 15 (Fatal error).
Explanation: The Secure BFX distribution installed was incorrectly built.
User Response: Contact Network Executive Software Support personnel to receive a new distribution.

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.

BFX415S PAD NOT SPACE.
Severity: 12 (Severe).
Explanation: "SPACE" is currently the only valid pad character.
User Response: Correct or remove the control card..

BFX416S PAD requires MODE = ASCII or EBCDIC on a Disk.
Severity: 12 (Severe).
Explanation: To use the PAD feature, the file must reside on a sector disk, and the transfer mode must be ascii.
User Response: Remove the control card or set the mode=ascii and ensure the file is on a sector disk.

BFX500F INVALID HEADER RECEIVED.
Severity: Fatal error)
Explanation: Secure BFX received a packet that did not start with a valid header. The transfer is terminated, and the header received is displayed.
User Response: Save the job output and contact Network Executive Software Customer Support.

BFX501E nnnnnnnnnnnn.
Severity: Error
Explanation: Secure BFX issued a call nnnnnnnn to the UNISYS System Communications API. The API returned an error condition. The error code is display in decimal, and octal. It is broken down to display an error number and an auxiliary status. These are also displayed in decimal an octal. The meaning of these codes can be found in the following UNISYS manuals:

- Appendix A of "COMMUNICATIONS APPLICATION PROGRAM INTERFACE (COMAPI) USER'S GUIDE".
- Section 3.10 of "COMMUNICATIONS PLATFORM PROGRAMMING REFERENCE MANUAL"
- Section 3.10 of "COMMUNICATIONS PLATFORM FOR OPEN SYSTEMS PROGRAMMING REFERENCE MANUAL"

Some of the error codes are duplicated, so be sure to check all sections.
User Response: Some of the more common error conditions will have a suggested resolution printed at the end of the error codes. These may include "CHECK BFXJS ON REMOTE HOST", "CHECK THAT COMAPI IS STARTED" and "RECEIVED CLOSE EVENT FROM REMOTE SIDE".

BFX502I End of Secure BFX global configuration inputs
Severity: Informational
Explanation: The preceding Secure BFX parameters were in the global configuration file. The parameters following were in the local run stream.
User Response: None.

BFX503E INVALID LINE IN CONFIGFILE (KEY = VALUE) nnnnnnnnnnn
Severity: Error
Explanation: While processing the configuration files, an invalid line was encountered. The format of the configuration file is KEY = VALUE Correct the configuration file and re-run the job.
User Response: Correct the input.

BFX504E GETOPTS HOST %s (CHECK DNS)
Severity: Error
Explanation: When processing a TCP SETOPTS command to the host, and error was reported.
User Response: Verify the Host name is valid, and check the BFX500 messages for the decoding of the error condition.

BFX505E INVALID GLOBAL/LOCAL CONFIGURATION PARAMETER(S) ENCOUNTERED
Severity: Error
Explanation: One or more errors were detected while processing the local or the global configuration files.
User Response: Correct the input.

BFX506E nnnnnnn mmmmmmmmm
Severity: Error
Explanation: While processing the configuration file, parameter nnnnnn was encountered. The value entered was invalid. mmmmmmm lists the format of the valid values.
User Response: Correct the input.

BFX507E UNKNOWN HOST nnnnn (CHECK DNS)
Severity: Error
Explanation: Secure BFX is attempting to transfer a file to HOST nnnnnnnn. DNS returned an error condition. Check that the HOSTNAME is correctly installed in either the local host file on this system or the DNS server is correctly configured.
User Response: If DNS is correctly configured, contact Network Executive Software Support.

BFX508I nnnnnmn not found using mmmmmmm
Severity: Informational
Explanation: A Secure BFX transfer tried to resolve the hostname nnnnnnnn. DNS returned a not found condition. Secure transfer will try again using hostname mmmmmmmm.
User Response: None

BFX509I IP ADDRESS =
Severity: Informational
Explanation: A Secure BFX transfer received this/these IP addresses from DNS. It will use this addresses in the connection attempts.
User Response: None

BFX510I CONNECTED ON IP ADDRESS =
Severity: Informational
Explanation: A connection to the remote was successful using the listed IP address.
User Response: No action is necessary.

