H215L Bulk File Transfer (BFX™) Utility

for IBM z/OS Systems

Release 1.2

Software Reference Manual

MAN-REF-H215L-1.2-1

Revision Record

Revision*	Description
01 (9/2017)	Preliminary manual released
1.2 (1/2019)	Updates corresponding to release 1.2
1.2-1 (2/2019)	Remove erroneous subsystem terms for Secure NetEx

© 2019 by Network Executive Software. Reproduction is prohibited without prior permission of Network Executive Software. Printed in U.S.A. All rights reserved.

Preface

This manual describes the user interface to the Network Executive Software ("NESi") H215L Secure Bulk File Transfer (BFX™) utility for IBM z/OS systems. Secure BFX is used in conjunction with NESi's Secure NETwork EXecutive (NETEX®) family of software products (Secure NetEx/IP) for use on IP networks.

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

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

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

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

"Appendix A: User Modules" on page 71 describes how to write Secure BFX user modules. The user modules are called from user exits within Secure BFX. These user exits were carefully placed within Secure 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. Secure BFX Internal Summary" on page 89 describes the Secure BFX internal structure as it relates to writing user block (or record) modules.

"Appendix C. Secure BFX Error Messages" on page 103 contains Secure BFX error messages.

"Appendix D. H215L User Abend Codes" on page 123 lists H215L user abend codes.

"Appendix E. Secure BFX Condition Codes" on page 125 lists H215L condition codes.

Secure BFX uses Secure NetEx/IP, but the reader does not need to understand Secure NetEx/IP to use the first four sections of this manual and Secure BFX. However, users creating their own user modules (as described in "Appendix A: User Modules" on page 71) are assumed to be familiar with Secure NetEx/IP.

Reference Material

The following manuals contain related information.

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

Notice to the Reader

The material contained in this publication is for informational purposes only and is subject to change without notice. Network Executive Software is not responsible for the use of any product options or features not described in this publication, 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™
International Business Machines Corp.	IBM, z/OS, QSAM, z Series

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 website 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

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

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

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

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

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

ISO: Acronym for International Standards Organization.

link: (1) A joining of any kind of networks. (2) The communications facility used to interconnect two different networks. Open Systems Interconnection (OSI): A seven-layer protocol stack defining a model for communications among components (computers, devices, people, and et cetera) of a distributed network. OSI was defined by the ISO.

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

path: A route that can reach a specific host or group of devices or an order of searching for a dataset.

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

NetEx is a registered trademark of Network Executive Software.

path: A route that can reach a specific host or group of devices or an order of searching for a dataset.

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

Contents

Figures

Tables

Introduction

The Secure Bulk File Transfer (BFX™) utility is a software package to be used with NetEx Software's Secure NETwork EXecutive (Secure NetEx/IP) software. Secure BFX and Secure NetEx/IP provide the software to rapidly move large amounts of sequential file data between processors. The processors may use different operating systems or be of different manufacturer, provided they have an IP network and the proper Secure NetEx/IP products installed (H215L Secure BFX requires H214).

Secure BFX uses batch processing facilities to perform file transfers. As a result, it has all the options of a batch program to access data base systems or other media supported by the queued sequential access method (QSAM). Also, as a batch utility, Secure BFX can be run as a batch job, from a TSO terminal, or as an operator procedure.

Secure BFX can be used to securely transfer files between different hosts which may require specialized record or data conversion. Secure BFX allows the use of Secure NetEx/IP code conversion. For other code conversion, Secure 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 "User Modules" on page 63.

Supported Configurations

Secure BFX uses the Secure NetEx/IP 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

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

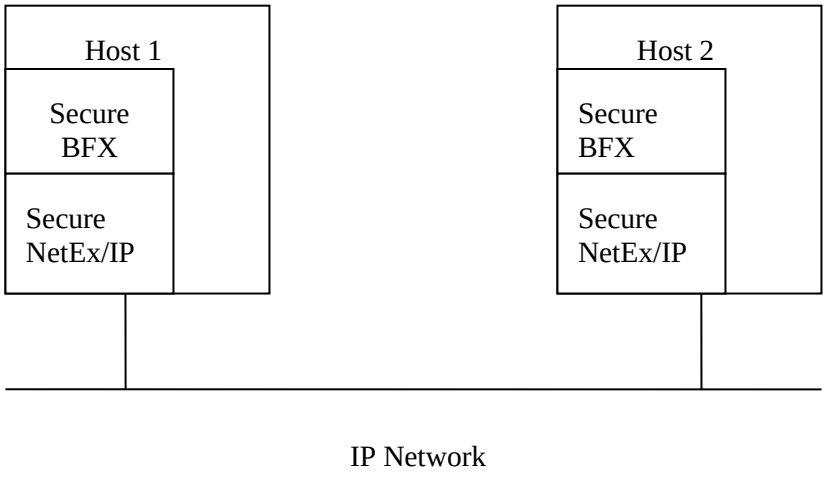

Figure 1 Sample Secure Netex/BFX network

Figure 1 is a simple example showing the following points:

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

Secure BFX Component Programs

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

The BFXTI program runs in the processor that initiates the transfer request. BFXTI processes 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 program.

