H301/H301L Bulk File Transfer (BFX™) Utility

for Unisys OS2200 Systems

Release 1.4.6

Software Reference Manual

MAN-REF-H301-1.4.6-01

Revision Record

Revision*	Description
A (12/82)	Manual released. Corresponds to BFX version 1.0
B (04/85)	Manual updated to include sample applications and revisions of the text.
01 (06/87)	Manual updated at release 1.1. Publication number has been changed from 4399H301 to 4060350. Revised “Introduction” and “Control Statements and Parameters” sections. Added new error messages to Appendix B.
02 (11/90)	Manual updated to correspond to release 1.2. Added references to H307. Added NONSDF, ARCHIVE, and RESTORE control parameters. Made other minor technical corrections.
03 (08/12)	Manual updated to correspond to release 1.4. Added parameters KEYVAL, HOSTCHECK, and HOBCHECK.
1.4.0 (2/2013)	Manual reformatted to NESi documentation standards. Publication number changed from 460350 to MAN-REF-H301. Miscellaneous content updates. Manual updated to correspond to release 1.4. Addition of secure job transfer.
1.4.1 (7/2013)	Miscellaneous fixes to the document. Additional disconnect messages.
1.4.2 (9/2013)	Add support for isolating the Netex traffic (including secure job transfer) to specific IP addresses by using NTX prefixes in the hostnames and in the resolver records.
1.4.3 (10/2013)	Added information regarding the “enque_name” parameter for secure jobfile transfers.
1.4.4 (6/2014)	Miscellaneous fixes corresponding to 1.4.4 MTU updates.
1.4.4-01 (6/2014)	Added Appendix to cover diagnosing connection problems.
1.4.6-01 (8/2015)	Notes for ignored parameters during secure jobfile transfers.

Preface

This manual describes the user interface to the Network Executive Software (“NESi”) H301 Bulk File Transfer (BFX™) utility for Unisys OS2200 systems. BFX is used in conjunction with NESi’s NETwork EXecutive (NETEX®) family of software products for use on IP networks.

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

“ECL and Control Parameters for BFX” on page 15 presents user control statements and parameters. This section also shows the ECL used when running BFX, including simple examples.

The third section, “Applications” on page 29, provides detailed examples using BFX.

The next section, “Installing BFX” on page 33, describes the installation of BFX. It includes defining default values for control parameters described in the second section.

“Appendix A: User Modules” on page 45 describes how to write BFX user modules. The user modules are called from user exits within BFX. These user exits were carefully placed within BFX to allow the user modules to perform code conversion when not provided by NESi hardware or software (NESi provides for many types of code conversion), to transfer non-sequential files, or to perform other special processing by outside modules.

“Appendix B. BFX Internal Summary” on page 55 describes the BFX internal structure as it relates to writing user block (or record) modules.

“Appendix C. BFX Error Messages” on page 69 contains BFX error messages.

BFX uses NETEX, but the reader does not need to understand NETEX to use the first four sections of this manual and BFX. However, users creating their own user modules (as described in “Appendix A: User Modules” on page 45) are assumed to be familiar with NETEX.

Reference Material

The following manuals contain related information.

Number	Title and Description
Current Release	*H300IPC NetEx/IP® for Unisys OS2200 Systems*

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™
Unisys Corp.	OS2200

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

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

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

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

code conversion: An optional feature in NetEx that dynamically converts the host data from one character set to another. NetEx 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).

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

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

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

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

ISO: Acronym for International Standards Organization.

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

NETwork 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 ..... **iv**

Notice to the Reader ..... **v**
Notice to the Customer v
Document Conventions vi
Glossary vii

Contents ..... **ix**
Figures xi
Tables xii

Introduction **1**
Supported Configurations 1
Sample Network Configuration 1
BFX Component Programs 2

Using BFX **5**
Manual Job Submission 5
Automatic Job Submission 7
Remote Job Submission 9
Remote Job Submission Example 9
Data Modes 9
Security 10
File Size 10
Using SECURE BFX 10
ECL For Executing the BFXSJS Program 12
ECL For Executing the BFXTI Program 13
SAMPLE OUTPUTS 13

ECL and Control Parameters for BFX **15**
ECL For Executing the BFXTI and BFXTR Programs 15
BFX Execution Parameter Syntax and Placement 15
BFX Execution Parameters 16
Control Statements 18
Control Statement Parameters 18
BFX Example 25
Special Considerations 26
Record Formats and Record Type Conversion 26

Applications **29**
Unisys OS2200 to HP NonStop 30
Unisys OS2200 to Linux 31

Unisys OS2200 to IBM 32

Installing BFX 33
Prerequisites for Installation 33
Release Distribution 33
Installation Process 34
Step 1: Obtain the BFX distribution file 34
Step 2. FTP the distribution file to OS2200. 34
Step 3. Check for required updates 35
Step 4. Customize BFX 35
Step 5: Remap Common and DBanks (if required) 35
Step 6: Execute the BFXJS Program 36
Step 7: Configure CPCOMM for SSL SECURITY 37
Step 8: Configure BFXSJS 38
Step 9: Verify the BFX Installation (BFXJS needed) 38
Step 10: (Optional) Customize BFX 38

Appendix A: User Modules 45
Writing Record Modules 45
FORTRAN Entry 46
Assembler Record Module Entry 48
Block and Record Module Exit and Entry Summary 48
Writing Block Modules 50
FORTRAN Entry 51
Assembler Block Module Entry 53
Writing Job Submission Modules 53

Appendix B. BFX Internal Summary 55
Block Diagram 58
BFX Module Descriptions 60
BFX Receive File Control (RCV) 60
BFX Send File Control (SND) 63
Receiving Block Module (RBM) 66
Sending Block Module (SBM) 66
Receiving Record Module (RRM) 67
Sending Record Module (SRM) 67

Appendix C. BFX Error Messages 69

Appendix D. SSL TRACING 83

Index 85

Figures

Figure 1. Sample 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....9
Figure 6. BFXTI / BFXTR example ECL....15
Figure 7. BFX Control Statements....17
Figure 8. Typical BFXTI Run Stream....26
Figure 9. Example Unisys initiated BFX file transfer from Unisys to HP NonStop....30
Figure 10. Example Unisys initiated BFX file transfer from Linux to Unisys....31
Figure 11. Unisys initiated BFX file transfer from Unisys to OS2200....32
Figure 12. Sample BFXJS ECL....36
Figure 13. BFXSET Parameters....39
Figure 14. FORTRAN Record Module Entry Parameters....46
Figure 15. Block module entry....51
Figure 16. BFXTI Module Block Diagram....58
Figure 17. BFXTR Block Diagram....59
Figure 18. BFXJS Module Block Diagram....60
Figure 19. File Receive Data Flow....62
Figure 20. Send Receive Initialization Flow....64
Figure 21. File Send Data Flow....65

Tables

Table 1. Sending Record Module Entry Indications 48
Table 2. Sending Record Module Return Actions 49
Table 3. Receiving Record Module Entry Indications 50
Table 4. Receiving Record Module Return Actions 50
Table 5. BFX Default modules 55

Introduction

The Bulk File Transfer (BFX™) utility is a software package to be used with NetEx Software's NETwork EXecutive (NETEX) software. BFX and NETEX 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 NETEX products installed (H301 BFX requires H300IPC).

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, BFX can be run as a batch job, from a terminal, or as a started job.

BFX can be used to transfer files between different hosts which may require specialized record or data conversion. BFX allows the use of NETEX code conversion. For other code conversion, BFX has a set of user exits that allow user modules to process data both before and after transfer over the network. These modules may take responsibility for accessing data. This makes it possible to copy and convert complex data between different systems, or copy direct access files or data bases. The requirements for creating the user modules are described in "Appendix A: User Modules" on page 45.

Supported Configurations

BFX uses the NETEX communications subsystem and IP networks for its data communications. It is designed to use all the capabilities of these products to move files at multimegabit speeds over local (LAN) or WAN IP-based networks.

Sample Network Configuration

BFX requires that cooperating batch application programs reside in both the source and destination CPUs, as shown in Figure 1 on page 2.

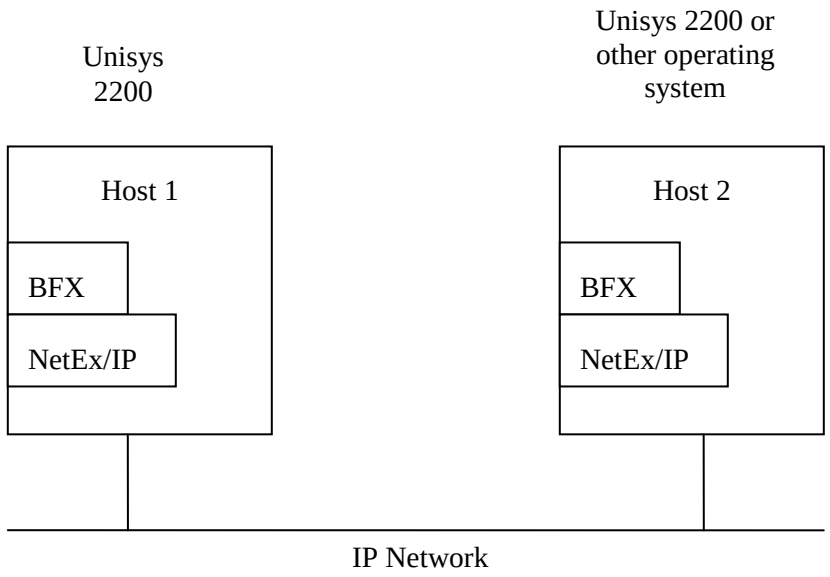

Figure 1. **Sample Netex/BFX network**

Figure 1 is a simple example and demonstrates the following points:

- Both BFX and NETEX must reside in both hosts. If the two hosts use different operating systems and/or are of a different manufacturer, data transfer is still possible provided the proper versions of BFX and NETEX are installed and active. BFX makes use of NETEX code conversion, and allows for user modules to perform more elaborate data conversion.
- Each application program invokes the BFX utility, rather than program NETEX directly. Therefore, the user does not need to learn how to program NETEX.
- BFX is an application program that can be used like any other file processing utility. It can be directly invoked by BFX users without effecting or being affected by other BFX users and NETEX applications that may be using NETEX at the same time.

BFX Component Programs

BFX is a system that consists of five separate programs: BFX Secure Transfer Initiator (BFXTI), BFX Transfer Responder (BFXTR), BFX Job Submitter (BFXJS), BFX Secure Job Submitter (BFXSJS) and the basic mode Transfer Initiator (BFXTIB).

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 or BFXSJS program. BFXTI will call BFXTIB to do the actual data transfer.

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.

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

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

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 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 BFX " on page 15.) 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 RESPOND commands and does not find any other BFX commands. BFXTR ends processing. BFXTI scans the INITIATE 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 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
@PASSWD   SECURI
@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.

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.

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
@PASSWD   SECURI
@. 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 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.

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, encryption, 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.

Security

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

File Size

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.

Using SECURE BFX

Secure BFX allows users to transmit the job file across the IP network in an encrypted form. This prevents user-ids and passwords 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 are transmitted in an encrypted form and which job files will be transferred as in past releases.

The user can set installation defaults for the following secure transfer parameters. These are located in the program sdefault/c in the release file. You can use the ECL in member sdefault-c to assemble and link the defaults. REMAPH301 must be rerun after assembling new defaults. You must change the dbanks to point at the dbanks you wish to map. All the parameters can be overridden by the configfile with the exception of the five SECHOST names, the enqueue_name, and the use name. These MUST be set in sdefault.c.

The configuration file is optional and need only include the parameters to be overridden. The element name is SJSCONFIG. It is pointed to by a @USE CONFIGFILE.,YOURFILE*NAME.

The parameters are:

KEYWORD	**Values**	**Meaning**
SECURE =	YES / NO	Yes - The job file will be securely transmitted. NO – The job file will be transmitted as it has been in the past The default is NO
QUALIFIER =	nnnnnnnn	nnnnnnn is the Qualifier of any files created by secure transfer. Currently only the five job files use this parameter. The default is NTXTMP

KEYWORD	Values	Meaning
JOBFILE =	nnnnnnnn	nnnnnnn is the file name used by BFXSJS. The number 1 through 5 will be appended to this. This represents the 5 maximum concurrent transfers that can be running. Additional requests will be queued. The default is JOBFILE
FILESIZE =	nnn	nnn represents the file allocation size for the job file. The default is 3000
RTLEVEL =	nn	nn represent the real time level for this task. The default is 30.
KEYIN =	nnnnnnn	nnnnnn is the character string used by the operator to communicate with the BFXSJS job submission task. The default is SJS
DEFAULTHOST =	nnnnnn	nnnnnn is a netex host name that will receive the job file if the TO = parameter is not coded in the BFX. (THIS MUST MATCH THE DEFAULT HOST SPECIFED BY "TO= " PARAMETER IN BFXUSER IF YOU WISH TO USE THIS FEATURE.) The default is NONE.
COMAPI =	n	n specifies the mode of the COMAPI application. The COMAPI parameter may point to a CPCOMM running in a mode other an "A". Modes A,B,C,and D are supported The default is A.
READTIMEOUT =	nn	Specifies the number of seconds before a transfer will abort because no data has been received. The default is 60 seconds
CONDELAY =	n	Specifies the number of seconds to wait between connection attempts. The default is 5 seconds. (Note: This parameter is ignored in the jobfile for secure jobfile transfers.)
CONRETRY =	n	The number of times to retry the IP connection. (Note: This parameter is ignored in the jobfile for secure jobfile transfers.)
LCLDMODE =	n	n represents the character set of this host. A value of 2 is ASCII. A value of 3 is EBCDIC. The default is 2.
DEBUG =	YES / NO	NO prevents diagnostic messages from being written to the print file. YES allows the diagnostic messages to be printed The default is NO

KEYWORD	Values	Meaning
SJSPORT =	nnnn	nnnn represent the port number BFXSJS will listen for a connection on. This value should only be altered on a request from Technical Support The default is 6950.
LOCALPORT =	nnnn	nnnn represent the port number BFXTI will use to connect with BFXSJS. 0 allows the system to assign a port. This value should only be altered on a request from Technical Support.
SEC_HOST[n]	nnnnnnnn	***SDEFAULT.C ONLY***** User may specify up to five netex host names that will default to secure=yes for the job transfer.
USE_NAME =	BFXSJSPRT	***SDEFAULT.C ONLY***** This is the use name associated with the breakpointed print file.
ENQUEUE_NAME	BFX*	***SDEFAULT.C ONLY***** Unisys TCP_CONNECT process encounters a resources unavailable error if an SSL TCP_CONNECT is issued when a second SSL TCP_CONNECT is in process to the same host. Netex will thread the connect process to a single host, by queuing on this HLQ* the name of the remote system. This insures no job keeps seeing a resource unavailable error.