BFX511E ALL CONNECTION ATTEMPTS HAVE FAILED
Severity: Error
Explanation: All connection attempts to the remote host have failed. See previous messages for failure reasons. The job is terminated.
User Response: Correct errors previously listed and re-run the job.

BFX512E Delete File nnn*nnn and re-run
Severity: Error
Explanation: SSL open connections must be single thread. This file ensures this occurs. The file did not get deleted after the connection was open.
User Response: Delete the file and re-run the job.

BFX513E INVALID SEQUENCE RECEIVED EXPECTED: nn
Severity: Error
Explanation: The Secure BFX transfer received a block of data with an incorrect sequence number. The transfer is aborted. The sequence number expected is displayed, and the BFX header is printed.
User Response: Contact Network Executive Software Customer Support.

BFX517E NEWHOST IS NOT SUPPORTED FOR SECURE TRANSFERS
Severity: Error
Explanation: The NEWHOST parameter is not supported for secure transfers.
User Response: Replace the NEWHOST with a TO= or FROM= and break the run stream into two separate executions of the Secure BFX program.

BFX518E KEYIN (%s) FAILED STATUS O-%o FLAG O-%o
Severity: Error
Explanation: While attempting to register the KEYIN with the system, an error occured. The return value is displayed.
User Response: Verify the KEYIN value starts with a character, is less than 9 characters, and is not already registered on the system. See ERCODE 0244 for additional information.

BFX519I BFXJS IS RESPONDING TO KEYIN: %s
Severity: Informational
Explanation: The BFXJS application has register the key in with the system. To terminate BFXJS use this value and the command "TERM".

BFX520I BFXJS TERMINATION REQUESTED
Severity: Informational
Explanation: The operator request BFXJS to terminate

BFX521I BFXSJS INVALID COMMAND

Severity: Informational
Explanation: The operator entered an unknown command using the BFXJS KEYIN
User Response: "TERM" is currently the only supported command

BFX522I BFXJS ERROR sending to Console %o
Severity: Informational
Explanation: While attempting to write to the console the ER KEYIN returned error status
User Response: See ERCODE 0244 for additional information.

TSB601I File $78 volume $26, $91 records sent.
Severity: 15 (Informational).
Explanation: In a tape transfer, the end of tape was encountered. The file name, the volume serial number and the total number of records sent are displayed.
User Response: None.

TSB602I File $78 volume $26, $91 records received.
Severity: 15 (Informational).
Explanation: In a tape transfer, the end of tape was encountered. The file name, the volume serial number and the total number of records received are displayed.
User Response: None.

TSB603I Sending reel set format-A.
Severity: 15 (Informational).
Explanation: The USSENDER rmod is transferring a tape file using A-format rules. This occurs when the BFX program is running under the "LETLABEL" accounting code and the program received an abnormal frame count that was not a 0 or a 5 while reading the tape file.
User Response: None.

TSB605I Sending volume $26 swapped, $27 blocks sent
Severity: 15 (Informational).
Explanation: In a tape transfer, the end of volume was encountered before the end of file. A new mount request will be issued. The current record count is listed.
User Response: None.

TSB606I Receiving volume $26 swapped, $27 blocks received
Severity: 15 (Informational).
Explanation: In a tape transfer, the end of volume was encountered before the end of file. A new mount request will be issued. The current record count is listed.
User Response: None.

TSB607I Sending reel set format 8-packed.
Severity: 15 (Informational).
Explanation: The USSENDER module is running under an accounting code that does not have label access. The file is sent using C-Format rules. If an abnormal frame count is received other than 0 or 5, the transfer will be restarted in Q-Format.
User Response: None.

TSB608I Tapemark detected on sending reel nnn blocks
Severity: 15 (Informational).
Explanation: The USSENDER module encountered a tapemark. The number of blocks transferred is displayed.
User Response: None.

TSB609I Tapemark detected on receiving reel nnn blocks
Severity: 15 (Informational).
Explanation: The USRECVR module encountered a tapemark. The number of blocks transferred is displayed.
User Response: None.