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

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

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

Using Secure BFX

To use Secure 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 Secure BFX product used on that procedures host. The structure and contents of the initiating and responding procedures vary according to the application. There are three basic 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 Secure BFX. Manual job submission enables users in the initiating and responding systems to manually submit Secure 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 NOSUB 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 Secure NetEx/IP and the IP network) and the files are transferred.

The example procedures in Figure 2 and Figure 3 on page 4 and the following text and diagrams illustrate the manual job submission process. Assume that a user on the initiating host (ZOS1) wants to send two members of a partitioned data set (BMD and BMC) to the responding host (ZOS2), and receive member NSEFOI from the responding host using manual job submission. The user writes the JCL jobs shown in Figure 2 and Figure 3.

```
//INITIATE  JOB (ACCT),'EXAMPLE USING BFXTI',
//              CLASS=A,MSGLEVEL=(1,1),
//              MSGCLASS=V,NOTIFY=SAMPLE
//BFXTI    EXEC PGM=BFXTI
//STEPLIB    DD DSN=BFX.R10.BFXLOAD,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//IN01       DD DSN=SAMPLE.FTO.ASM(BMD),DISP=SHR
//OUT01      DD DSN=SAMPLE.FTO.ASM(MEMBER2),DISP=SHR
//IN02       DD DSN=SAMPLE.FTO.ASM(BMC),DISP=SHR
//SYSIN      DD *
  SEND    TO=ZOS2,ID=ACCT,MODE=CHAR,INDD=IN01,B=4200,NOSUB
  RECEIVE MODE=BIT,OUTDD=OUT01
  SEND    MODE=CHAR,INDD=IN02
/*
```

Figure 2. Host ZOS1 manual job submission for initiating BFXTI job

```
//RESPOND   JOB (ACCT),'EXAMPLE USING BFXTR',
//              CLASS=A,MSGLEVEL=(1,1),
//              MSGCLASS=V,NOTIFY=SAMPLE
//BFXTR    EXEC PGM=BFXTR
//STEPLIB    DD DSN=BFX.R10.BFXLOAD,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//OUT01      DD DSN=SAMPLE.FTO.ASM(MEMBER3),DISP=SHR
//IN01       DD DSN=SAMPLE.FTO.ASM(NSEF001),DISP=SHR
//OUT02      DD DSN=SAMPLE.FTO.ASM(MEMBER4),DISP=SHR
//SYSIN      DD *
  RECEIVE FROM=ZOS1,ID=ACCT,MODE=CHAR,OUTDD=OUT01,B=4200
  SEND    MODE=BIT,INDD=IN01
  RECEIVE MODE=CHAR,OUTDD=OUT02
//
```

Figure 3. Host ZOS2 manual job submission for responding BFXTR job

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

The user on ZOS1 runs the INITIATE job, which invokes the BFXTI program. Immediately after this, the user on ZOS2 runs the RESPOND job, which invokes the BFXTR program.

The z/OS systems begin processing the jobs. (All of the JCL statements are described in "JCL and Control Parameters for Secure BFX" on page 13.) On ZOS1, BFXTI is executed and the files to be sent (identified by the DD names IN01 and IN02) are allocated. Similarly on ZOS2, BFXTR is executed and the files to be received (identified by the DD names OUT01 and OUT02) are allocated.

The SEND statement in the initiating procedure specifies that a file is to be sent from ZOS1 to ZOS2. The SEND statement also provides additional parameters related to the file transfer. Unspecified parameters are assigned default values. In this example, the NOSUB parameter selects manual job submission.

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 INDD parameter. In this case, the IN01 DD statement specifies that the BMD member is to be sent. This member is sent to the RESPOND job, which receives it and calls it MEMBER3, as specified in the OUT01 DD statement.

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 RESPOND job then SENDs member NSEF001 over the network. BFXTI RECEIVEs it as MEMBER2 as specified by the OUT01 DD statement in the INITATE job.

The BFXTR and BFXTI programs continue processing. The INITIATE job SENDs member BMC over the network. BFXTR RECEIVEs it as MEMBER4 as specified by the OUT02 DD statement in the RESPOND 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.

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.
- This example shows BFXTI in host ZOS1 and BFXTR in host ZOS2. Normally, all three programs would exist in each host to allow either side to initiate a session.

Automatic Job Submission

Automatic job submission is the most commonly used method of transferring files using Secure 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. If the transfer is to be secure, the subsequent connection(s) opened on the SEND/RECEIVE is secure.