In addition to these configuration parameters, the sdefault/c program may specify up to a maximum of five NetEx host names that will change the default SECURE = parameter to YES. These are in the sdefault/c program as SECHOST1, SECHOST2, …, SECHOST5. The sdefault/c also contains the line "Customer modifiable string to identify defaults in run". Customers may modify this string to specify the date or identify the values that are now assembled in. This line will be outputted in every BFXTI print file.

ECL For Executing the BFXSJS Program

Sample ECL for executing the Secure Job Submission server is located in the release file. The element name is BFXSJS-SAMP. The BFXSJS program must be running on the host that receives the secure job file. BFXJS, the original server must also be running if you wish to receive jobs that are NOT encrypted. By starting either one or both of these jobs, you can control receiving only encrypted or unencrypted job files.

Users may wish to pre-allocate the file BFXSJS$LOCK. This file only allows one copy of BFXSJS to execute at a time. This program does multi-task and will support up to five concurrent secured transfers. Additional requests will be queued.

Users may wish to pre-allocate the five job files and set the cycle limit to five or ten. Using the defaults, these files would be name NTXTMP*JOBFILE1, NTXTMP*JOBFILE2, … NTXTMP*JOBFILE5.

The BFXSJS (Secured job transfers) programs respond to the following commands: (SJS is the operator KEYIN)

SJS SWITCH	Close the current cycle of the print file, so users may access it. A new cycle is created. The USE_NAME used is defined in the sdefault.c module, and defaults

	to BFXSJSPRT. The file name used in the switch is the next cycle of the USE_NAME.
SJS DEBUG OFF	disables debug tracing on the job submission server
SJS DEBUG ON	enables debug tracing on the job submission server
SJS TERM	terminates the server – same as ABORT
SJS ABORT	terminates the server – same as TERM

ECL For Executing the BFXTI Program

Secure transfer was designed to run with your existing ECL run-streams. It will use the assembled in defaults from sdefault/c. If SECURE = YES was specified, the job file will be securely transmitted. If SECURE = NO was specified, the job file would be transmitted as it was in the past. The only exception would be if the TO = parameter of the transfer matched one of the five hosts specified by the SECHOST parameters. The file would be sent using secure transfer.

To override the defaults, you may add a @USE statement before the @XQT BFX.BFXTI statement. It would look like:

```
@USE CONFIGFILE.,YOUR*FILE
```

where YOUR*FILE points to a file containing the element SJSCONFIG. This must be a quarter word file containing any SECURE TRANSFER parameters you wish to override. The parameters should be entered as the parameter = value. Each parameter must be on a separate line. Comments are not permitted.

SAMPLE OUTPUTS

When BFXSJS receives a file, BFX017I START OF FILE TRANSFER is issued. This message is continued with the name of the system that is sending the file, and the JOB CARD being submitted. This allows users to easily track which system submitted a particular job.

```
                                BFXSJS
TIMESTAMP:Wed Jan 23 14:26:39 2013
TASK:1 BFX017I START OF FILE TRANSFER
TASK:1 BFX017IC RECEIVING FROM: unid4150.netexsw.com
TASK:1 BFXRMT BFX017I START OF FILE TRANSFER:1
TASK:1 BFX017IC JOB:@RUN,/A   DABR1,0/SECRET,NETEX
TASK:1 BFXRMT BFX101I FILE DON1 TRANSFERED, 3 RECORDS SENT
TASK:1 BFX201I FILE NTXTMP*JOBFILE1(+0). TRANSFERED 3 RECORDS RECEIVED
TASK:1 BFX047I JOB SUBMITTED
TIMESTAMP:Wed Jan 23 14:26:40 2013
```

When BFXTI executes, the parameters in effect are logged. The line ”Customer modifiable string to identify defaults in run” is outputted from the sdefault/c module. This line may be modified by the customer to identify the date or options used. This can insure the defaults you believe should be in effect are actually in effect.

```
                                BFXTI
Copyright NETWORK EXECUTIVE SOFTWARE 2013
Customer modifiable string to identify defaults in run
CONFIGURATION VALUES IN EFFECT
```

```
LOCALPORT = 0
FILESIZE = 1000
COMAPI = A
QUALIFIER = NTXTMP
SJSPORT = 6950
DEFAULTHOST = NONE
KEYIN = SJS
LCLDMODE = 002
RTLEVEL = 30
READTIMEOUT = 60
JOBFILE = JOBFILE
SECHOST1 = BUILDSRV
SECHOST2 = AIX2
SECHOST3 = UNID4150
SECHOST4 =
SECHOST5 =
DEBUG = NO
SECURE = YES
TIMESTAMP:Wed Jan 23 14:26:39 2013
BFX015I PROCESSING HAS BEGUN
RMTMSG: BFX221I RECEIVING FILE NTXTMP*JOBFILE1(+1).
BFX312I BLOCKSIZE N/A DATAMODE 0202 LCM 1
BFX220I SENDING SECURED FILE DON1 to UNID4150
BFX220IC JOB:@RUN,/A   DABR1,0/SECRET,NETEX
BFX101I FILE:DON1 TRANSFERRED  3 RECORDS SENT
RMTMSG: BFX201I FILE NTXTMP*JOBFILE1(+0). TRANSFERED; 3 RECORDS
RECEIVED
RMTMSG: BFX047I JOB SUBMITTED
TIMESTAMP:Wed Jan 23 14:26:40 2013
```

ECL and Control Parameters for 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 6. Complete examples of BFXTI and BFXTR applications are contained in the "Applications" section on page 29.

```
@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 6. BFXTI / BFXTR example ECL

The following statements appear in Figure 6:

@USE

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

@ASG,A

Assign the file to this job.

@XQT NETEX*BFX.BFXTI

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

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.

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 7 on page 17.

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.

```
|SEND     |   | TO=destination host name,   |
|RECEIVE  |   | FROM=originating host name, |
|JOBSUBMIT|
                 ID=bfx identifier name
                [ARCHIVE=store bit mode records in SDF files]
                [BLOCK=size of data block to send over network]
                [BMOD=module to send/receive blocks]
                [BPARM=string for block module]
                [|CLOSE  |]
                 |NOCLOSE|
                [CONNDELAY=seconds to delay between connection attempts]
                [CONNMAX=times to try the connect]
                [FILE=name of data file to send/receive]
                [HOBCHECK=|ABORT    |
                          |CHECKONLY|
                          |CLEAR    |
                          |OFF      |
                [HOSTCHK   |]
                  NOHOSTCHK|
                [JID=offered name of bfxjs job]
                [JOBFILE=name of remote job input file to send]
                [KEYVAL=identifies the operator key-in to communicate with BFXJS]
                [MODE=|ASCII  |]
                      |BIT    |
                      |EBCIDIC|
                      |FDATA  |
                      |OCTET  |
                [MSGLVL=severity of messages to be logged]
                [NEWHOST=new name of host]
                [NONSDF=access character format file as binary]
                [NOSUBMIT]
                [RATE=value to reduce the BFX transfer rate]
                [RESTORE=retrieve ARCHIVEd records]
                [RMAXL=maximum record length]
                [RMOD=module to read/write file records]
                [RPARM=string for Record Module]
                [|SOE  |]
                 |NOSOE|
                [SPERRY]   I/O and file handling between two 2200 systems]
                   [RPARM=TRACKIO]
                   [RPARM=LABELS]
                   [RPARM=LABELS,FILES=nnn|*]
                   [CHECKSUM]
                [TIMEOF=offer time out]
                [TIMEOU=read time out]
                [|TIMESTAMP  |]
                 |NOTIMESTAMP|
                |.EOT statement
                |.END statement
```

Figure 7. BFX Control Statements

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 host that the file will be transferred to. This name is configured by the systems programmer when the NETEX 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 host that will send the file. This name is configured by the systems programmer when the NETEX 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 host that the JOB will be transferred to. This name is configured by the systems programmer when the NETEX 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 7 on page 17). Defaults for many of these parameters are defined in the BFXSET program. Refer to “Installing BFX” on page 33 for more information.

ID

This required parameter (not required with the JOBSUBMIT statement) specifies a unique identifier for the transfer. This value may be any uppercase alphanumeric string 1 to 8 characters in length. Only one BFX program with this ID should be present in the initiating and responding CPUs. Users are strongly encouraged to use the jobname of the “remote” job as the ID in both the BFXTI and BFXTR programs. The ID parameters for the BFXTI and BFXTR parameters must agree for file transfer to take place.

This parameter is required on the first SEND/RECEIVE card in both the procedures using the BFXTI and BFXTR programs.

ARCHIVE

This optional parameter allows records to be received from the remote host in bit mode, but stored on the 2200 in SDF files.

The ARCHIVE option is useful for sending files in bit mode to the 2200 for storage, *with the intention of restoring them back to the originating host* (see the description of the RESTORE parameter).

Since SDF files have an internal end of file, the exact length of the original file and bit-length of each record are retained. However, the file on the 2200 is not an exact copy of the file on the originating host. The data are the same, but control images are added. SDF file format contains the length of the record in words, and ARCHIVE places into the low order portion of the SDF im-

age control word the number of unused bits in the last word of the record. This allows the RESTORE command to return back to the originating host the exact number of bits that were originally sent in the record. Records received from a remote host using ARCHIVE should not exceed 8100 characters (2047 words).

The control images are inconsequential if the file is simply being stored on the 2200. However, if the data are used for further processing (before RESTORE), the control images may cause a problem. In that case, regular MODE= BIT or MODE= OCTET can be used.

The output from ARCHIVE may be a data file or a symbolic program file element. The program file table of contents can be used to maintain a 'directory' to archived data sets.

BLOCK

This optional parameter specifies the size in words of the buffers of data to be sent through NETEX to the other host. The size in BLOCK must be at least large enough to accommodate the largest logical record 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 largest BLOCK allowed is 20,000 words.

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

BMOD

This optional parameter specifies the name of the user module (the block module) that provides or accepts buffers of information sent through the IP network via NetEx/IP. In order to be accessed, the block module must have been previously link edited into the BFX program.

Certain user-written block modules may not use record modules to read file information. In that case, the RMODULE and RPARM parameters will be ignored.

If this parameter is not supplied, an installation-defined default is used. The NESi default block module is designed to call a record module to obtain or store file information, and to provide or decode the protocol needed to block and de-block records flowing over the IP network.

See “Appendix A: User Modules” on page 45 for more information on user-provided block modules.

BPARM

This optional parameter specifies a string of parameter information that will be passed to the specified block module for processing. User-written block modules may examine this string to control special processing options. The default NESi-provided block module ignores the BPARM string.

The string parameter may be specified as a single keyword, as a list of items enclosed in parenthesis, or as a string delimited by apostrophes.

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

CLOSE and NOCLOSE

This optional parameter specifies whether SCLOSE should be used to terminate the file transfer or not (instead of SDISC). If SCLOSE is used, any other BFX components to be communicated with (BFXTI, BFXTR, and BFXJS) must also use SCLOSE. It is recommended the CLOSE parameter be used, if all BFXs support it.

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

CONNDELAY

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

The default is 10 seconds.

CONNMAX

This optional parameter specifies the number of times to try connect when a connect attempt gets a 3501 or 3502 error.

FILE

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.

If this parameter is not supplied, an installation defined default is used (defined by the FILE parameter in the BFXSET subroutine - see "Installing BFX" on page 33).

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.

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 H301 product. The default for the H301L product is HOSTCHECK.

This option should not be specified in any of the following conditions:

- If dissimilar NCTs are used on each of the connecting hosts
- If group host names are used
- If a local loopback host name is used (NTXLCLxx)

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.

If this parameter is not supplied, an installation defined default is used (defined by the BFXSET subroutine. See "Installing BFX" on page 33 for more details).

KEYVAL

This BFXJS parameter (unsecured job transfers) specifies an eight character key-in identifier the operator may use to communicate with BFXJS from the system console. For H301, a KEYVAL of "******" will not start the operator interface. H301 only supports a KILL command. H301L only supports the TERM command.,

MODE

This optional parameter specifies the nature of the information in the file. The ASCII and FDATA parameters imply that the entire contents of the file consist of human-readable ASCII or FIELDATA 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 or MODE=FDATA is specified, the ASCII or FIELDATA 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 MODE=OCTET is specified, binary data are received or sent in multiples of 8 bits, but do not need to be 28-word multiples.

If MODE=EBCDIC, the NETEX EBCDIC mode (3) will be used. This is for applications that want to "store and forward" the EBCDIC date 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.)

If the MODE parameter is not supplied, an installation defined default is used (defined by the MODE parameter in the BFXSET subroutine- see "Installation" on page 25).

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 0-15. 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, NETEX, or newly written user modules.

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.

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

NEWHOST

This optional parameter specifies a new host name of the NETEX 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 of character format, but is to be accessed as a binary file. This option supports files which consist of 9-bit data, but are stored as a continuous stream of bytes, not in SDF format. The NONSDF feature reads and writes data using direct IOW$ calls.

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.

Even though the data is accessed as binary, it is treated in the BFX record as character mode, subject to code conversion. Though this option, the user may access a specific disk file format or ANSI tapes containing ASCII character data.

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.

RATE

This optional parameter (expressed in a range of 1 – 24,000 kilobits/second) can be used to slow down the BFX transfer. NETEX will attempt to reduce the rate to this value by adding timed waits between transmissions. RATE is used primarily to reduce BFX impact on a loaded system

RESTORE

This optional parameter causes the data originally received by means of the ARCHIVE parameter to be retrieved from the SDF file specified by the FILE directive. This SDF file will be sent via bit mode to the destination host; using the RESTORE implies MODE=BIT.

NOTE: Use RESTORE only for files which were previously ARCHIVED.

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. RMAXL must not exceed 2499 words for ASCII mode or 1666 words for FDATA mode. 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.

If this parameter is not supplied, an installation defined default is used (defined by the BLOCK parameter in the BFXSET subroutine - see “Installing BFX” on page 33).

RMOD

This optional parameter specifies the name of the subroutine (the Record Module) that will provide or accept the logical records of the file. In order to be accessed, this Record Module must have been previously compiled or mapped into the BFX program.

If this parameter is not supplied, an installation defined default is used (defined by the RMOD parameter in the BFXSET subroutine - see "Installation" on page 25). The default Record Module (called STANDARD) transfers character files between all types of host processors, and moves binary or structured information to another host without change on a logical record basis.

See “Appendix A: User Modules” on page 45 for more information on user provided Record Modules.

RPARM

This optional parameter specifies a string of parameter information that will be passed to the specified Record Module for processing. User-written Record Modules may examine this string to control special processing options. The default Record Module provided ignores the RPARM string.

The string parameter is specified as a single keyword.

If no parameter is supplied, an installation defined default is used (defined by the RPARM parameter in the BFXSET subroutine see "Installation" on page 25). RPARM=RECSIZE=xxxx is used to tell the MODE=OCTET feature how long each record is, allowing it to partition the file into records of that size. (When using this RPARM statement, do not include spaces in the RECSIZE = xxxx portion.)

SOE and 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.

SOE is the default setting.

SPERRY

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

RPARM=TRACKIO

This statement is used only if the transfer is between two 2200 systems and is a disk-to-disk transfer. The SPERRY and RPARM=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 optional checksum). 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.

RPARM=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, H301 adds additional "non-data" records to the transfer. The non-data records inform the receiving BFX (H301 also with SPERRY and RPARM= LABELS) how to write the destination tape. Without the SPERRY and RPARM = 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 RPARM = LABELS options. Since operator mounting of tapes can be time-consuming, the BFX parameter TIMEOU should be set large enough to allow for tapes to be mounted. The NETEX parameter READTO should also 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. If H307 is being used, see the DATATIME parameter.

RPARM=LABELS,FILES=nnn|*

This statement adds the FILES=nnn|* clause to the RPARM =LABELS (see preceding discussion of RPARM= 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 H301 BFX writes the required EOF records on the tape to separate the files.

CHECKSUM

This option is used only with the RPARM = TRACKIO statement. It causes a one-word checksum value to be included with each track from the sending UNISYS host. The receiving UNISYS BFX recalculates the checksum and compares it to the value sent with the track record. An unsuccessful comparison causes the transfer to abort. To use this feature, the following input must be included for both the sending and receiving BFX.