TSB610E RECEIVER Tape must be assigned in 8-PACKED mode
Severity: 15 (ERROR).
Explanation: The USRECVR module detected a Q-format assign mode. It must be 8-PACKED. USRECVR will change the tape mode to Q-format if required.
User Response: Assign the tape in 8-PACKED mode.

TSB612I Bfx using UCSTLBL for labels.
Severity: 15 (Informatonal).
Explanation: The BFX application is using the UCSTLBL macro to process tape labels
User Response: None.

TSB613I Bfx is not authorized for labels
Severity: 15 (Informational).
Explanation: The usermode cannot access the tape labels through FIOW maco
User Response: None

TSB614I Bfx is authorized for tape labels
Severity: 15 (Informational).
Explanation: The BFX program has authority to process tape label through FIOW
User Response: None.

TSB664S Not under Privilege Account
Severity: 15 (Error).
Explanation: The receiving tape job received a data record containing VOL1 in the first four characters. The job was not started under the "LETLABEL" accounting code. The job is terminated.
User Response: Alter the accounting code.

TSB690I PDAY: nnnn ID: nnnnnn CURRENCY:00
Severity: 15 (Informational).
Explanation: When processing tape labels a user header label was encountered. Selected fields are printed.
User Response: None.

TSB691I COUNTRY: nnnnnn FTYPE n DAILY FNUM: nnn
Severity: 15 (Informational).
Explanation: When processing tape labels a user header label was encountered. Selected fields are printed.
User Response: None.

TSB692I UHL1 nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
Severity: 15 (Informational).
Explanation: When processing tape labels a user header label was encountered. The label is printed.
User Response: None

TSB693I NUM OF DEBITS nnnnnnn VALUE nnnnnnnnnnn.nn
Severity: 15 (Informational).
Explanation: When processing tape labels a UTL1 label was encountered. Selected fields are printed.
User Response: None.

TSB694I NUM OF CREDITS nnnnnnn VALUE nnnnnnnnnnn.nn
Severity: 15 (Informational).
Explanation: When processing tape labels a UTL1 label was encountered. Selected fields are printed.
User Response: None

TSB695I UTL1nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
Severity: 15 (Informational).
Explanation: When processing tape labels a UTL1 label was encountered. The label is printed.
User Response: None

TSB696I MULTIREEL: SWAPPING FROM vvvvvv TO vvvvvv
Severity: 15 (Informational).
Explanation: The input tape was a multireel tape. At EOV a tapeswap occurs. Both the old volser and the new volser are displayed.
User Response: None

TSB697I HDR2 INFO: FORMAT n BLOCK nnnnn RECORD nnnnn
Severity: 15 (Informational).
Explanation: When processing tape labels a HDR2 label was encountered. Selected fields are printed.
User Response: None.

TSB698I VOLSER: CHANGING TSN FRON nnnnnn TO nnnnnn TYPE: VOL1 nnnnnnnn
Severity: 15 (Informational).
Explanation: When processing tape labels a VOL1 label was encountered. The old volser and the new volser numbers are printed, as well as the character set of the VOL1 label.
User Response: None.

TSB699F FATAL: RECORDS NOT MULTIPLE OF TAPE BLOCK
Severity: 15 (FATAL).
Explanation: Tape records must reside in the same physical block on tape.
User Response: None

TSB700I NUM OF AUDDIS RECORDS nnnnnn
Severity: 15 (Informational).
Explanation: When processing tape labels a UTL1 label was encountered. The label is printed.
User Response: None

TSB702I REELID: USING TSN *volser* on FILE *xxxxxxx*
Severity: 4 (Informational)
Explanation: The output tape is writing on the named volser, for the output file
User Response: None.

TSB705F ONLY SPERRY/RBM ALLOWS RPARM=BIGBLK
Severity: 15 (Fatal)
Explanation: RPARM=BIGBLOCK was encountered. The SPERRY or the RBM parameters were not coded.
User Response: Add either SPERR or the RBM parameter or delete the BIGBLOCk parameter.

TSB706F ONLY TAPE FILES ALLOW RPARM=BIGBLK
Severity: 15 (Fatal)
Explanation: RPARM=BIGBLOCK was encountered. The file to be transferred is not a tape file
User Response: Correct the parameters.