Secure Automatic Job Submission

If the job is to transfer securely, BFXTI opens a secure connection to BFXJS and sends the remote command set to the remote host. The local and remote command sets then transfer the specified files as they did for the manual job submission process. In this case the Authority file is checked to see that the submitting user/Secure NetEx/IP hostname is authorized to submit a job on this system. If the user is not authorized, the job is not submitted and an error returned to BFXTI. Otherwise, the job is run and if the transfer is to be secure, the subsequent connection(s) opened on the SEND/RECEIVE is secure.

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

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

Automatic Job Submission Sample

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

The sample initiating job (Figure 4 on page 7) and the following text and diagrams illustrate the automatic job submission process.

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

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

BFXTI retrieves the job control language for the RESPOND job from its internal file (RMTJOB). The BFXTI program then sends the RESPOND job to the BFXJS program on the remote host ZOS2.

The BFXJS program submits the RESPOND job that was sent by BFXTI for processing on the responding host. The RESPOND job calls the BFXTR program that is resident on the ZOS2 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 INDD parameter. In this case, the IN01 DD statement specifies that the BMD member is to be sent. This member is sent to the RESPOND job, which receives it as MEMBER3, as specified in the OUT01 DD statement in the RESPOND procedure.

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 BFXTR program then SENDs member NSEF001 over the network. BFXTI RECEIVEs it as MEMBER2, as specified by the OUT01 DD statement in the INITIATE procedure.

The BFXTR and BFXTI programs continue processing. The BFXTI program SENDs member BMC over the network. BFXTR RECEIVEs it as MEMBER4, as specified by the OUT02 DD statement in the RESPOND procedure.

BFXTR then terminates when it does not find any other BFX commands. BFXTI also terminates when it does not find any commands.

```
//INITIATE JOB (ACCT),'EXAMPLE USING BFXTI',
//             CLASS=A,MSGLEVEL=(1,1),
//             MSGCLASS=V,NOTIFY=SAMPLE
/* This procedure runs on host ZOS1
//BFXTI    EXEC PGM=BFXTI
//STEPLIB    DD DSN=BFX.R10.BFXLOAD,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//IN01       DD DSN=SAMPLE.FTO.ASM(BMD),DISP=SHR
//OUT01      DD BSN=SAMPLE.FTO.ASM(MEMBER2),DISP=SHR
//IN02       DD DSN=SAMPLE.FTO.ASM(BMC),DISP=SHR
//SYSIN      DD *
  SEND     TO=ZOS2,ID=ACCT,MODE=CHAR,INDD=IN01,B=4200,MS=4
  RECEIVE  MODE=BIT,OUTDD=OUT01
  SEND     MODE=CHAR,INDD=IN02
/*
//RMTJOB     DD DATA,DLM=ZZ
//RESPOND   JOB (ACCT),'EXAMPLE USING BFXTR',
//              CLASS=A,MSGLEVEL=(1,1),
//              MSGCLASS=V,NOTIFY=SAMPLE
/* This procedure runs on host ZOS2
//BFXTR    EXEC PGM=BFXTR
//STEPLIB    DD DSN=BFX.R10.BFXLOAD,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//OUT01      DD DSN=SAMPLE.FTO.ASM(MEMBER3),DISP=SHR
//IN01       DD DSN=SAMPLE.FTO.ASM(NSEF001),DISP=SHR
//OUT02      DD DSN=SAMPLE.FTO.ASM(MEMBER4),DISP=SHR
//SYSIN      DD *
  RECEIVE FROM=ZOS1,ID=ACCT,MODE=CHAR,OUTDD=OUT01,B=4200,MS=4
  SEND    MODE=BIT,INDD=IN01
  RECEIVE MODE=CHAR,OUTDD=OUT02
ZZ
//
```

Figure 4. BFXTI Automatic Job Submission

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 Secure 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:

```
//SAMPLE1  JOB (ACCT),'EXAMPLE USING BFXTI',
//             CLASS=A,MSGLEVEL=(1,1),
//             MSGCLASS=V,NOTIFY=&SYSUID
//BFXTI   EXEC PGM=BFXTI
//STEPLIB   DD DSN=BFX.R10.BFXLOAD,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  SUBMIT TO=ZOS2,ID=ACCT
/*
//RMTJOB    DD DATA,DLM=ZZ
//JOBNAME   JOB (ACCT),'REMOTE JOB',
//          CLASS=A,MSGLEVEL=(1,1),
//          MSGCLASS=A,NOTIFY=SAMPLE
//STEP1   EXEC PGM=IEFBR14
//IN      DD DSN=OLD.FILE(MEMBER),DISP=(OLD,DELETE)
//SYSPRINT DD   SYSOUT=*
//
ZZ
//
```

Figure 5. Remote Job Submission

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

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

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

When BFXTI finds a SUBMIT command, it establishes a Secure NetEx/IP connection with BFXJS on the remote host. BFXTI then transfers the remote job to BFXJS on ZOS2 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 Secure BFX.