```
SPERRY
RPARM=TRACKIO
CHECKSUM
```

Other considerations for 2200 transfers:

- RMAXL must be at least 40 if LABELS is specified.
- RMAXL must be at least 2000 if TRACKIO is specified (and BLOCK must be at least 9/8 of RMAXL).
- MODE = BIT must be at least 2000 if TRACKIO is specified (and BLOCK must be at least 9/8 of RMAXL).
- MODE= BIT.

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 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.

TIMEOU

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

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.

TIMESTAMP and **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.

.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 8 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 8. Typical BFXTI Run Stream

Special Considerations

Record Formats and Record Type Conversion

The supplied Record Module uses the standard SDF library routines (SDFI$/SDFO$) to read and write ASCII or FDATA 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 togeth-

er, 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. If variable length bit mode records need to be stored on the UNISYS 2200, see the ARCHIVE feature of BFX for a possible solution.

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 HP NonStop
- Unisys OS2200 to Linux
- Unisys OS2200 to IBM OS2200

Unisys OS2200 to HP NonStop

The example in Figure 9 shows the Unisys OS2200 job used to perform a BFX transfer from a Unisys initiating host to an HP NonStop remote host. The Netex name of the Unisys initiating host is CHICA, and the Netex name of the HP NonStop 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         .Create a temporary job file
:TRINFILE $AUDIT.BFXJS.TRINFILE
FILE $DSMSCM.BFXSAV.LOTESTX1
ID I101
FROM CHICA
MODE BIT
BLOCK 30000
MSGLEVEL 0
RECEIVE
:JOB $AUDIT.BFXJS.TESTJOB1
SPOOLCOM JOB (LOC #JS), DELETE !
RUN $SYSTEM.BFX.TR/NAME $TR,IN $AUDIT.BFXJS.TRINFILE,MEM 64/
@END
@.
@USE FILE2SEND,FILE*SENDFILE.
@ASG,A    FILE2SEND.
@XQT file.BFXTI
SEND
MODE=BIT
JOBFILE=UNI2JOB
ID=I101
TO=WASHC
FILE=FILE2SEND
BLOCK=3000
RMAXL=22
.END of BFX JOB
@FIN
```

Figure 9. Example Unisys initiated BFX file transfer from Unisys to HP NonStop

Unisys OS2200 to Linux

The example in Figure 10 shows the Unisys OS2200 job used to perform a BFX transfer from an OS2200 initiating host to a Linux remote host. The Netex name of the OS2200 initiating host is CHICA, and the Netex name 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
/opt/bfx/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.
@XQT file.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 BFX transfer from Unisys initiating host to an IBM remote host. The NetEx name of the Unisys initiating host is CHICA, and the NetEx 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.PFXT.PXTLOAD,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
@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 BFX

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

Prerequisites for Installation

H301 BFX needs the following prerequisites for successful operation:

- A Unisys OS2200 system that is running the H300IPC NetEx/IP software product
- At least one other host with its own NetEx/IP and BFX software.
- Before installing H301 BFX fox OS2200, the H300IPC NetEx/IP installation should first be completed.

Release Distribution

H301 BFX for OS2200 is distributed as a downloadable file. See Step 1 below.

Installation Process

This section describes the installation procedure for the H301 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.

Step 1: Obtain the BFX distribution file.

Step 2. FTP the distribution file to OS2200.

Step 3. Check for required updates.

Step 4. Customize BFX

Step 5: Remap Common and DBanks (if required).

Step 6: Execute the BFXJS Program

Step 7: Configure CPCOMM for SSL SECURITY

Step 8: Configure BFXSJS

Step 9: Verify the BFX Installation (BFXJS needed)

Step 10: Customize BFX (optional)

Step 1: Obtain the BFX distribution file.

The H301 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.

Step 2. 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 H301_RELEASE_nnn_CFMT H301*RELEASE-nnn.
5) Quit your FTP client

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

H301*RELEASE-nnn

Step 3. 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.

Step 4. Customize BFX

For SECURE TRANSFER:

Secure transfer allows the site to set default values to be used by all secure BFX jobfile transfers. These are set in the C program sdefault/c in the distribution file. These parameters can be updated any time after installation. You will need to complete this step and step 5 for the new values to take effect. See the section "Using Secure BFX" for a listing of the parameters and values. You may use an editor of your choice to update this member. Please follow all C programming rules and insure the program compiles and links successfully. You may use member sdefault-c to compile and link this program into the distribution library.

```
@ed,uq  H301*RELEASE-14x.sdefault/c
Make required changes like:
1:
c /Customer modifiable string to identify defaults in run/MY NEW PARMS/
l SECHOST0
c /""/"NETHOST"/
exit
@ed,uq  H301*RELEASE-14x.sdefault-c
l xxxxxxx
c /xxxxxxxxxxxxxx/H301*RELEASE-14x./
exit
@ADD H301*RELEASE-14x.sdefault-c
```

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

For Data File Transfers (optional):

See Step 9 Customize BFX if you wish to modify the defaults for data file transfers or incorporate user written routines. If you don't have user-written modules and wish to accept the default parameters, you may skip this step. New users may wish do initial validation of the BFX software before customizing it. If you wish to change the parameters at a later date, you must remap the common and dbanks for the changes to take effect.

Step 5: Remap Common and DBanks (if required)

The following NETEX Common banks were used to build BFX:

- NTX$COMN 0401101

- NTX$DB0 0401102
- Fortran Common BANKS starting with 0400027 as specified by UNISYS

If these are the BDIs used by your installation for NETEX, and you have not modified any defaults, this step is not required. If you wish to build additional data banks (NTX$DB1, NTX$DB2 …) or these were not the BDIs used in your installation, you need to rebuild BFX for the correct BDIs. In the release program file you will find an element named remaph301.

- Modify the first several statements to conform to your file names.
- Change (0401101|0401603) to match the BDI of your NETEX common DBank (NTX$COMN).
- Change (0401102 | 0401604) to match the BDI of the NETEX data bank for this BFX install (NTX$DB0, NTX$DB1, NTX$DB2…)
- Change /1101-2 to the version desired for these BFX absolutes BFXJS, BFXTIB, BFXTR, BFXTI
- @ADD RELEASEFILE.REMAPH301 to create the new programs. BFXTI is now an extended mode program and does not need the common and dbanks linked in. The name of the BFXTI program determines which basic mode copy of BFXTIB to call. If you execute BFXTI/1106, when the extended mode program has completed, it will call BXTIB/1106 to do the actual data transfer. This allows users to continue to spread the BFX workload across multiple dbanks without modifying the existing ECL.

Step 6: Execute the BFXJS Program

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 36. This sample ECL is contained in the release file element: BFXJS-SAMP.

The run breakpoints the print file to NETEX*BFXJS. The use statement points to the current release library for BFX. BFXJS will remain in execution until it is cancelled or a JS KILL or TERM (KEYVAL = JS was used as a BFXJS parameter) command is issued from the system console. If NETEX terminates, BFXJS will remain in execution and print a message every 30 seconds until NETEX is started again.

```
@RUN,/A   BFXJS,0/SECURITY,NETEX,99,999
@SYM,DF   PRINT$
@USE      BFX,H301*RELEASE-nnn
@CAT,PG   BFXJS(+1),F///99999
@BRKPT    PRINT$/BFXJS
@COPY,A   BFX.BFXJS
@XQT      BFXJS
MSGLEVEL = 0
JOBFILE = JOBFILE
TIMESTAMP
KEYVAL = JS
.END
@PMD,L
```

Figure 12. Sample BFXJS ECL

The following statements appear in Figure 12:

- **MSGLEVLEL=0** produces the most amount of information in the BFXJS log. See the MSGLEVEL control statement for setting the value appropriate to your installation.
- **TIMESTAMP** specifies messages written by BFXJS will include the current time.
- **KEYVAL=JS** specifies the operator can issue commands to BFXJS (unsecured transfers) by using the JS as the console key-in.
- **.END** signals the end of the parameters.

Step 7: Configure CPCOMM for SSL SECURITY

(OMIT THIS STEP IF YOU DO NOT WISH TO USE SECURE TRANSFER)

Before secure transfers can be initiated, the OS2200 system will need to be configured.

If your certificates will NOT be signed by a third party, the self-signed certificates will need to be installed on the systems using secure transfer. For a system using CPCOMM, the certificate will need to be copied into the CPCOMM configuration file pointed to by the "TRUSTED_CERTIFICATE_FILE" parameter of the SSLTLS-security statement. For CPCOMM/OS, all certificates installed are trusted.

For CPCOMM the sample SSL/TLS statement could look like:

```
SSL/TLS-SECURITY,SECCPFTP ;
  RSA-PRIVATE-KEY-FILE,DON*KEY.TESTPR ;
  RSA-CERTIFICATE-FILE,DON*CERT.CERTSIGNED ;
  TRUSTED-CERTIFICATES-FILE,DON*CERT.CERTSIGNED ;
  CIPHER-SUITE-MINIMUM,RSA_WITH_RC4_128_MD5
```

For CPCOMM/OS the sample SSL/TLS statement could look like:

```
SSL/TLS-SECURITY,SECCPFTP ;
      RSA-PRIVATE-KEY-FILE,UNID4150key.pem ;
      RSA-CERTIFICATE-FILE,UNID4150cert.pem ;
.     These above two files reside in the SAIL partition
      CIPHER-SUITE-MINIMUM,RSA_WITH_NULL_MD5
```

Care should be used when specifying the "CIPHER-SUITE-MINIMUM". Setting it to high could prevent system from negotiating a connection. Setting it to low, allows for simpler algorithms to be used, allowing for easier cracking of the security. The OS2200 operating system will negotiate to the highest possible level of security with partner system.

Configure the COMAPI interface into CPCOMM;

The PROCESS statement for COMAPI **must include** the following three statements:

```
SSL/TLS-CLIENT-AUTH,WEAK ;
SSL/TLS-SECURITY,SECCPFTP ;          Points to the correct SSL/TLS statement
INPUT-QUEUE-THRESHOLDS,50,1000,1000000
```

IT MUST NOT INCLUDE:

```
SSL/TLS-SECURE-PORTS,nnnn
```

The COMAPI interface can support application imposed security or system imposed security. It cannot support both features in the same application. Multiple copies of COMAPI can be configured if both features are required. Secure Transfer allows a connection to the COMAPI interface running in a mode other than "A". (See the COMAPI = configuration parameter). Secure transfer MUST communicate with the same copy of CPCOMM as H300IPC. (See Configure DNS).

CONFIGURE DNS

Secure Transfer finds the IP address of the NETEX host, by issuing a call to DNS. This can be an external server or a file that is pointed to by the HOST-FILE parameter of the TCP/IP-DNR CPCOMM statement. For CPCOMM/OS, this file exists in the SAIL partition. You must use the SAIL CONTROL CENTER application to update it. **Each Netex host that will be using secure transfer must be added to the DNS sever. Users may either add NTX*netexhostname* or *netexhostname*. The NTX*netexhostname* allows the user to specify a subset of the IP addresses defined on the host.** Secure transfer will retrieve up to 10 IP address for each netex host, if they have been configured in DNS. No GNA addressing information is required for secure transfer. The DNS server will simply have the Netex host name and the IP addresses. These IP addresses will be owned by the copy of CPCOMM that is communicating with Netex. Comapi must also be able to use this IP address.

Step 8: Configure BFXSJS

(OMIT THIS STEP IF YOU DO NOT WISH TO USE SECURE TRANSFER)

Refer to the Section "Using SECURE BFX" earlier in the manual for implanting a secure job transfer environment.

Step 9: Verify the BFX Installation (BFXJS needed)

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

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

Follow the on screen prompts as directed.

Step 10: (Optional) Customize BFX

The installation systems programmer may specify installation defaults for parameters supplied to the BFXTI, BFXTR and BFXJS programs using the BFXSET subroutines. However, re-mapping (see step 5) is necessary after any changes.

BFXSET – Defaults for BFXTI, BFXTR and BFXJS

BFXSET is a FORTRAN subroutine contained in BFX.BFXUSER. This subroutine is divided into two parts. The first part is a block IF statement that will be executed once per BFX run. This block is used to establish the default value for the parameters that need to be set only once per transfer: If the user overrides these values, the new values will remain in effect for the duration of the BFX run.

The second part contains the parameters that are to be reset after each file transfer. These parameters will initially be set with the first pass parameters. After each file transfer the parameters will be reset to their default values.

The BFXUSER element should be edited if any default values are to be changed. Once these values are changed, the new source is to be saved and compiled with the other BFX modules. Sample ECL is supplied in BFX.BFXUSER-FOR to compile. Use the REMAPH301 to link the updates into BFX.

BFXSET Parameters

Figure 8 shows the BFXSET parameters that can be used to customize BFX default values to fit the needs of a particular installation.

```
[BLKMX=default block size]
[BLOCK=default blocking size]
[BMOD=default Block Module name]
[BPARM=default string to pass to Block Module]
[CHKSUM=default to pass a checksum value between 2200 systems doing trackio
[CLOSE=SCLOSE/SDISC termination flag]
[DTIME=delay time between SCONNect retries]
[DTIMES=retries for Busy Process]
[FILE=default assigned file name]
[HOSTCK = default for Host Check]
[JBLOCK=default job block size]
[JFILE=default assigned job file name]
[JFLAG=JOBSUBMIT flag]
[JID=default BFXJS ID name]
[JRMAXL=default job record length]
[MODE=file transfer mode]
[MSGLV=default message log level]
[RBM=default for remote byte machine (H301L only)
[RMAXL=default maximum record length]
[RMOD=default Record Module name]
[RPARM=default string to pass to Record Module]
[SEND=default Send/Receive mode]
[SENDJB=send a job flag]
[TIMEFL=BFX message time stamp flag]
[TIMEOF=default NETEX offer time out]
[TIMEOU=default NETEX read time out]
[TO=default remote host name
```

Figure 13. BFXSET Parameters

The meaning and defaults of these parameters are as follows:

BLKMX

This parameter specifies the default length of the buffer size (in words) that the user can specify via the BLOCK directive. The default is set at 18000 words for BFXTI and BFXTR, and 1024 words for BFXJS. For H301L the default is MIN0(BLOCK,8000).