TSB707F FATAL: RMAXL MUST EXCEED 2 FOR RPARM=BIGBLK
Severity: 15 (Fatal)
Explanation: RMAXL was set to 1 or 2. RMAXL must be larger than 2 to run BIGBLK
User Response: Correct the parameters

TSB708F FATAL: BIGBLK HAS SEGMENT SEQUENCE ERROR
Severity: 15 (Fatal)
Explanation: BIGBLK detect segment error when receiving a file
User Response: Contact Network Executive Software Customer Support

TSB709F FATAL: FORMAT-A DATA TO FORMAT-C TAPE
Severity: 15 (Fatal)
Explanation: User is transferring A-Format data and the tape is assigned in C-Format mode.
User Response: Correct the tape drive assignment on the receiving job

TSB710I BIGBLK AUTO-ACTIVATED WITH RMAXL =
Severity: 4 (Informational)
Explanation: User is transferring a tape to a second Unisys system. Bfx will use BIGBLK. The autoactivation message will always appear, Even when bigblk was requested.
User Response: None

TSB711I BIGBLK ENDED
Severity: 4 (Informational)
Explanation: BigBlk has completed the file transfer
User Response: None

TSB713F BIGBLK PROTOCOL RECEIVED IN NON BIGBLK MODE
Severity: 12 (Severe)
Explanation: The receiving side of the transfer was not running in BIGBLK mode, but received protocol from the BIGBLK record module.
User Response: Verify both sides are running the BIGBLK record module.

TSB714F BIGBLK in 8-PACKED mode detected a Q-FORMAT tape
Severity: 15 (Severe)
Explanation: BIGBLK received an abnormal frame count other than 0 or 5. This indicated the tape was written in Q-format. The job is terminated.
User Response: Assign the tapes in 8-PACKED mode and re-run.

TSB715E Unable to MULTIREEL CREOV=*vvvvv*
Severity: 15 (Error)
Explanation: Tape labels were not available in order to create EOV records on the receiving side.
User Response: Insure a standard label tape was used as the sending tape.

TSB716E Going to MULTIREEL CREOV=*vvvvv*
Severity: 4 (Informational)
Explanation: The receiving side requires an additional reel of tape. EOV labels will be created and a new tape will be used.
User Response: None.

TSB719E CREOV Invalid: Needs a format-A tape file.
Severity: 15 (Error)
Explanation: The CREOV option requires the tape to be assigned in A-MODE.
User Response: Remove the CREOV option or change the assign on the output tape.

TSB720E CREOV Invalid: needs MODE=OCTET and a recsize or the RBM parameter.
Severity: 15 (Error)
Explanation: The CREOV option requires the MODE to be OCTET or it requires the RBM parameter.
User Response: Correct the run stream and re-run the job.

TSB721E CREOV Invalid: Tape must be assigned UNLABELED.
Severity: 15 (Error)
Explanation: The tape was assigned with standard labels. It must be assigned as unlabeled.
User Response: Correct the tape assignment.

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

BFX900I Hxx5 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

BFX901F BIT-STRING RECORD SIZE MUST BE A SECTOR MULTIPLE.
Severity: 15 (Informational – always issued).
Explanation: Indicates that an I/O other than the last one is not a multiple of 28 words.
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 IS 9999 BYTES.
Severity: 15 (Severe error)
Explanation: The maximum value of RMAXL is 32,767 in CHARACTER mode. A record was read that exceeded 32,767 bytes during a CHARACTER mode send.
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.

BFX909S Labels and DSLAB are mutually exclusive
Severity: 12 (Severe)
Explanation: Labels and don't send labels were specified.
User Response: Select the option you want.

BFX911I END OF FILE nnnn: TRANSFERRED rrrr RECORDS.
Severity: 4 (Informational)
Explanation: nnnn is the file number; rrrr is the number of records.
User Response: None.

BFX914S RECORD LENGTH MUST BE AT LEAST 40 TO PROCESS LABELS.
Severity: 12 (Severe error)
Explanation: The record length must be at least 40 to process tape labels.
Response: Increase RMAXL to at least 40.