Remote Hostname Substitution Table

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

When using this feature, the dataset name of the file is specified in the Secure BFX installation job. However it is not automatically created as part of the installation job. The customer must create the file and add hostname substitution entries to it prior to using it from Secure BFX jobs. This file can be a sequential file, or a member of a PDS. Any record size up to 1K can be used. The minimum size of each record must be sufficient to contain three table entries, with each entry being up to 8 characters. When Secure BFX jobs are subsequently executed, this file is dynamically allocated by dataset name when the BFX SEND, RECEIVE and SUBMIT control statements are processed. Since this dataset is dynamically allocated, using this feature does not require any changes to be made to BFX job streams.

An example of this file can be seen in Figure 6. Remote Hostname Substitution Table on page 9.

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

Figure 6. Remote Hostname Substitution Table

Data Modes

The Secure 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.

Secure BFX is designed to read a sequential file on the sending host, transfer it to the receiver, and write a sequential file on the receiving host. If the user requires non-sequential operations, encryption, or sophisticated data conversion, 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 Secure BFX, and converted back to non-sequential format.

Security

Secure BFX is an ordinary batch job to the host, however provides jobfile and user data transfers the option to have a private/secure tunnel. Other security mechanisms are not required because the existing host restrictions on file access, validation, and accounting will remain in effect.

File Size

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

Special Record Modules

There are two optional VSAM record modules for sending and receiving data in VSAM format. These two special record modules are installed automatically and are available at execution time by specifying RMODULE=BFXRVM or RMODULE=BFXSVM in the BFX control statements.

BFXRVM

BFXRVM is the receiving VSAM module. This record module receives keyed data from the remote BFX and writes it to an existing VSAM CLUSTER.

BFXRVM opens the VSAM file and determines its characteristics in order to update the file by the following steps.

1. LOAD new records if the data set is empty.
2. INSERT incoming record if a new key value

   or

3. UPDATE the record if the key previously exists.

The user can control INSERT or UPDATE processing times with the following optional execution parameters.

```
RPARM='OPT=SEQ' (Use Sequential Process.)
RPARM='OPT=SKP' (Use Skip Sequential Process.)
RPARM='OPT=DIR' (Use Direct Process - default.)
```

To invoke BFXRVM in the BFX execution parameters, see the following example.

```
Receive .... RMODULE=BFXRVM, RPARM='OPT=SEQ'
```

Upon completion of the file transfer, BFXRVM will issue the following statistics message.

```
MSG350I BFXRVM-Records Recv=nnnnnnn,Load=nnnnnnn,Updt=nnnnnnn,lsrt=nnnnnnn,Errors=nn
```

The statistics have the following meanings.

Recv Number of records received.

Load Number of records loaded in the file.

Updt Number of records updated in the file.

Isrt Number of records inserted in the file.

Errors Number of logical file errors encountered (maximum of 16).

Note: If logical file error(s) occur, message(s) will be issued to the system log reporting the FDBK value for the keyed record.

BFXSVM

BFXSVM is the sending VSAM module. This record module reads records from an existing VSAM cluster and sends them to the remote BFX. The optional execution parameters are as follows:

```
RPARM='GEN=kkkk....k' (Send all generic keys = kkkk...)
RPARM='KEQ=kkkk....k' (Send specific Keyed record.)
RPARM='KGE=kkkk....k' (Send all keys >= kkkk...)
(No RPARM parameter transfers the entire cluster.)
```

To invoke BFXSVM in the BFX execution parameters, see the following example.

```
SEND .... RMODULE=BFXSVM, RPARM='GEN=ABC'
```

Data Generating Modules

There are two data generating record modules - BFXDTG and BFXDTE.

The BFXDTG module generates data in records from 1 to 65535 bytes in length and transfers them to BFX. The BFXDTE module accepts the records without performing I/O functions to the file subsystem. These modules can be used to do data transfer timings between BFX's without the overhead of file I/O.

The RPARM parameter is used to identify the number of records and the size of records to send, as follows.

```
RPARM='nnnnnnnssssss'
```

where:

nnnnnnn Number of records to send.

sssssss Size of records to send.

For example, when invoking BFXDTG in the BFX execution parameters, the following execution parameters would send 500 65515-byte records.

```
SEND TO=NTXLCL,RMODULE=BFXDTG,RPARM='00005000065515',MODE=BIT,B=65535
```

When invoking BFXDTE in the BFX execution parameters, the following sample execution parameters would receive records up to 65535 bytes in size.

```
RECEIVE FROM=NTXLCL,RMODULE=BFXDTE,MODE=BIT,B=65535
```

JCL and Control Parameters for Secure BFX

This section describes the JCL 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.