BLOCK

This parameter specifies the default block size (in words) to be used to move file data over the IP network by NETEX. The default is BLKMX.

BMOD

This parameter specifies the alphabetic name of the default Block Module to be used for transferring the file. If not specified, the default is STANDARD, which is a symbolic name that calls the standard Block Modules.

BPARM

This parameter specifies a default string of character parameters to be passed to the invoked Block Module. This string, if provided, may be up to 64 characters in length. By default, a string of all blanks is passed to the BMODULE. The standard Block Modules ignore the BPARM.

CHKSUM

This parameter specifies the default if a checksum value is to be passed between 2200 systems when doing trackio. The default is false.

CLOSE

This optional parameter specifies whether SCLOSE should be used to terminate the file transfer or not (instead of SDISC). IF SCLOSE is used, any other BFX components to be communicated with (BFXTI, BFXTR, and BFXJS) must also use SCLOSE. It is recommended to use SCLOSE if all BFXs support it. The default is *false* (use SDISC).

DTIME

This parameter specifies the number of seconds between SCONNect tries. If for some reason the SCONN fails, up to DTIME retries are attempted before BFX aborts the file transfer. DTIME specifies the delay time between these retries. The default is 10 seconds

DTIMES

This parameter specifies the number of attempts for an SCONNECT. A failure with error code 3502 (application busy) or 3501 (application not offered) will be retried DTIMES times with an intervening delay of DTIME seconds. Other errors are not retried. The default for DTIMES is 20.

FILE

This parameter specifies the default assigned name of the file to use during the transfer. The default is OUTFILE.

HOBOPT

This parameter specifies the default value for the HOBCHECK parameter.

HOSTCHK

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. This option should not be specified in any of the following conditions:

- If dissimilar NCTs are used on each of the connecting hosts
- If group host names are used
- If a local loopback host name is used (NTXLCLxx)

The default for H301 is TRUE.

The default for H301L is FALSE

JBLOCK

This parameter specifies the default block size to use when transmitting the jobfile over the network.

JFILE

This parameter specifies the default name to use in BFXJS if the JOBFILE parameter is not specified.

JFLAG

This optional parameter controls whether or not this is a job submission. If JFLAG is true, then a job file is sent to the remote host. No attempt will be made to send a data file after the job file transfer. The job file will be sent to BFXJS using the JID as a connect process name. If JFLAG is false, then a data file will be sent to the remote host. No job file will precede the transfer unless this is the first data file transfer. The default is false.

JID

This parameter specifies the process name that is to be passed to NETEX to establish a connection with the BFXJS module on the remote host. The default is JID = BFXJS.

JRMAXL

This parameter (valid only for BFXTI) specifies the maximum length of the records for the jobfile in words. The default is 28 words. (Note: This parameter is ignored in the jobfile for secure jobfile transfers.)

MODE

This parameter specifies what the default MODE for transferring files will be. Mode=0 specifies bit mode. Mode=1 specifies ASCII mode. MODE=2 specifies Fieldata. The default is MODE=1.

MSGLV

This parameter specifies the default "verbosity" of the messages to be printed. The default is MSGLV= 4; that is, the significant events during the transfer process are logged. (See MSGLVL parameter in "Control Statements Parameters" on page 17.

RBM

This parameter is valid for H301L only. It specifies the remote machine is byte addressable. The default is FALSE.

RMAXL

This parameter specifies the maximum length of the records (in words) in the file being transferred. The default is 28 words.

RMOD

This parameter specifies the alphabetic name of the default Record Module to be used for transferring the file. If not specified, the default for H301 is STANDARD, which is a symbolic name that calls the Network Executive Software-provided Record Modules. For H301L the default is USSENDER and USRECVER

RPARM

This parameter specifies a default string of character parameters to be passed to the invoked Record Module. This string, if provided, may be up to 64 characters in length. By default, a string of all blanks is passed to the RMODULE. The standard Record Modules ignore the RPARM.

SEND

This optional parameter controls the direction of the file transfer. If SEND is true then the file will be sent to the remote host. If SEND is false then the file will be received from the remote host. The default is *true*

SENDJB

This parameter specifies whether or not to send a job file to BFXJS by default. Files may be transferred using automatic job submission or ·manual job submission (described in "Introduction" on page 1). Using automatic job submission, a BFXTR job file is sent to BFXJS to be started. Once started, the job will connect to BFXTI. Using manual job sub- mission, BFXTI is started first on the local host. BFXTR is then started by the operator on the remote host. BFXTR will then connect to BFXTI. If SENDJB is true then a job file will be sent to the remote host. If SENDJB file is false no job file is sent. The default is SENDJB= .TRUE.

TIMEFL

This parameter specifies if the user wants a time stamp printed with the BFX messages by default. The default is TIMEFL=.TRUE.

TIMEOF

This parameter specifies the maximum number of seconds that a BFXTI job will wait for a BFXTR job to be started on the remote host. This also is used to time out the offer of BFXJS. However, when this occurs, BFXJS will always re-offer and not terminate. The default is 240 seconds.

TIMEOU

This parameter specifies the maximum number of seconds that a NETEX Read will remain outstanding. The default is 60 seconds.

TO

This parameter specifies a default TO or FROM host address to be used **if** this parameter is omitted by the BFX user. This parameter is particularly useful in the environment where nearly all file transfers are between two processors. The default for H301 is LOOPBAK. The default for H301L is spaces.

Addition of User-Written Modules

As distributed the BFX program supports and uses the standard Block and Record Modules to copy sequential files. BFX is designed to allow user- written modules to be added at will. (Refer to “Appendix A: User Modules” on page 45 for more information about user-written modules.) For a user-written module to be added to BFX, one or more entries must be added to the BFXUIM subroutine (in the BFXUSER element). The subroutine tests for a user-installed module in two parts (the request for sending modules and the request for receiving modules).

Format for Adding User Modules

Each user module combination (for Block and Record Modules) must be explicitly added for each desired block/record combination. The combination can be added to either the send or receive mode module list. Each entry is made of a FORTRAN blocked-IF statement. The blocked if statement determines if the RMOD/BMOD combination is valid. If the combination is valid a call (to BFXSND if the file is to be sent, or to BFXRCV if the file is to be received) will be initiated. The format of the call and the blocked-IF is given below:

```
      IF ((BMOD.EQ.'blkname ').AND.(RMOD.EQ.'recname ')) THEN
        CALL bfx---(TIMEOF,TIMEOU,SCOND,FILE,ID,TO,RPARM,BPARM,
  +                 BFXCON,******,++++++,ABRTD,MODE)
        GO TO 5
      ENDIF
```

Where:

blkname Referenced (BMOD) name of the Block Module.
recname Reference (RMOD)name of the Record Module.
bfx--- BFXSND for send mode, BFXRCV for receive mode.
BFXCON Parameter passed to BFXUIM. 'When BFXUIM is called, this parameter is either BFXSOF (offer routine) or BFXSCN (connect routine).
****** Assembled name of the Block Module. This name should also be defined as 'EXTERNAL'.
++++++ Assembled name of the Record Module. This name should also be defined as 'EXTERNAL'.
ABRTD Indication of whether transfer successful ABRTD is set to TRUE if the transfer failed.

Placement of Statements

The assembled (external) name of the Block/Record Modules should be added to an EXTERNAL statement near the beginning of the BFXUIM subroutine, after the comment block 'EXTERNAL DEFINITIONS FOR EACH OF THE REFERENCED USER BLOCK/RECORD MODULES GOES HERE.' The blocked if statement for the send Block/Record Modules should be added after the comment block 'TEST FOR SENDING RECORD/BLOCK MODULES GOES HERE.' The blocked if statement for the receive Block/Record Modules should go after the comment block 'THE USER INSTALLED RECEIVE RECORD/BLOCK MODULES GOES HERE.' Editing of BFX source modules is not encouraged. Changes to the BFX product should only be made to the BFXUSER module. It is the user's responsibility to incorporate the local changes to BFXUSER in future releases.

Appendix A: User Modules

For applications where custom coding is required, BFX allows for the incorporation of user-written modules to read and write file data and to process jobs before delivery to the remote host. The user may write Record or Block Modules to perform this function.

Users may wish to write Block Modules under the following conditions.

- Character sets or assembly/disassembly modes are used that BFX does not provide.
- Users wish to supply logical blocks.

User-written Record Modules should satisfy most other special needs not provided by BFX.

It should be noted that the standard Block Modules are highly specialized; they call Record Modules. User-written Block Modules would not necessarily call Record Modules.

BFX supports user code that is written either in Assembler language or in a higher level language. High level languages are written as subroutines, external procedures, etc., rather than as main programs.

After any user module has been coded, it will have to be added to the main BFX module. See "Installation" on page 25 for a description of the installation of user modules.

The rest of this section describes the writing of Record and Block Modules. Appendix A provides BFX internal information which may be helpful when writing Block Modules.

Writing Record Modules

Record modules are called by NESi block modules and are designed to be entered from a single entry point. (User-written block modules would probably not call record modules.) The first time the record module is called, it is expected to open a file for output or input. On subsequent calls, the record module will need to either provide or accept a logical record each time it is called. Record modules can also insert messages to be sent to the other side of the BFX transfer process.

It should be noted that the conventions used here are those enforced by the standard NESi bock modules. If the user writes their own block modules, the user block modules may or may not call record modules, or may call a record module using completely different conventions.

Record modules are normally written to be either sending or receiving record modules. The argument list passed to both types of modules is identical and is described below. The use of these parameters differs between module types and is discussed in the parameter description.

The following paragraphs describe the FORTRAN entry to the record module, Assembler entry to a record module, a block and record module entry and exit summary, and examples of a sending and receiving record module.

FORTRAN Entry

The entry to the record module should be declared as shown in Figure 14.

```
SUBROUTINE rmod (BUF,BUFLEN,BUFLEV,MSG,MSGLEN,MSGLEV,
+                 RPARM,MODE,DDNAME)
 type BUF(1)
 INTEGER*4 BUFLEN,BUFLEV,MSGLEN,MSGLEV,MODE
 CHARACTER*128 MSG
 CHARACTER*64 RPARM
 CHARACTER*8 DDNAME
```

Figure 14. FORTRAN Record Module Entry Parameters

The following parameters appear in Figure 14:

rmod

This parameter is the record module name.

BUF

This parameter specifies the start of the logical record information. For a receiving record module, the logical record information will start at the beginning of the BUF array. For a sending record module, the record should be placed starting at the beginning of the BUF array.

BUFLEN

This parameter specifies the length of the logical record in bytes. When the record module is called for the first time, BUFLEN will have the maximum logical record length permissible with the user's BLOCK parameter. If the record module decides this value it can change the value of BUFLEN (and hence the BFX buffer size) before returning to the block module. On subsequent calls, the receiving record module will obtain the logical record length in this parameter. Sending block modules should specify the length of the outgoing record with this variable.

BUFLEV

This parameter specifies a binary number that contains the delimiter level of the record to be sent or accepted. This is a binary value in the range of 0 to 15. Normally, this value is one to indicate the normal end of a logical record. A value of 15 indicates that this record is to be the last record transferred, referred to as End of Information in the BFX specifications. Intermediated values are used by computer operating systems that have a hierarchy of end delimiters in their file structure, such as the EOR, EOF, and EOI indications in CDC CYBER data files.

BUFLEV has a second use: to signal when the record module is entered for the first time. On that first entry, BUFLEV will have a value of -1. On all other entries, it will have a value greater than or equal to zero. Thus BUFLEV < 0 can be used as a conclusive test for first time (initialization) entry to the record module.

If the record module is receiving data, then BUFLEV will contain the delimiter level of the file delimiter following the record provided in BUF. A record may or may not accompany this delimiter; if it does not, then BUFLEN will be set to zero. If the passed BUFLEV is 15, then this will be the last call to the record module. The module should perform whatever close processing is needed before returning. It is good form to return a message in the MSG parameter (see below) to summarize the status of the file transfer.

If the record module is sending data, then the value of BUFLEV passed to the record module is used to signal end of transfer. If BFX detects that the transfer cannot proceed for some reason, then it will call the record module with the value 15 in BUFLEV. This indicates that the file transfer process is being aborted, and that the record module should perform whatever termination processing is needed and return for the last time to the block module. During normal processing, BUFLEV will be passed to the record module with a zero value.

For normal operation of a sending record module, BUFLEV must be set by the module before returning. Normally BUFLEV should be set to one to indicate a normal logical record. If the sending module decides to terminate the transfer, then BUFLEV should be set to the EOI value (15). If return is made with BUFLEV= 15 then control will not be returned to the record module. When this end indication is returned, it is good form to return an alphabetic message in the MSG parameter to indicate the status of the entire file transfer.

MSG

This parameter specifies an area that allows alphabetic information to be returned to the calling block module and then to the main body of BFX. If the record module wants to return a message, it should place a string of EBCDIC character information in this parameter. The maximum length of the message is 128 bytes.

MSGLEN

This parameter indicates the length of the message returned by the record module. The length passed to the record module is zero. If a zero MSGLEN is returned, it is assumed that no message is present.

MSGLEV

This parameter indicates the importance of the message returned to the calling block module. This should be specified as a decimal value between 0 and 15. If the value of the message is greater than or equal to MSGLEV, then the message will be sent on the local SYSPRINT log. If it is to be printed, the block module will be sent over the NETEX connection to the opposite BFX program. If the level of the message is greater than or equal to the MSGLEV of the remote BFX program, then it will be printed on the remote BFX's SYSPRINT file.

If MSGLEV= 15, then the message is considered to be a terminal error. Control will not be returned to the record module, so any needed termination processing should be done before returning with this value.

RPARM

This parameter is the 64-byte string of information provided by the user in the RPARM field of the BFX control parameters. If this field was not specified, then this field will contain 64 blanks.

MODE

This parameter is the BIT or CHAR file mode specified by the user in the control parameters. If MODE=BIT was specified, then this parameter will be set to zero. If MODE= CHAR was specified, then the mode will be set to one. This field is provided for informational purposes only; changing its value will have no effect.

DDNAME

This parameter contains the DDNAME of the file to be moved. A receiving record module will get the OUTDD = DDNAME or the FILEOUT default in this field. A sending record module will have the INDD field placed in this parameter. This information is for informational purposes only; the record module is not obligated to use this DDNAME or any other to provide or accept file information.

type

This parameter is the type of record module: RREC (Receive Record Module) or SREC (Send Record Module)

Assembler Record Module Entry

Assembler modules written to satisfy the FTN calls must follow FTN coding conventions:

- X11 is the return address; (that is, `J 0,X11` to return from the module).
- A0 is the pointer to the parameter list address; (that is, the address at which to load the first parameter):

```
L,U   A1,0,A0     . A01 is the # of parms
L     A2,*0,A1    . A2 contains parm 1
```

  The parameters and their order are described above.