BFX916F TRACKIO REQUIRES AT LEAST 1792 WORD RECORDS.
Severity: 12 (Severe error)
Explanation: At least 1792 word records are required for TRACKIO.
Response: Increase RMAXL to at least 1792.

BFX917S TAPE LABELING ERROR ee SUBSTATUS ss.
Severity: 12 (Severe error)
Explanation: Error code given to ER TLBLS by the exec.
User Response: Check 2200 Programmer's Reference for TLBLS error codes

BFX918S TSWAP error =.
Severity: 12 (Severe error)
Explanation: An error occurred while mounting a new tape volume.
User Response: Check 2200 Programmer's Reference for TLBLS error codes

BFX919S RBM and LABELS are mutually exclusive
Severity: 12 (Severe error)
Explanation: These options cannot be used together.
User Response: Specify RBM or LABELS; not both

BFX920S Illegal option coded without Sperry parameter
Severity: 12 (Severe error)
Explanation: A parameter was coded that requires the SPERRY parameter
User Response: Correct the parameters

BFX921S Word files are not supported
Severity: 12 (Severe error)
Explanation: Files must be allocated in sectors.
User Response: Unisys word files are not supported. Recreate the file in sector format.

BFX922I Calling File Open
Severity: 4 (Informational)
Explanation: Secure BFX is calling the system file open routine.
User Response: None

BFX923I Returned From File Open
Severity: 4 (Informational)
Explanation: The system has returned control to Secure BFX after opening the file
User Response: None

BFX930Itransfered nnn KB in ttt msecs = nnn KB/sec
Severity: 6 (Informational)
Explanation: BFX is reporting the number of Kilobytes transferred and the throughput rate.
Response: No response is required.

BFX940S Sperry and Unpack verbs are mutually exclusive
Severity: 12 (Severe)
Explanation: These verbs cannot be used together.
User Response: Remove one of the verbs.

BFX941S The Unpack verb requires a format-A output tape.
Severity: 12 (Severe)
Explanation: When converting 9 bit to 8 bit, the tape must be assigned in quarter word mode
User Response: Correct the tape assignment.

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.

BFX943F File= is NOT assigned.
Severity: 12 (Severe)
Explanation: The file must be assigned.
User Response: Add an assign for the proper file.

BFX944I DECODE EXIT ID=%s tstat=%o
Severity: 15 (Information)
Explanation: A Unisys UCSTLBL macro was issued. ID= request, tstat= is the octal return code.
User Response: None.

BFX945I %s
Severity: 15 (Informational)
Explanation: This is an English explanation of the return code in BFX944I
User Response: None

Appendix C. SSL TRACING

In the event of SSL connection problems, please gather the following information:

JOBLOG (If you are connecting to a remote system)

- Run the job specifying DEBUG BFX parameter.

COMAPI PRINT FILE

- Before the job is submitted, enter using your comapi keyin:
    - <*keyin*> log high (Turns on logging)
    - <*keyin*> log close (Starts a new print cycle)
    - Run the job
    - <*keyin*> log close (Close the print cycle This is the file with the data)
    - <*keyin*> log off (Turns off logging)

CPCOM TRACE FILE

- Before the job is submitted, enter using your cpcomm keyin:
    - <*keyin*> trace api-ssl,medium (Trace these records)
    - <*keyin*> trace api-tcp,medium (Trace these records)
    - <*keyin*> trace network,medium (Trace these records)
    - <*keyin*> trace ssl,medium (Trace these records)
    - <*keyin*> trace ip,medium (Trace these records)
    - <*keyin*> trace tcp,medium (Trace these records)
    - <*keyin*> trace close (Starts a new trace cycle)
    - Run the job
    - <*keyin*> trace close (Close the trace cycle This is the file with the data)
    - <*keyin*> trace off (Turns off tracing)
- Print the trace file.
    - Execute SYSLIB*CPCOMM.LTA (To print the trace)
    - The trace file name was displayed at close time
        - I (analyze interactively)
        - !HEX (print data in HEX)
        - !STATUS (Do a STATUS)

- !ALL (Print all trace Records)
- !QUIT (End)
- Send in the printed trace file. The name is displayed