JCL For Executing the BFXTI and BFXTR Programs

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

```
//jobname1 JOB  accounting information
//*  Execute BFX Transfer Initiate program
//        EXEC  PGM=BFXTI
//STEPLIB   DD  bfx load module library
//* Define output for error or status messages
//SYSPRINT  DD  SYSOUT=A
//* Optional file containing remote job control statements
//RMTJOB    DD  remote job file spec
//* Define file information for input or output files
//FILEIN    DD  input file spec
//*   or
//FILEOUT   DD  output file spec
//* Optional Remote Hostname Substitution trace messages
//STDOUT    DD  SYSOUT=A
//* Optional file of control statements//SYSIN     DD  control file spec

If the partner system is also a z/OS system, the contents of the RMTJOB file
would be:

//jobname2 JOB  accounting information
//        EXEC  PGM=BFXTR,PARM='control statement'
//STEPLIB   DD  bfx load module library
//SYSPRINT  DD  SYSOUT=A
//FILEIN    DD  input file spec
//* or
//FILEOUT   DD  output file spec
//* Optional Remote Hostname Substitution trace messages
//STDOUT    DD  SYSOUT=A
//* Optional file of control statements
//SYSIN     DD  control file spec
```

Figure 7. BFXTI / BFXTR example JCL

The following statements appear in Figure 7:

JOB

This parameter defines the jobname and accounting information for the BFXTI program.

EXEC PGM=BFXTI

This statement invokes the transfer initiate part of BFX. The program does not need to be authorized in any way. No parameters are required. However, if a single file is to be transferred, the single control statement may be supplied on a PARM statement. If a PARM statement is supplied, the SYSIN dataset will be ignored.

STEPLIB

This statement defines the BFX load module library.

SYSPRINT

This statement defines a file to be used for diagnostic and informational messages.

FILEIN or FILEOUT

These statements are used to specify the data file to be used for the BFX transfer if only one file is being transferred. Depending on the direction of file transfer specified in the input parameters, only one of the two DDNAMES will be used. FILEIN will only be opened for input (transfer from the local CPU to the remote CPU), and FILEOUT will only be opened for output (transfer from the remote CPU to the local CPU). The file is any sequential file that can be accessed using standard QSAM get/put macros. The maximum record size of the file can be any size up to and including the maximum size supported by Secure NetEx/IP (a NETEX initialization parameter.)

The user may change the DDNAMES of the input or output file through the use of the Secure BFX control parameters. This facility is particularly useful when several files are to be transferred with a single execution of Secure BFX.

SYSIN

This statement specifies execution parameters for the Secure BFX transfer process. The contents of this file and the available options for Secure BFX are described under “BFX Execution Parameters” later in this section. SYSIN parameters may optionally be supplied through the PARM field of the EXEC statement. If the PARM technique is used, a SYSIN file need not be provided.

If a SYSIN file is used for control parameter input and more than one record of input is provided, then Secure BFX will run in a batched mode of execution. The file transfer specified by the first control statement will run to completion. After it is complete, the second SYSIN statement for both the procedures using the BFXTI and BFXTR programs will be analyzed and processed. Clearly, both the procedures using the BFXTI and BFXTR programs must contain the same number of SYSIN statements and each must match the previous corresponding one in the other procedure. All parameters may be changed from one SYSIN statement to the next; the direction of transfer (SEND/RECEIVE) may be changed as well.

If a single file is to be transferred, the single control statement may be supplied as the PARM field (of the EXEC statement) for the procedure using the BFXTI or BFXTR program. If this control statement is supplied, the SYSIN field will be ignored.

RMTJOB

This statement specifies an input file that contains a complete job to be run on the remote CPU. It is the user’s responsibility to ensure that this data set contains all the valid JCL, JOB statements, passwords, and so on to run on the remote CPU. When the BFXTI program is invoked, this job data set will be sent to the internal reader of the remote CPU. The contents of the remote job input are discussed below.

//jobname2 JOB

This statement indicates a job card suitable for submission on the remote CPU. Note that since non-IBM Secure BFX implementations are available, it is possible that this remote job is not a z/OS system at all, but rather batch job statements for a completely different operating system

EXEC PGM=BFXTR

This statement invokes the BFX module that establishes communication with the original, initiating BFX procedure. Like BFXTI, this module need not be privileged and does not require PARM information. However, if a single file is to be transferred, the single control statement may be supplied on a PARM statement. If a PARM statement is supplied, the SYSIN dataset will be ignored.

SYSPRINT

This statement defines a file for the display of error and informational messages.