- The minor register set may be destroyed by the called module (X8-X11,A0-A5,R1-R3).

Block and Record Module Exit and Entry Summary

Table 1 on page 48 through Table 4 on page 50 summarize the use of the BUFLEN, BUFLEV, MSGLEN, and MSGLEV parameters upon exit from the BFX default block module and entry to the user-written record module.

BUFLEN	BUFLEV	MSGLEN	MSGLEV	Significance
0	-1	0	0	Record module is being called for the first time.
0	0	0	0	Normal data transfer. Data or message should be returned.
0	16	0	0	File transfer aborted. Record module may return a message to be printed locally. Always last time record module is called.

Table 1. Sending Record Module Entry Indications

BUFLEN	BUFLEV	MSGLEN	MSGLEV	Significance
>=0	0-14	0	0-15	Normal data transfer. If BUFLEN = 0 a zero length record will be sent to the receiver.
0	0-14	> 0	0-15	Normal message provided. Message printed locally and sent for printing on receiver. Rmod will be called again. Data will be ignored.
> 0	0-14	> 0	0-15	Both message and data are provided. Data will be delivered to receiver, followed by a message. The message will be printed locally.
0	15	0	0-15	EOI with last record sent already. Last call to sending module.
> 0	15	0	0-15	EOI following included record. Last call to sending module.
0	15	> 0	0-15	EOI message provided. Message is printed locally and sent for printing on receiver. Last call to the record module. Last call to sending module.
> 0	15	> 0	0-15	EOI message and last buffer provided. Buffer is forwarded to the receiver. Message is printed locally and forwarded to the receiver. Last call to sending module.
0	16	0	0-15	Abort transfer. All data provided before will be delivered to the receiver, along with a default abort message. Receiver will pass the abort to the receiving module.
> 0	16	0	0-15	Abort transfer after sending data. All data provided will be delivered to the receiver, along with a default abort message. Receiver will pass the abort to the receiving module.
0	16	> 0	0-15	Abort transfer. All data provided before will be delivered to the receiver, along with the provided abort message. Receiver will pass the abort to the receiving module.
> 0	16	> 0	0-15	Abort transfer after sending data. All data provided will be delivered to the receiver, along with the provided abort message. Receiver will pass the abort to the receiving module.

Table 2. Sending Record Module Return Actions

BUFLEN	BUFLEV	MSGLEN	MSGLEV	Significance
0	-1	0	0	Receiving module is being called for the first time.
> = 0	0-14	0	0	Normal data transfer. Data should be processed. Non-ending delimiter should be processed. If BUFLEN=0, then a zero length record is being provided.
0	15	0	0	Normal EOI. The receiving module should close the file and return with an optional message. Last time receiving module is called.
> 0	16	0	0	File transfer aborted. Receiving module should close the file and return with an optional message. Last time receiving module-is called.

Table 3. Receiving Record Module Entry Indications

BUFLEN	BUFLEV	MSGLEN	MSGLEV	Significance
--	0-15	0	0-15	Normal return from data accept.
--	0-15	> 0	0-15	Normal return from accept with message provided. Message will be printed locally and sent back to the sender.
--	16	0	0-15	Abort transfer. A default message will be generated, printed locally, sent as an abort message to the sending BFX. The module will not be called again.
--	16	>0	0-15	Abort transfer. The provided message will be printed locally and sent as an abort message to the sending BFX. The module will not be called again.

Table 4. Receiving Record Module Return Actions

Writing Block Modules

The block modules are also designed to be entered from a single entry point. They differ from the simpler record modules specified above in several ways:

- They take complete responsibility for providing and accepting blocks of information from NETEX. All protocol needed by the block modules, such as record lengths or end of file indications, must be passed from the sender to the receiver through the file buffer.
- The block modules have total control over the NETEX DATAMODE parameter. They can set or examine this value to send specialized data formats.
- Unlike record modules, block modules cannot insert messages to be sent to the other side of the BFX transfer process. If they want to send information along with the file data, it must be embedded in the file protocol used by the block modules.

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

IMPORTANT: When writing modules that call NETEX, it is important to clear the high byte of register 13 prior to calling NETEX. If a flag occupies this register when NXMUIFOO is called, an abnormal termination will occur.

FORTRAN Entry

The entry to the block module should be declared as shown in Figure 15.

```
SUBROUTINE bmod (BUF,BUFLEN,BFUBIT,BDTMOD,
+                  BUFLEV,MSG,MSGLEN,
+                  MSGLEV,LCM,MODE,DDNAME,
+                  BBPARM,RMOD,BRPARM)
 type BUF(1)
 INTEGER*4 BUFLEN,BFUBIT,BDTMOD,BUFLEV,MSGLEN
 INTEGER*4 MSGLEV,LCM,MODE
 CHARACTER*128 MSG
 CHARACTER*64 BRPARM,BBPARM
 CHARACTER*8 RMOD,DDNAME
```

Figure 15. Block module entry

The parameters in Figure 15 are described in the following paragraphs.

bmod

This parameter is the name of the block module.

BUF

This parameter defines the start of the block. For a receiving block module, the block information will start at the beginning of the BUF array. For a sending block module, the block to be sent over NETEX should be placed starting at the beginning of the BUF array.

BUFLEN

This parameter specifies the length of the block in bytes. When the block module is called for the first time, BUFLEN will have the maximum logical block length permissible with the user's BLOCK parameter. If the block module dislikes this value it can change the value of BUFLEN (and hence the BFX buffer size) before returning to the control module. On subsequent calls, the receiving block module will obtain the physical block length in this parameter; sending block modules should specify the length of the outgoing block with this variable.

BFUBIT

This parameter is the NETEX unused bit count. This value is principally used for sending or receiving precise amounts of file information to systems whose word sizes are not a multiple of eight bits.

If the block module is a sending block module, then this value will be zero when the module is called. If it wants to use the unused bit count facility, then it should place a nonzero value in the field. NETEX will perform algebra on the DATAMODE and total number of useful bits sent and will provide a resultant length and unused bit count to be delivered to the receiving block module.

The receiving block module will have this value set on entry. It may use this field or ignore it.

BDTMOD

This parameter is the DATAMODE field that is used to support the optional code conversion and assembly/disassembly features of NETEX. Each time a transmitting block module is called, the

BDTMOD field will contain the DATAMODE that was negotiated at connection time based on the system types and the MODE= parameters specified by both BFX implementations. The block module may leave this value unchanged, or it may supply a DATAMODE in this field that will be used to transfer the data buffer supplied by this call.

A receiving block module will have this value set to the incoming DATAMODE of the block that is delivered to the block module.

BUFLEV

This parameter is the same as the BUFLEV parameter for record modules.

MSG

This parameter specifies an area that allows alphabetic information to be returned to the calling control module. If the block module wants to return a message, it should place a string of EBCDIC character information in this parameter. The maximum length of the message is 128 bytes. Unlike the record module, this message will only be logged locally; messages sent between BFX programs must be accommodated in the protocol implemented by the block modules.

MSGLEN

This parameter indicates the length of the message returned by the block module. The length passed to the block module is zero; if a zero MSGLEN is returned, it is assumed that no message is present.

MSGLEV

This parameter indicates the importance of the message returned to the calling block module. This should be specified as a decimal value between 0 and 15. If the value of the message is greater than or equal to MSGLVL, then the message will be sent on the local SYSPRINT log.

If MSGLVL= 15, then the message is considered to be a terminal error. Control will not be returned to the block module, so any needed termination processing should be done before returning with this value. Before returning with MSGLVL = 15, a sending block module should already have sent or concurrently send an end of transfer indication to its corresponding receiving block module so that the receiving end may properly terminate.

LCM

This parameter specifies the least common multiple addressable word size that was determined during the connection negotiation process. Every logical record contained within the block should begin at a multiple of LCM bytes from the beginning of the block. The result is that the receiving block module will have all its information starting on a word boundary. The block module on the other side will have a corresponding LCM value so that the information sent to a receiving UNISYS block module will also begin on a multiple of LCM bytes.

If the system containing the other BFX is an UNISYS host or another byte addressable system, then LCM will be set to 1.

Also note that the use of LCM is not required; the block modules may use any convention they find appropriate to transfer logical records.

MODE

This parameter specifies the BIT or CHAR file mode specified by the user in the control parameters. If MODE=BIT was specified, then this parameter will be set to zero; if MODE= CHAR was specified, then the mode will be set to one. This field is provided for informational purposes only; changing its value will have no effect.

DDNAME

This parameter is the DDNAME of the file to be moved. A receiving block module will get the OUTDD DDNAME or the FILEOUT default in this field. A sending block module will have the INDD field placed in this parameter. This information is for informational purposes only; the block module is not obligated to use this DDNAME or any other to provide or accept file information.

BBPARM

This parameter is the BPARM character string passed to the block module.

RMOD

This parameter is the Record module called

BRPARM

This parameter is the RPARM character string passed to the record module.

type

This parameter specifies the type of block module: RBLK (Receiving Block Module) or SBLK (Sending Block Module).

Assembler Block Module Entry

Assembler modules written to satisfy the FTN calls must follow FTN coding conventions:

- X11 is the return address; (that is, `J  0,X11` to return from the module).
- A0 is the pointer to the parameter list address; (that is, the address at which to load the first parameter):

```
L,U    A1,0,A0        . A01 is the # of parms
L      A2,*0,A1       . A2 contains parm 1
```

  The parameters and their order are described above

- The minor register set may be destroyed by the called module (X8-X11,A0-A5,R1-R3).

Writing Job Submission Modules

Job submission modules are designed for those installations that have other means than the BFXJS program to submit jobs to other hosts. The inclusion of user modules allows any means of programmable job submission to be employed by the installation.

Job submission modules are designed to be written in Assembler language by an installation systems programmer.

When the job submission module is called, it is passed a pointer to the completed transfer option list data structure in R1 (see Data Structures in “Appendix B. BFX Internal Summary" on page 55). It may obtain or generate a job from any source, and submit the job through any means available in the user programming environment. On return, it should set the TROLPRC code to indicate if the job submission was successful or not.

Frequently, installations may want to modify or generate extra statements in the job submitted by the user, using the standard RMTJOB DD input and BFXJS job submission techniques. Rather than coding an entirely new JMODULE, it will frequently be easier to provide a new RMODULE to be used during the job submis-

sion process. This RMODULE merely needs to provide the job text to its caller one statement at a time, and may insert or modify records from the RMTJOB file as its logic chooses.

If a record module is written in lieu of a new job submission module, it must be entered as a record module.

Appendix B. BFX Internal Summary

The following paragraphs briefly summarize the internal structure of BFX as it relates to writing user modules. Table 5 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 5 are three block diagrams showing the interaction of these modules in each of the three BFX programs.

Table 5. BFX Default modules

Module	Description
BFXSJS	BFXSJS is an extended mode program that listens for a request to receive an encrypted job file.
BFXTI	BFXTI is an extend mode program that determines if the job file needs to be securely sent to the BFXSJS server. If a secure connection is required, the encrypted file is sent, and the BFXTIB program is executed to do the actual data transfer. The BFXTI will call the same version of the program as the BFXTI program. If BFXTI/1102 is executed, BFXTI will call BFXTIB/1102 to do the data transfer.
BFXTIB	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.

Table 5. BFX Default modules	
Module	**Description**
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 CONNECT-ing 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.
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.

Table 5. BFX Default modules	
Module	**Description**
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.)
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.
BFXSET	BFXSET upon first execution will set all the default parameters for the file transfers. Upon subsequent transfers BFXSET will set only a subset of the default parameters.
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 16 is a block diagram for the BFXTI program. BFX Module Descriptions on page 60 describes each component of the block diagram.