FILEIN or FILEOUT

These statements define the file to be used for sequential file input or output on the remote CPU. As in BFXTI, only one of these DD statements will be used by the Secure BFX program. FILEIN will only be opened for input when a file is to be transferred from the responding CPU to the initiating CPU; FILEOUT will only be opened for output when a file is to be transferred from the initiating CPU to the responding CPU.

The user may change the DDNAMES of the input or output file through the use of the Secure BFX control parameters. This facility is particularly useful when several files are to be transferred with a single execution of Secure BFX.

SYSIN

This statement defines the BFX execution parameters to be used by the procedure on the responding CPU. The format of these control statements is described below. It is the user's responsibility to ensure that the control statements passed to BFXTI and the statements passed to BFXTR are entirely consistent with one another.

If a single file is to be transferred, the single control statement may be supplied as the PARM field (of the EXEC statement) for the procedure using the BFXTI or BFXTR program. If this control statement is supplied, the SYSIN field will be ignored.

Secure BFX Execution Parameter Syntax and Placement

All input to Secure BFX is free-form input. The control statement (such as SEND) must precede all parameters but may follow any number of leading blanks. Keyword parameters may be separated by commas or spaces.

The Secure BFX parameter scanner treats equal signs, spaces, and commas as being equivalent delimiters between words. For example, the following two sets of parameters are equivalent:

```
S T=XYZ,ID=ABC

S,T XYZ ID ABC
```

Secure BFX Execution Parameters

The Secure BFX execution parameters consist of three control statements, SEND, RECEIVE, and SUBMIT, and a set of parameters. All three control statements use the same parameters, as shown in Figure 8 on page 17.

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

```
| SEND    |  | TO=destination host name,   |
| RECEIVE |  | FROM=originating host name, |
| SUBMIT  |
                ID=bfx identifier name
               [BLOCK=size of data block to send over network]
               [BMODULE=module to send/receive blocks]
               [BPARM=string for block module]
               [BUFFERS=number of block buffers]
               [BUSYMAX=No longer used
               [CONTRY=number of seconds between connect and/or offer attempts]
               [DISCMAX=number of seconds to wait for the remote program to be
                offered]
               [DSEC]
               [|HOSTCHK  |]
                 NOHOSTCH |
               [INDD=ddname of input file]
               [JBLOCK=job submission buffer size]
               [JBMODULE=block module for job submission]
               [JBPARM=string for sending block mod on j.s.]
               [JCLDD=ddname of remote job input]
               [JID=offered name of bfxjs job]
               [JMODULE=module to submit remote job]
               [JPARM=string for job module]
               [JRMODULE=record module for job submission]
               [JRPARM=string for sending record mod on j.s.]
               [JSAUTH]
               [JSEC]
               [MODE=| CHAR |]
                     | BIT  |
               [MSGLVL=severity of messages to be logged]
               [NODSEC]
               [NOJSAUTH]
               [NOJSEC]
               [NOSECURE]
               [NOSUB]
               [NTXID=NETEX subsystem name] [,NTXID=(ntxid1,ntxid2 ... ntxid8)]
               [OUTDD=ddname of output file]
               [RMODULE=module to read/write file records]
               [RMTJOB=jobname of job on remote host]
               [RPARM=| string for record module |]
                      | TRANSL                   |
               [SECURE]
               [SNXALL]|[SNXDBG]|[SNXERR]|[SNXGTRC]|[SNXINF]|[SNXLETRC]|[SNXOFF]|
                 [SNXTRC]
               [TIMEOUT=time in seconds for bfxtr to start]
               [-] the line continuation character
               [ | CLOSE   | ]
                 | NOCLOSE |
               [ | TIME    | ]
                 | NOTIME  |
               [ | UPPER | ]
                 | UPLOW |
```

Figure 8. Secure BFX Control Statements

Control Statements

SEND

This statement (abbreviated **S**) indicates that the file transfer will be from the local host to the remote host. A SEND statement implies that the FILEIN or INDD specified DDNAME will be used for input file I/O.

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

RECEIVE

This statement (abbreviated **R**) indicates that the file transfer will be from the remote host to the local host. A RECEIVE statement implies that the FILEOUT or OUTDD specified DDNAME will be used for output file I/O.

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.

SUBMIT

This statement (abbreviated **U**) indicates that a remote job is to be submitted for execution on the responding host. The remote job is included in the BFXTI execution statements where the BFXTR job is otherwise included. When the SUBMIT 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 SUBMIT statements use the following parameters (shown in Figure 8 on page 17). Defaults for many of these parameters are defined in the BFXDEF#xxxxx definitions in "Installing " on page 35.

ID

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

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

BLOCK

This optional parameter (abbreviated **B**) specifies the size of the buffers of data to be sent through Secure NetEx/IP to the other host. The size in BLOCK must be at least large enough to accommodate the largest logical record 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. This parameter can be specified as *nn*K (for example, `B=60K`). The largest BLOCK allowed is 65,535 bytes. BLOCK * BUFFERS typically cannot exceed 7M (depending on your OS).