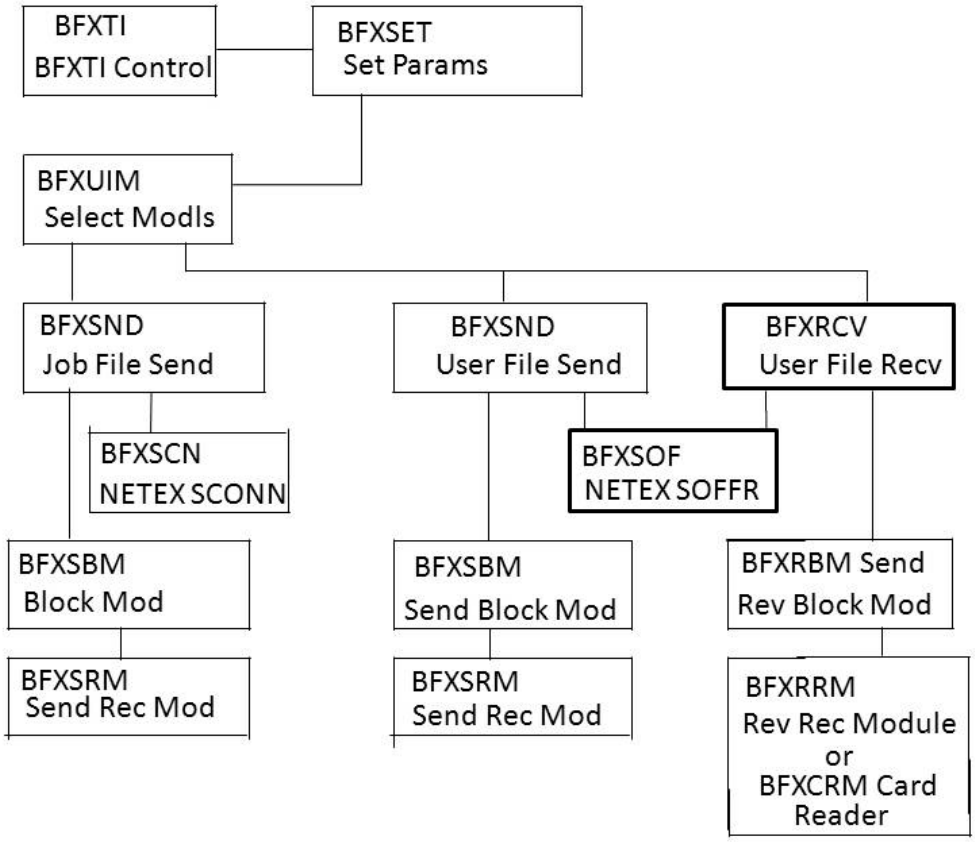

Figure 16. BFXTI Module Block Diagram

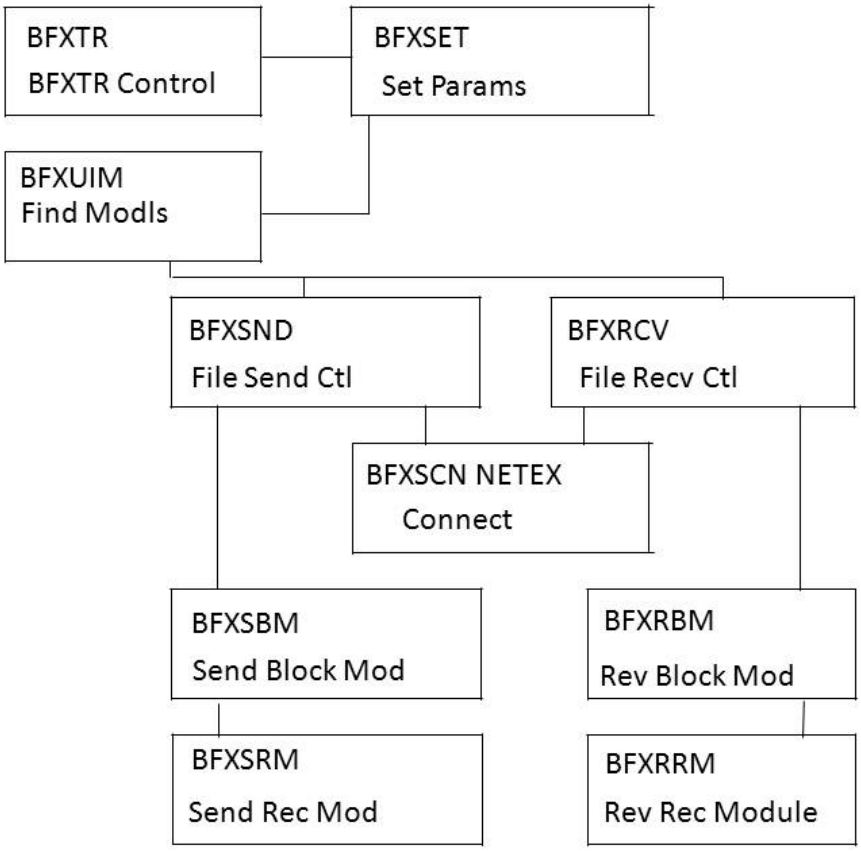

Figure 17. BFXTR Block Diagram

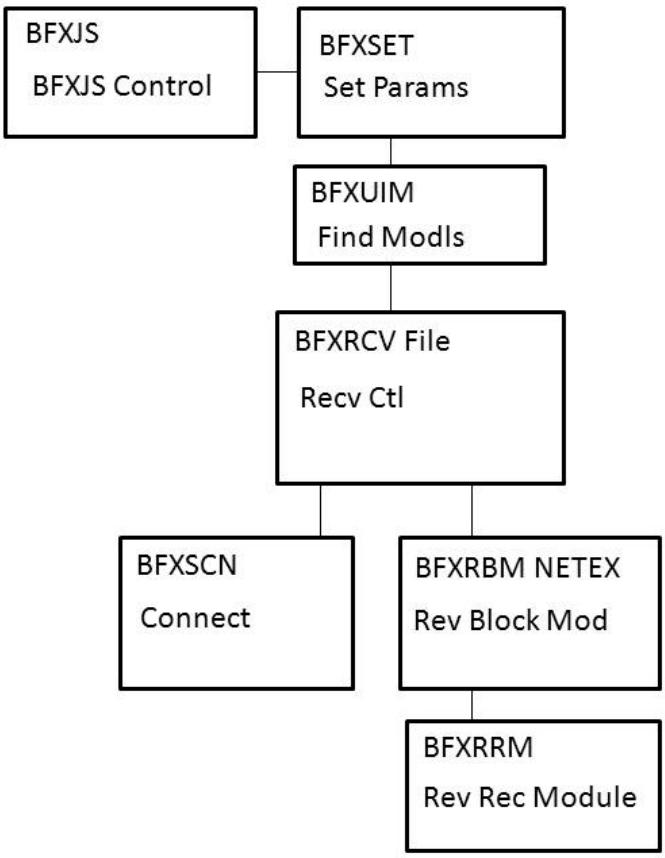

Figure 18. BFXJS Module Block Diagram

BFX Module Descriptions

The following paragraphs describe seven of the default BFX modules. Understanding these modules will help the reader write their own user 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 processing of the incoming record with the file writes performed by the Block Module. Figure 18 shows the normal flow of data transfer.
4. Whenever the Block Module is called to write a block received from NETEX, 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.

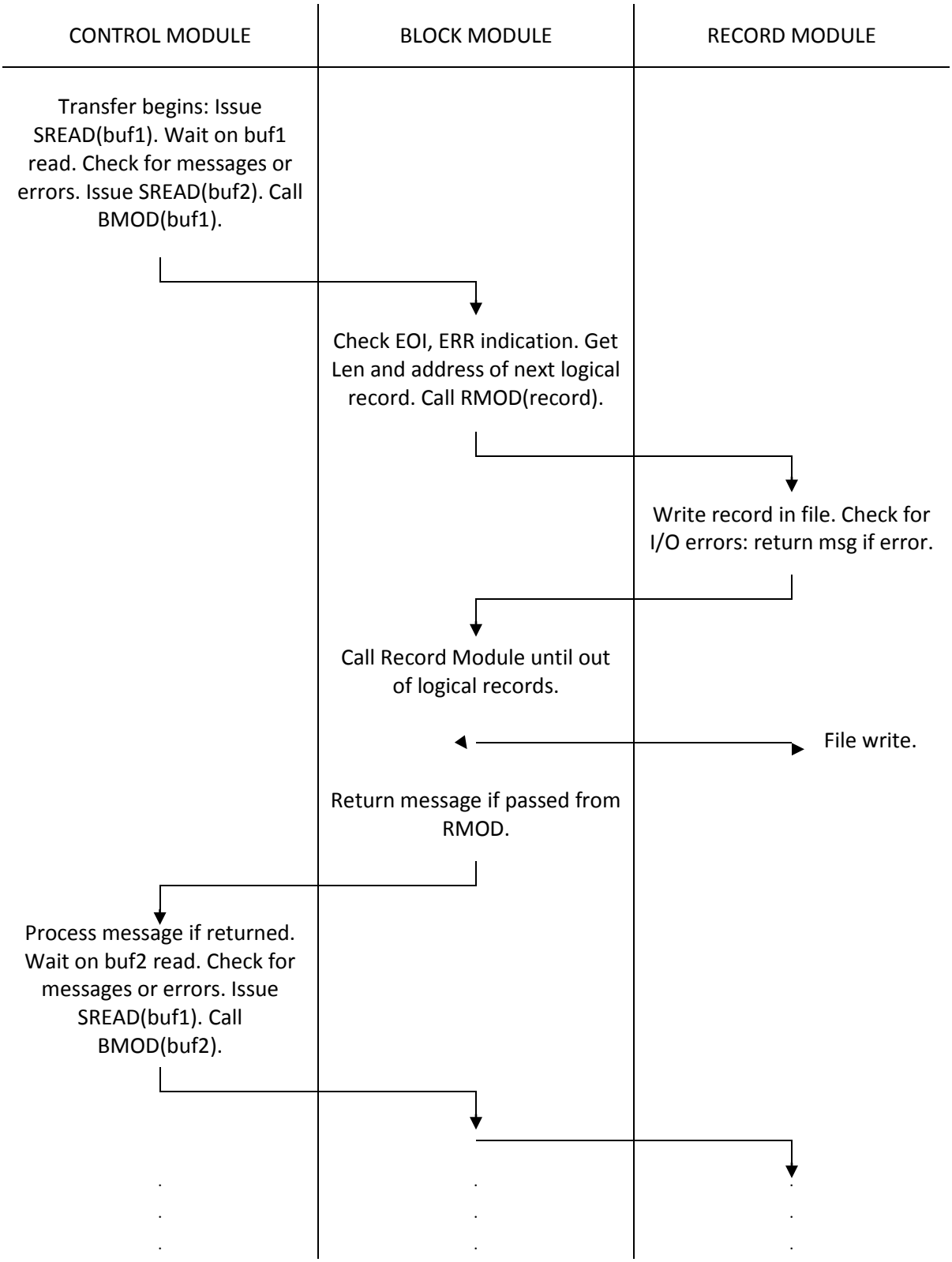

Figure 19. File Receive Data Flow

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 19.
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 processing of the incoming record with the file writes performed by the Block Module. Figure 20 below shows the normal flow of data transfer.
4. Whenever the Block Module is called to provide a block to be sent to NETEX, 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.

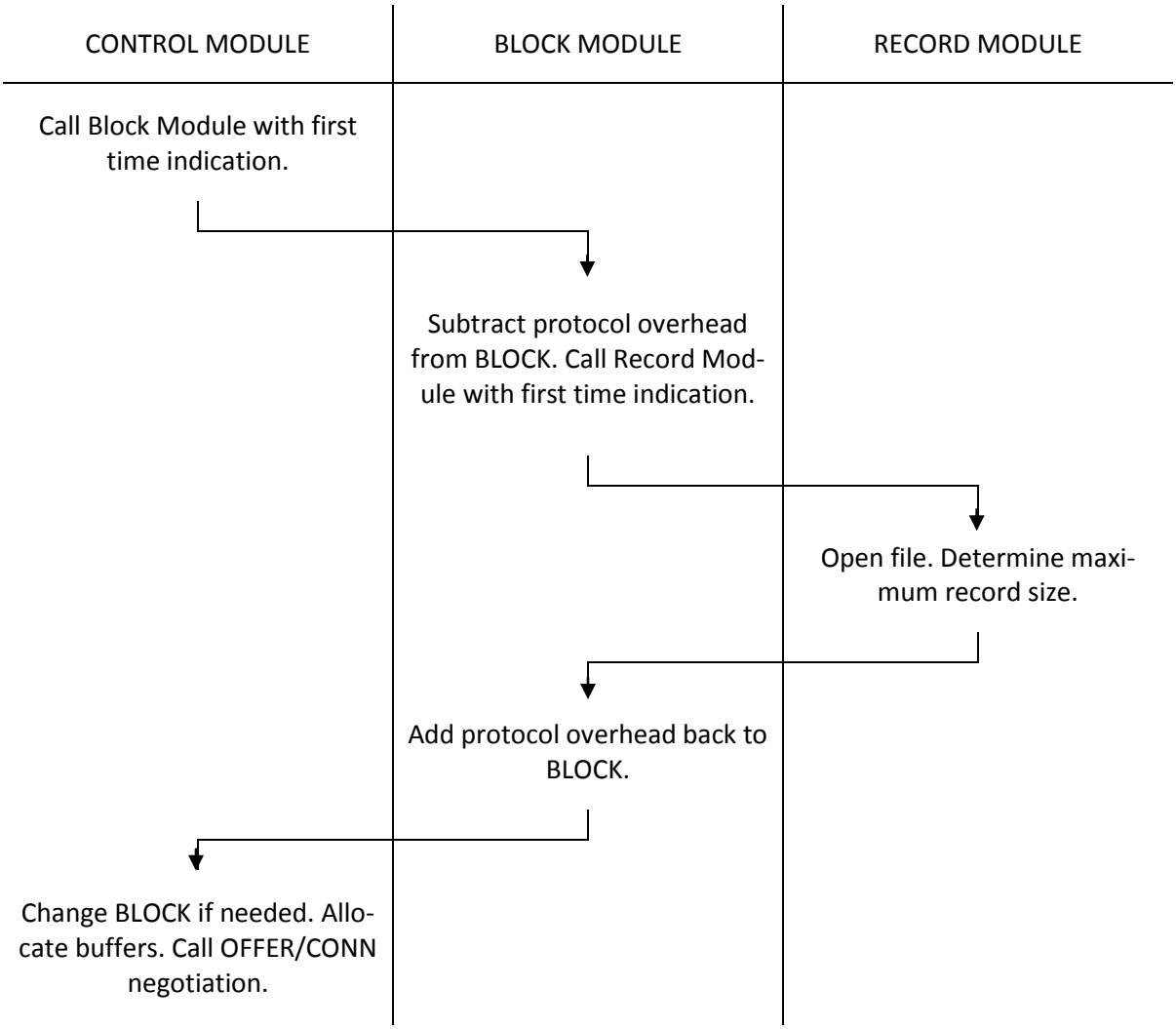

Figure 20. Send Receive Initialization Flow

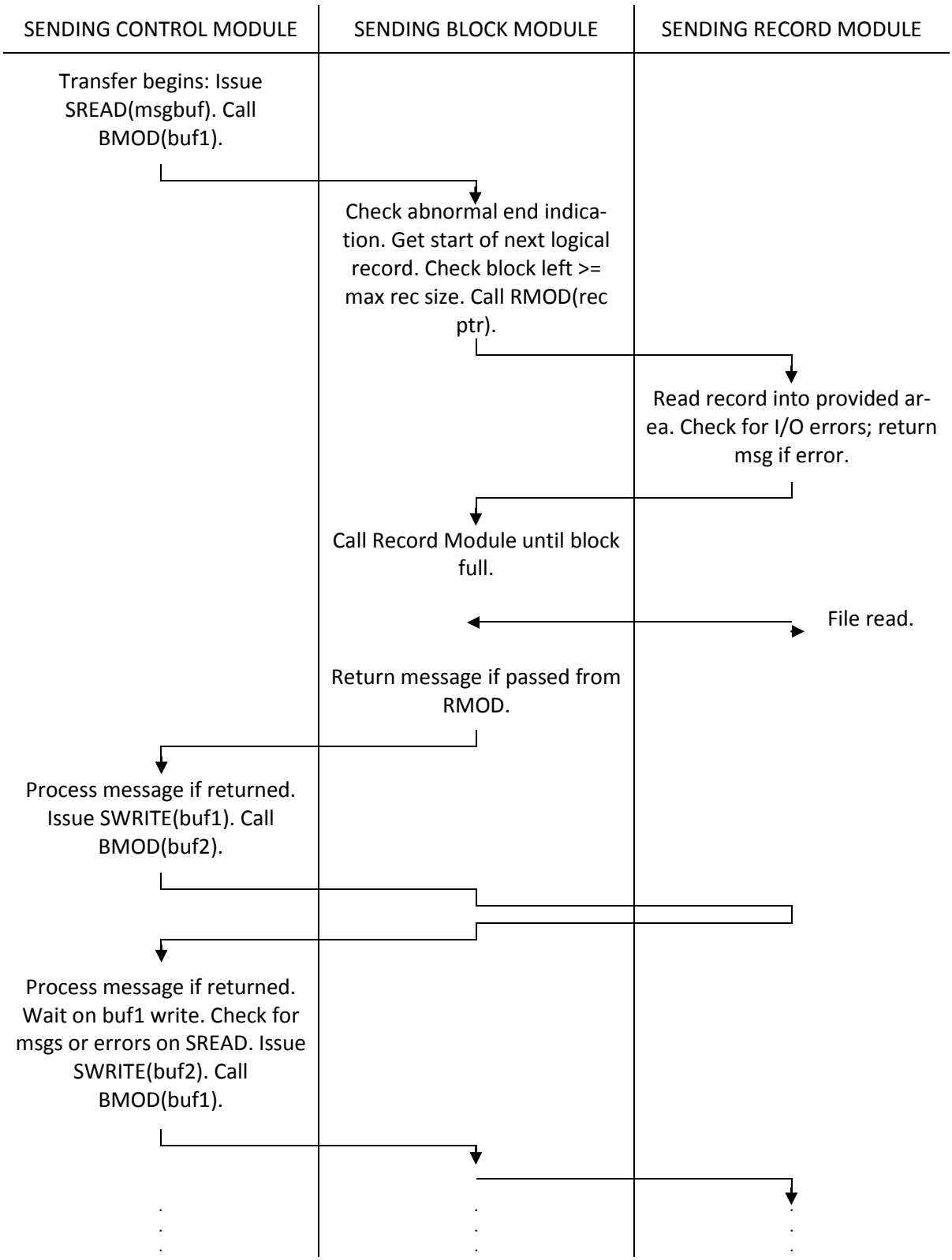

Figure 21. File Send Data Flow

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. It is the responsibility of the receiving block module to decode all file transfer protocol information contained within the NETEX 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 C. BFX Error Messages

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

The messages, which are sent to the PRINTS file, have the following format:

```
BFXnnns message text
```

BFX - indicates that this is a BFX error code.

nnn - the error number. The BFX messages are listed in this order.

s - the message severity. The following codes are used:

- **I** - informational messages
- **E** - error messages
- **S** - parameter severe error messages
- **W** - warning messages
- **F** - fatal error messages
- **R** - messages requiring an operator response

message text - the message text.

The following are the messages issued by BFX.

BFX001F JOB SUBMISSION FAILED.

Severity: 15 (Fatal error)
Explanation: Transfer Initiate was unable to submit a job to the remote host. The BFX program will terminate.
Response: The reason for job submission failure will be indicated in a previous message. Take the corrective action indicated by the previous message description.

BFX006W "xxxxxxxxxx" NOT RECOGNIZED AS A DIRECTIVE.

Severity: 15 (Severe error)
Explanation: When processing the parameters, an unrecognizable parameter was found, "xxxxxxxxxx" contains the ten characters of the parameter in error. Generally, BFX will not transfer any files after encountering this error. The directive syntax scan will continue.
Response: Correct the syntax error and resubmit the BFX job.

BFX007W Parameter "nnnnnnnn" NOT VALID FOR THIS JOB -- IGNORED.

Severity: 9 (Recoverable error)
Explanation: The parameter specified by “nnnnnnnnn” is not applicable to the BFX program. The parameter in question is included in the error message. The statement is ignored.
Response: Although processing will continue, the probable cause is an operations or setup error. Verify that the remainder of the BFX run proceeded as intended.

BFX010F TO = or FORM = HOST NAME OMITTED.

Severity: 15 (Fatal error)

Explanation: The control parameters for the first statement of either a BFXTI or BFXTR run did not specify the name of the opposite host to allow a connection to take place. Furthermore, a default host was not specified during installation of the BFX program.

Response: Supply the TO = or FROM = parameter required and rerun the job.

BFX011F ID = BFX IDENTIFIER OMITTED.

Severity: 15 (Fatal error)

Explanation: The ID parameter which uniquely identifies the BFX job on the initiating machine was not supplied. There is no default for this parameter.

Response: Supply the ID = parameter and rerun the job.

BFX012F SPECIFIED BUFFER SIZE TOO LARGE.

Severity: 15 (Severe error)

Explanation: The BLOCK = or JBLOCK= parameter specified a value greater than 8192 words.

Response: Correct the BLOCK = or JBLOCK= parameter and resubmit the job.

BFX014I ERROR(S) PREVIOUSLY FOUND. EXECUTION OF TRANSFER BYPASSED.

Severity: 4 (Detailed informational)

Explanation: The user specified SOE, and errors were encountered. The syntax of the directives for following transfers will be checked, but the transfers will not take place.

Response: Correct the errors as described in the preceding error messages and resubmit the job.

BFX015I PROCESSING HAS BEGUN.

Severity: 4 (Detailed informational)

Explanation: The directives for the file transfer have been processed, the file transfer is about to commence.

Response: None.

BFX016I *host* not found using *host1*

Severity: 4 (Detailed informational)

Explanation: DNS returned a not found condition on *host* name. A second attempt was used using *host1*

Response: none.

BFX017I START OF FILE TRANSFER NUMBER: i

Severity: 4 (Detailed informational)

Explanation: The BFX program keeps a count of the current file transfer.

Response: None.

BFX017IC JOB: nnnnnnnnnnnnnni

Severity: Continued informational)

Explanation: When sending a secured jobfile, the job card is displayed.

Response: None.

BFX018W SPECIFIED RMAXL (or JRMAXL) SIZE TOO LARGE.

Severity: 15 (Severe error)

Explanation: The value assigned to RMAXL was greater than 2499 words in ASCII mode, 1666 words in Fieldata mode or RMAXL exceeds the BLOCK size given.

Response: Correct the record size value given.

BFX019W UNRECOGNIZABLE DIRECTIVE VALUE vvvvvvvvv.

Severity: 15 (Severe error)

Explanation: The value assigned to the directive was of improper syntax, missing, or greatly exceeds the maximum value assignable to the directive.
Response: Correct/supply the directive

BFX020F NETEX COMMUNICATIONS SUBSYSTEM IS NOT RUNNING.
Severity: 15 (Fatal error)
Explanation: When the BFX programs attempted to establish communications, they found that the NETEX subsystem was not running in the local host at the current time. Execution is terminated, as transfer of data is not possible at the current time,
Response: Consult with computer operations to determine if NETEX should have been active at the time of the BFX run. Resubmit the job when NETEX is active.

BFX021 DELETE *filename* AND RERUN.
Severity: 15 (Fatal error)
Explanation: BFXSJS attempted to do a secure connection to a remote system. When trying to assign the filename *filename*, an error was returned. An attempt to catalog the file and reassign the file also failed. (See SDEFAULT.C for enqueue_name).
RESPONSE: Manually delete the filename listed, and rerun the job. If the error persists, manually delete the file, and catalog the file looking at the return codes.

BFX021 DELETE *filename* AND RERUN.
Severity: 15 (Fatal error)
Explanation: BFXSJS attempted to do a secure connection to a remote system. When trying to assign the filename *filename*, an error was returned. An attempt to catalog the file and reassign the file also failed. (See SDEFAULT.C for enqueue_name).
RESPONSE: Manually delete the filename listed, and rerun the job. If the error persists, manually delete the file, and catalog the file looking at the return codes.

BFX029W SPECIFIED MSGLEVEL TOO BIG.
Severity: 15 (Severe error)
Explanation: The value assigned the MSGLEVEL was too large. The directive is ignored and processing continues.
Response: Specify a correct MSGLEVEL. Resubmit the job for execution.

BFX030S NETEX ERROR: NRBSTAT=SSSS, NRBIND = III.
Severity: 12 (Severe error)
Explanation: NETEX has reported an error to the BFX program that is not an expected condition, "ssss" is the four digit status code returned by. NETEX; "iii" is the data or event indication type. The transfer of this file will be aborted. If batched execution is being used, BFX will attempt to transfer subsequent files.
Response: Refer to NETEX documentation to determine the cause of the error. Frequently this error may be caused by earlier, more comprehensible errors. If other BFX error messages precede this one, take the corrective action suggested by those messages.

BFX031I ALL FILE TRANSFERS HAVE BEEN PROCESSED.
Severity: 7 (Informational)
Module: BFXTI, BFXTR.
Explanation: BFX has encountered a .END directive or the end of the directive file.
Response: None.

BFX043S BFX PROTOCOL ERROR - PREMATURE DISCONNECT.
Severity: 12 (Severe error)

Explanation: The remote BFX program terminated the connection at a time that was not anticipated by the local BFX program.
Response: This is an internal BFX error. It should be brought to the attention of installation BFX support personnel.

BFX046S BFX PROTOCOL ERROR - DATA AFTER EOF.
Severity: 15 (Severe error)
Explanation: The remote BFX program sent data after sending a record with a record level of EOF.
Response: This is generally caused by user-written Record and/or Block Modules. Recode 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 input.
Response: None.

BFX048S JOB SUBMISSION FAILED. RC = eee.
Severity: 12 (Severe Error)
Explanation: The job file submission failed. The error is described by the system return code eee.
Response: Correct the job stream, resubmit job.

BFX049I JOB FILE NOT STARTED.
Severity: 7 (Informational)
Explanation: The job file received was not started due to errors previously encountered.
Response: Correct the previous error and resubmit job.

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

BFX051F Character mode record length llll exceeds 9999 characters.
Severity: 15 (Fatal error)
Explanation: When transferring records in character mode, the BFX limit is 9,999 characters.
Response: The file maybe transferred in binary mode.

BFX080I FILE ffffffffffff RECEIVED.
Severity: 7 (Informational)
Explanation: The file ffffffffffff was received. This message is generated if the receiving record module does not return a message on EOF.
Response: None.

BFX081F RECORD MODULE RETURNED A DATA RECORD DURING INITIALIZATION.
Severity: 15 (Fatal Error)
Explanation: The Record Module, during initialization, returned a data record. It is improper to do so at that time.
Response: This error is caused by a user-written Record Module. During initialization, (when the buffer level is -1) no data are to be returned to the Block Module. Recode the Record Module.

BFX082I FILE ffffffffffff SENT.
Severity: 7 (Informational)
Explanation: The file ffffffffffff was sent. This message is generated when the Record Module does not return a message on EOF.

Response: None.

BFX083S FILE ffffffffffff ABORT PROCESSED.

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

BFX085F MESSAGE IN DATA BLOCK.

Severity: 15 (Fatal Error)
Explanation: A message record was found inside a data block.
Response: This is caused by a user Block Module inserting a message record into a data block. Correct the Block Module and resubmit job.

BFX086S COMMUNICATIONS LOST.

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

BFX088S OFFER OF hhhhhhhh FAILED.

Severity: 12 (Severe Error)
Explanation: The NETEX offer of hhhhhhhh failed. The offer is then retried up to 5 more times.
Response: If the job aborted from this error, correct the problem by resolving the error that was previously logged to this message.

BFX089S CONNECT TO hhhhhhh FAILED.

Severity: 12 (Informational)
Explanation: BFX could not connect to host hhhhhhh. This is generally caused by job errors on the remote host.
Response: Check the job and try again.

BFX090S Connecting host (hhhhhhhh) not equal HOST parm iiiiiii.

Severity: 12 (Informational)
Explanation: BFX was expecting HOST iiiiiii to connect. Host hhhhhhhh attempted the connection. The connection was terminated. HOSTCHECK was a specified parameter in the BFX job. Insure two BFX jobs are not using the same offer ID.
Response: Correct the job and rerun.

BFX101I FILE ffffffffffff TRANSFERRED, nnnnnn RECORDS SENT.

Severity: 6 (Informational)
Explanation: The standard Sending Record Module has detected normal end of file on the input file ffffffffffff. The total number of logical records sent for this file is nnnnnn. At the time the message was issued, the last record- of the file will already have been sent to the receiving BFX.
Response: None.

BFX102S FILE ffffffffffff 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, BFX will attempt to transfer the rest of the specified files.
Response: Determine the cause of the I/O error. If the error can be corrected, rerun the BFX jobs.

BFX103S CANNOT FIND RECORD MODULE mmmmmmmm AND BLOCK MODULE bbbbbbbb COMBINATION.

Severity: 12 (Severe error)

Explanation: The Record Module specified by the RMOD= or the Block Module specified by the BMOD= parameter was not installed in this copy of the BFX program. Transfer of this file is aborted; if batched transfer of files is being performed, BFX will attempt to transfer the rest of the specified files.

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. BFX will stop the batched transfer of files.

Response: Correct the error encountered as described by the previous log messages. Rerun the BFX jobs.

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, BFX will attempt to transfer the subsequent files.

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, BFX will attempt to transfer the remaining files.

Response: Correct the error generated by or detected by the user-written module and resubmit 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, BFX will attempt to transfer the remaining files.

Response: Correct the error generated by or detected by the user-written module and resubmit 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, BFX will attempt to transfer the remaining files.

Response: Correct the error generated by or detected by the user-written module and resubmit the jobs.

BFX12IS BLOCK, RMAXL, OR JRMAXL TOO SMALL FOR LOGICAL RECORD SIZE.

Severity: 12 (Severe error)

Explanation: When the two BFX programs established a connection with one another, the BLOCK = parameter as determined by the user specified parameters was insufficient to hold the maximum logical record length specified by the other BFX.
Response: Adjust the BLOCK = parameter in one of the two BFX programs so it is sufficient to transfer the file.

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 transferred records are either missing or duplicated. This may be due to an internal BFX or NETEX integrity error, or due to a user-written Block Module that is incorrectly sending data to the standard Receiving Block Module.
Response: If the error was caused by a user-written Block Module, correct the coding error that caused incorrect data to be sent. If this is the standard BLOCK Module, bring the error to the immediate attention of Network Executive Software's support organization.

BFX124S MODE= PARAMETERS INCONSISTENT FOR BOTH BFX JOBS.
Severity: 12 (Severe error)
Explanation: In a BFX program pair, one side had MODE=BIT specified and the second had MODE-ASCII or MODE = FDATA. The transfer of the file is aborted; if more files remain to be transferred in a batched BFX run, BFX will attempt to transfer the subsequent files.
Response: Correct the erroneous specification and transfer the files that were not sent.

BFX201I FILE fffffffffff TRANSFERRED, nnnnnn RECORDS RECEIVED.
Severity: 7 (Informational)
Explanation: This message is normally issued when the receiving BFX processes the last record of the file. When issued, it indicates that the last record was received and the output file was successfully closed. File fffffffffff contains nnnnnn logical records.
Response: None

BFX204E FILE fffffffffff TRANSFER ABORTED, nnnnnn RECORDS RECEIVED.
Severity: 10 (Error)
Explanation: This message is issued by the receiving BFX code when the file transfer process is aborted either due to the loss of NETEX communication or some error detected by the sending BFX. fffffffffff contains the name of the input file; nnnnnn contains the number of records sent to the output file before the abort caused the transfer to stop. The transfer of this file is always aborted; subsequent files in a batched run will be transferred. The original error will be reported by other BFX messages.
Response: Correct the error that caused the abort in the first place. Transfer the file again.

BFX206E FILE fffffffffff TRANSFER ABORTED, nnnnnn RECORDS SENT.
Severity: 10 (Error)
Explanation: This message is issued by the sending BFX code when the file transfer process is aborted either due to the loss of NETEX communication or some error detected by the receiving BFX. fffffffffff contains the name of the input file; nnnnnn contains the number of records read from the input file before the abort caused transfer to stop. The transfer of this file is always aborted; subsequent files in a batched run will be transferred. The original error will be reported by other BFX messages.
Response: Correct the error that caused the abort in the first place. Transfer the file again.