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

BMODULE

This optional parameter (abbreviated **BM**) 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 Secure 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 71 for more information on user-provided block modules.

BPARM

This optional parameter (abbreviated **BP**) 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.

BUFFERS

This optional parameter (abbreviated **BF**) is used to specify the number of intermediate buffers between the file access and Secure NetEx/IP. Performance losses may result if a single buffer is allocated. Two buffers are normally allocated. Allocation of more than two buffers will generally have small effect unless I/O performance to Secure NetEx/IP or the file system is "bursty" in nature. BLOCK * BUFFERS typically cannot exceed 7M (depending on your OS).

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

BUSYMAX

No longer used.

CONTRY

This parameter (abbreviated **CON**) specifies the time interval between CONNECT attempts by BFXTI or BFXTR if the offering side is reported "busy" or "not present" by the remote Secure NetEx/IP. This parameter is also used as the time interval between OFFER attempts by BFXTI or BFXJS if the Secure NetEx/IP maximum number of sessions is exceeded. Default is 5 seconds.

DISCMAX

This parameter (abbreviated **DIS**) specifies the maximum actual time that the BFXTR job will wait for the BFXTI job to be offered on the remote host. The connecting program will retry every CONTRY seconds and will receive a not present indication a maximum of DISCMAX/CONTRY times before informing the user that the connection is not possible. If omitted, the default is 30 seconds.

Note: When the BFXTI job is referencing a tape dataset, or is started independently of the BFXTR job, this value must be set to a high value (for example, the value of 1800 or 30 minutes) to allow time for BFXTI to be offered.

DSEC

With this optional parameter user data transmissions will be encrypted.

HOSTCHK and NOHOSTCH

The optional HOSTCHK parameter (abbreviated HOS) 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 group host names are used
- If a local loopback host name is used (NTXLCLxx)

NOHOSTCH (abbreviated NOH) specifies that HOSTCHK will not be performed.

If neither parameter is supplied, an installation-defined default is used.

INDD

This optional parameter (abbreviated **I**) specifies an alternate DDNAME for the FILEIN DD statement.

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

JBLOCK

This optional parameter (abbreviated **JB**) specifies the block size to be used to move the job file over the NetEx/IP network. The JBLOCK parameter is valid only for BFXTI. This value will be adjusted upwards by the NESi record module if it is insufficient to contain a logical record in the file. The default is 800 bytes.

JBMODULE

This optional parameter (abbreviated **JBM**) specifies the block module to be used by BFXTI during job submission.

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

JBPARM

This parameter (abbreviated **JBP**) specifies a string of character parameters to be passed to the invoked sending block module during job submission. This string, if provided, may be up to 64 characters in length. By default, a string of all blanks is passed. The standard Secure BFX sending block module ignores the JBPARM.

This 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.

JCLDD

This optional parameter (abbreviated **J**) specifies an alternate DDNAME for the RMTJOB DD statement.

If this parameter is not supplied, the default is JCLDD=RMTJOB.

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).

JMODULE

This optional parameter (abbreviated **JM**) specifies the name of the user module (the job module) for submitting jobs. Many installations may already have mechanisms in place to submit jobs for execution on another host processor. In order for users to take advantage of this existing capability, Secure BFX internally calls a job module that accepts all responsibility for submitting a job on the remote host. The default Secure BFX job module reads a file from the RMTJOB DDNAME and sends it to the BFXJS program residing on the user-specified remote host. User-written JMODULES may obtain the text of the job from any source and submit it for execution by any means available.

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

See “User Modules” on page 63 for more information on user-provided job submission modules.

JPARM

This optional parameter (abbreviated **JP**) specifies a string of parameter information that will be passed to the specified job module for processing. User-written job modules may examine this string to control special processing options. The default Secure BFX job module ignores the JPARM 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.

JRMODULE

This parameter (abbreviated **JRM**) is the record module to be used by BFXTI during job submission to the remote host.

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

JRPARM

This optional parameter (abbreviated **JRP**) specifies a string of parameter information that will be passed to the sending record module during the job submission process. User-written modules may examine this string to control special processing options. 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.

The default Secure BFX job module examines this string for the parameter TRANSL (abbreviated **TR**) or TRSQ. If JRPARM = TR or JRPARM = TRANSL is specified, the string '(.' is translated to a left bracket ([), and the string '.)' is translated to a right bracket (]). For example, the string SAMPLE (.TRANSLATE.) would be translated to SAMPLE [TRANSLATE.] If JRPARM = TRSQ is specified, the left brace ({) is translated to a left bracket ([), and the right brace (}) is translated to a right bracket (]). For example, the string SAMPLE {TRANSLATE} would be translated to SAMPLE [TRANSLATE]. Other than these translation options, the default Secure BFX job record module ignores the JRPARM parameter.