BFX207F INVALID @RUN CARD - SUBMISSION ABORTED.
Severity: 15 (Error)
Explanation: If the received job file does not have a @RUN as the first received image, the submission is aborted.

Response: Send a valid @RUN as the first image of the job.

BFX208F JOB LINE 2 NOT @PASSWD - SUBMISSION ABORTED.

Severity: 15 (Error)

Explanation: When BFXJS used the RSI card reader submission method (RMOD = RSICARD) a ©PASSWD card is required immediately after the @RUN card. The 'userid' in the account/userid on the @RUN card must agree with the password provided on the @PASSWD card.

Response: Correct the @PASSWD card and resubmit the job.

BFX209I BFXJS SCHEDULED JOB & FOR EXECUTION.

Severity: 15 (Informational)

Explanation: When BFXJS used the RSI card reader submission method (RMOD = RSICARD) to submit a job.

Response: No action is necessary.

BFX210S CANNOT OPEN INPUT FILE fffffffffff. RC = rr.

Severity: 12 (Severe error)

Explanation: The sending BFX program was unable to open the input file whose name is specified by fffffffffff. The return code rr is in octal. The transfer of this file is aborted. BFX will attempt to transfer subsequent files in a batched run following this error.

Response: Correct the reason for the open failure. Transfer the failing files once again.

BFX211S CANNOT OPEN OUTPUT FILE fffffffffff. RC = rr.

Severity: 12 (Severe error)

Explanation: The receiving BFX program was unable to open the output file whose name is specified by fffffffffff. The return code for the open is given in octal. Transfer of this particular file is aborted. BFX will attempt to transfer subsequent files in a batched run following this error.

Response: Correct the reason for the open failure. Transfer the failing files once again.

BFX212E JOB ABORTED.

Severity: 12 (Severe error)

Explanation: The job being received encountered an error and the job was not submitted.

Response: A previous error message indicates the cause of the failure.

BFX220I SENDING [SECURED }FILE fffffffffff. [TO nnnnnnnnnn]

Severity: 4 (Informational)

Explanation: The sending BFX has successfully opened the input file and is ready to begin transfer of data. Transmission will begin as soon as this message is issued. The name of the file 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 fffffffffff.

Severity: 4 (Informational)

Explanation: The receiving BFX has successfully opened the output file and is ready to receive file data. The name of the file is “fffffffffff”.

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

Response: Correct the job and rerun.

BFX223F IBMTAP file & needs to be a tape.

Severity: 15 (Fatal)

Explanation: When transferring file fffffffff, using the IBMTAPE record module, the file must reside on tape.
Response: Correct the job and rerun.

BFX224F IBMTAP file ffffff*fffff 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 was set and MODE=EBCDIC, tape blocks would be limited to 9,999 characters..
Response: Correct the job and rerun.

BFX225F IBMTAP - State entry (&) is invalid.
Severity: 15 (Fatal)
Explanation: An unexpected error occurred when processing the tape.
Response: Notify 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. .
Response: Verify the tape is a properly labeled tape.

BFX227I IBMTAP - Tape vvvvvv read..Swapping to next tape.
Severity: 15 (Informational)
Explanation: BFX has completed reading the tape with volume-id vvvvvv. BFX is now switching to the next tap specified for a multi-volume transfer.
Response: Mount the next tape to allow the job to continue processing.

BFX228I IBMTAP - Tape vvvvvv completes file ffffff.
Severity: 15 (Informational)
Explanation: The end of file marks were encountered on the tape. The file transfer has been completed.
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 ssssssss.
Severity: 2 (Diagnostic)
Module: BFXSOF in BFXTI, BFXJS.
Explanation: The BFXJS Job Submitter or the BFXTI program has issued a NETEX SOFFER to wait for the corresponding program to connect to it. The name offered is "ssssssss", which will be the specified ID= of the user's input parameters.
Response: None.

BFX302I CONNECTING TO ssssssss ON HOST hhhhhhhh.
Severity: 2 (Diagnostic)
Explanation: When BFXTR is connecting back to its starting BFXTI, it has issued a NETEX SCONNECT to establish communications, "ssssssss" is the name to connect to as specified in the ID= or JID= user parameters; "hhhhhhhh" is the host name specified in the TO = or FROM = parameters.
Response: None.

BFX304I CONNECT COMPLETE nref=nnn.

Severity: 0 (Diagnostic)

Explanation: A previously issued NETEX SCONNECT has successfully completed. The nref is the internal NETEX connection id.

Response: None.

BFX307I OFFER COMPLETE nref=nnn.

Severity: 2 (Diagnostic)

Explanation: A previous NETEX SOFFER request has completed successfully. The nref is the internal NETEX connection id.

Response: None.

BFX308I CONFIRM ISSUED.

Severity: 2 (Diagnostic)

Explanation: A NETEX SCONFIRM is being issued in response to a previously completed SOFFER.

Response: None.

BFX309I CONFIRM COMPLETE nref=nnn

Severity: 2 (Diagnostic)

Explanation: A previously issued NETEX SCONFIRM has completed successfully. The nref is the internal NETEX connection id.

Response: None.

BFX310I CONNECT CONFIRM READ ISSUED.

Severity: 2 (Diagnostic)

Explanation: Following a successful SCONNECT request, the BFX program has issued an SREAD to obtain the SCONFIRM response from the other program.

Response: None.

BFX31II CONNECT CONFIRM COMPLETE nref=nnnn

Severity: 2 (Diagnostic)

Explanation: The SREAD issued to accept a SCONFIRM message from the remote BFX has successfully completed. The NETEX session establishment process is now complete. The nref is the internal NETEX connection id.

Response: None.

BFX312I BLOCK SIZE bbbb, DATAMODE dddd, LCM ll.

Severity: 2 (Diagnostic)

Explanation: This diagnostic message is issued by all BFX modules when the session negotiation process is complete, "bbbb" is the actual NETEX block size (in words) that will be used to transfer the file or job. "dddd" consists of four hexadecimal digits that give the NETEX DATAMODE to be used based on the requirements of the two BFX programs. "ll" is the Least Common Multiple size negotiated, which contains a value other than one when data are being sent to non-IBM processors with more than one character per word. ll is in words.

Response: None.

BFX313S OFFER OF ssssssss failed.

Severity: 12 (Severe error)

Explanation: The NETEX SOFFER of ssssssss failed. The offer is not retried.

Response: A NETEX-type error message will have preceded this message. Take action based on that error message and resubmit the job.

BFX314I DISCONNECT COMPLETED nref=nnn

Severity: 2 (Diagnostic)

Explanation: A previously issued NETEX SCONFIRM has completed successfully. The nref is the internal NETEX connection id.
Response: None.

BFX320F BAD CONNECT DATA RECEIVED.
Severity: 15 (Informational)
Explanation: BFX connect/confirm protocol from remote job was not correct.
Response: Trace data for reason. Check datamodes.

BFX325F CANNOT INITIALIZE THE CARD READER. RC = xx
Severity: 12 (Severe error)
Explanation: Error code from RSI open request.
Response: Check 2200 PRM for error reason.

BFX326S CANNOT TERMINATE CARD READER. RC = xx
Severity: 12 (Severe error)
Explanation: Error code from RSI term request.
Response: Check 2200 PRM for error reason.

BFX327F TRANSFER ABORTED, nn RECORDS RECEIVED. RC = xx
Severity: 12 (Severe error)
Explanation: Error code on record transfer.
Response: Check 2200 PRM for error reason.

BFX328I CARD READER SYMBIONT INITIALIZED.
Severity: 6 (Informational)
Explanation: RSI open successful.
Response: None

BFX329I RECEIVING JOB STREAM INTO INTERNAL CARD READER.
Severity: 4 (Informational)
Explanation: If BFXJS uses RSI card reader method of job submission, no job file is used. This message replaces the one indicating that the job file is being received.
Response: None

BFX330I JOB STREAM TRANSFERRED, nnn RECORDS RECEIVED.
Severity: 4 (Informational)
Explanation: When BFXJS used RSI card reader method of submission, this indicates end of job stream reception.
Response: None

BFX500F . INVALID HEADER RECEIVED
Severity: FATAL)
Explanation: Secure BFX received a packet that did not start with a valid header. The transfer is terminated, and the header received is displayed.
Response: Save the job output and contact technical support.

BFX501E nnnnnnnnnnnn.
Severity: Error

Explanation: Secure Transferred issued the call nnnnnnnn to the UNISYS 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.
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 SBFXJS ON REMOTE HOST”, “CHECK THAT COMAPI IS STARTED” and “RECEIVED CLOSE EVENT FROM REMOTE SIDE”.

BFX502E KEYIN (nnnnnnn) FAILED STATUS O-nnnnn FLAG O-nnn.
Severity: Error
Explanation: When the secure server (BFXSJS) attempted to register for the KEYIN nnnnnn, the registration failed with the following error codes. Refer to UNISYS EXECUTIVE REQUEST PROGRAMMING GUIDE for the KEYIN ER.
Response: Check the KEYIN nnnnnn is currently not in use, or contact Support

BFX503I OPERATOR ENTERED nnnnnnnnnnnnnn COMMAND.
Severity: Informational
Explanation: The operator issued a command request to BFXSJS. The command is logged.
Response: None

BFX504E KEYIN INPUT NOT RECOGNIZED
Severity: Error
Explanation: The operator entered a command to BFXSJS that was not valid.
Response: Correct the input. The commands are:
DEBUG OFF
DEBUG ON
TERM
ABORT

BFX505E INVALID LINE IN CONFIGFILE (KEY = VALUE) nnnnnnnnnnn
Severity: Error
Explanation: While processing the configfile, an invalid line was encountered. The format of the configuration file is KEY = VALUE Correct the configuration file and rerun the job.
Response: Correct the input. The commands are:

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

BFX507E UNKNOWN HOST nnnnn (CHECK DNS)
Severity: Error
Explanation: 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.
Response: If DNS is correctly configured, contact technical support.

BFX508E INVALID SEQUENCE RECEIVED EXPECTED: nn
Severity: Error

Explanation: The BFX transfer received a block of data with an incorrect sequence number. The transfer is aborted. The sequence number expected is display, and the BFX header is printed..
Response: Contact technical support.

BFX509I IP ADDRESS =
Severity: Informational
Explanation: Secure transfer received this/these IP addresses from DNS. It will use this addresses in the connection attempts..

BFX510I CONNECTED ON IP ADDRESS =
Severity: Informational
Explanation: A connection to the remote was successful using the listed IP address.
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..
Response: Correct errors previously listed and rerun the job.

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.

BFX911I END OF FILE nnnn: TRANSFERRED rrrr RECORDS.
Severity: 6 (Informational)
Explanation: nnnn is the file number; rrrr is the number of records.
Response: None.

BFX912S TRACKIO TO TAPE IS UNSUPPORTED.
Severity: 12 (Severe error)
Explanation: The TRACKIO option is for disc files only.
Response: Remove the TRACKIO option for tape files.

BFX913S OUTPUT MUST BE A LABELED TAPE.
Severity: 12 (Severe error)
Explanation: The output must be a labeled tape.
Response: Remove the LABELS option or use a labeled output tape.

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.

BFX915F RECORD MODULE OPTIONS REQUIRE BINARY MODE.
Severity: 12 (Severe error)
Explanation: Binary mode is required for record module options.
Response: Use MODE=BIT parameter.

BFX916F TRACKIO REQUIRES AT LEAST 2000 WORD RECORDS.
Severity: 12 (Severe error)
Explanation: At least 2000 word records are required for TRACKIO.
Response: Increase RMAXL to at least 2000.

BFX917E TAPE LABELING ERROR ee SUBSTATUS ss.
Severity: 12 (Severe error)

Explanation: Error code given to ER TLBLS by the exec.
Response: Check 2200 Programmer's Reference for TLBLS error code ee.

BFX918E TSWAPS ERROR ee.
Severity: 12 (Severe error)
Explanation: Error code given to ER TSWAPS by the exec.
Response: Check 2200 Programmer's Reference for TSWAPS error code ee.

BFX919W RECSIZE REQUIRED FOR OCTET READ. DEF:nnnn OCTETS.
Severity: 6 (Informational)
Explanation: OCTET mode read had a bad RPARM = RECSIZE = nnnn statement.
Response: Supply valid RPARM statement.

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.

BFX999I UNKNOWN MESSAGE. INDEX = xxx.
Severity: 15 (Informational)
Explanation: A message was not included in the message data base. Should never appear.
Response: Report to Networks Systems Technical Support.

Appendix D. 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 = YES in the SJS configuration file.

BFXSJS Log file (If a remote system is connecting to use)

- Before the job is submitted, enter using your BFXSJS keyin:
  - <*keyin*> switch (Start a new print cycle)
  - <*keyin*> debug on (turn on debugging information)
  - Run the job
  - <*keyin*> debug off (turn off debugging information)

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

Index

.

.END 25
.EOT 25

A

ARCHIVE 18
ASCII vii
asynchronous vii
Automatic Job Submission 7

B

BFX Execution Parameters 16
BFXJS 36
BFXSET 38
BFXSET Parameters 38
BLKMAX 39
BLOCK 19, 39
BMOD 19, 39
BPARM 19, 39
buffer vii

C

CHECKSUM 24
CHKSUM 39
CLOSE 19, 39
code conversion vii
configuration manager vii
CONNDELAY 19
CONNMAX 20
Control Statement Parameters 18
Control Statements 18

D

Data Modes 9
DTIME 40
DTIMES 40

F

FILE 20, 40
FILES 24
Five BFX Programs 2

H

header vii
HOBCHECK 20
HOBOPT 40
host vii
HOSTCHK 20, 40
HP NonStop 30

I

IBM 32
ID18
Installation Process 34
Internet Protocol (IP) vii
ISO vii

J

JBLOCK 40
JFILE 40
JFLAG 40
JID 20, 41
JOBFILE 20
JOBSUBMIT 18
JRMAXL 41

K

KEYVAL 21

L

LABELS 23, 24
Linux 31

M

Manual Job Submission 5
MODE 21, 41
MSGLV 41
MSGLVL 21

N

Network Configuration Table (NCT) vii
NEWHOST 22
NOCLOSE 19
NONSDF 22
NOSOE 23
NOSUBMIT 22
NOTIMESTAMP 25

O

Open Systems Interconnection (OSI) vii

P

path vii

R

RATE 22
RECEIVE 18
Remap DBanks 34, 35
Remote Job Submission 9
RESTORE 22

RMAXL 22, 41
RMOD 23, 41
RPARM 23, 24, 41

S

Sample Network Configuration 1
SEND 18, 41
SENDJB 41
SOE 23
SPERRY 23
Supported Configurations 1

T

TIMEFL 41
TIMEOF 24, 42
TIMEOU 24, 42
TIMESTAMP 25
TO 42
TRACKIO 23