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

JSAUTH

With this optional parameter Bfxjs will authorize the user and bfxti will send user information to job submission.

JSEC

With this optional parameter job transmissions will be encrypted.

MODE

This optional parameter (abbreviated **M**) is used to specify the nature of the information in the file. The CHAR parameter (abbreviated **C**) implies that the entire contents of the file will be translated to/from EBCDIC. The BIT parameter (abbreviated **B**) implies that the file contains at least some information that is binary in nature, such as binary integers, floating point numbers, packed decimal numbers, or others.

If the file being transferred is going to another IBM processor, then MODE=CHAR or MODE=BIT will usually result in exactly the same file being written on the receiving host. If the destination processor is not an IBM host, however, the processing is considerably different. If MODE=CHAR was specified, the data text will be converted to the character set of the receiving system when received. Also the logical records of character information will be converted to the corresponding logical record structure of the non-IBM host. 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.

Both sides must specify MODE=BIT or MODE=CHAR. If the two specifications are inconsistent, an error will occur.

If this parameter is not supplied, an installation-defined default is used (see "Installation" on page 37).

MSGLVL

This optional parameter (abbreviated **MS**) specifies the type of messages that the user wants to see in the SYSPRINT log 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 Secure BFX in detail, and are intended only for use in diagnosing problems with Secure BFX, Secure NetEx/IP, 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 transfer of the file 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.

NODSEC

With this optional parameter user data transmissions will not be encrypted.

NOJSAUTH

With this optional parameter Bfxjs will not authorize the user and bfxti will not send user information to job submission.

NOJSEC

With this optional parameter job transmissions will not be encrypted.

NOSECURE

With this optional parameter job transmissions will not be encrypted; for backward compatibility.

NOSUB

This optional parameter (abbreviated **NS**) specifies that the BFXTI job does not contain a BFXTR job for submission on the remote host. (NOSUB selects manual job submission as described in the introduction.) If NOSUB 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.

NTXID

This optional parameter is silently ignored in Secure NetEx/IP.

OUTDD

This optional parameter (abbreviated **O**) specifies an alternate DDNAME for the FILEOUT DD statement.

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

RMODULE

This optional parameter (abbreviated **RM**) specifies the name of the user 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 link-edited into the Secure BFX program.

If this parameter is not supplied, an installation-defined default is used. The Secure BFX default record module is designed to transfer character files between all types of host processors, and to move binary or structured information to another host without change on a logical record basis.

See “User Modules” on page 63 for more information on user-provided record modules.

RPARM

This optional parameter (abbreviated **RP**) 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 string parameter may be specified as a single keyword, or as a string delimited by apostrophes or quotation marks. If apostrophes are to be embedded, double apostrophes may be used.

The default Secure BFX record module examines this string for the parameter TRANSL (abbreviated TR) or TRSQ. If RPARM=TR or RPARM=TRANSL is specified, the string '(.' is translated to a left bracket ([), and the string '.)' is translated to a right bracket (]). For example, the string SAMPLE (.TRANSLATE.) would be translated to SAMPLE [TRANSLATE]. If RPARM= RSQ is specified, the left brace ({) is translated to a left bracket ([), and the right brace (}) is translated to a right bracket (]). For example, the string SAMPLE {TRANSLATE} would be translated to SAMPLE [TRANSLATE]. Other than these translation options, the default Secure BFX record module ignores the RPARM parameter.

If no parameter is supplied, an installation-defined default is used.

SECURE

With this optional parameter job transmissions will be encrypted; for backward compatibility.

SNXALL

With this optional parameter all message levels will be output by Secure NetEx.

SNXDBG

With this optional parameter all levels of debug will be output by Secure NetEx.

SNXTRC

With this optional parameter all NRBs, input, processing and completion status will be output by Secure NetEx.

SNXINF

With this optional parameter informational messages about the connection will be output by Secure NetEx.

SNXERR

With this optional parameter errors that occur during the transfer will be output by Secure NetEx.

SNXOFF

Default value. With this optional parameter no messages will be output by Secure NetEx.

SNXGTRC

This optional parameter will trace the gsk traffic (file gsk.<process>.ff.trace will be put in the z/OS UNIX /tmp directory).

SNXLETRC

With this optional parameter calls to LEC will be output by Secure NetEx.

TIMEOUT

This optional parameter (abbreviated **TM**) specifies the amount of time that the BFXTI job will wait for the BFXTR job to be connected to it. The value provided to TIMEOUT is a decimal integer specifying the number of seconds that the BFXTI program will SOFFER while waiting for the BFXTR program to connect back to it.

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

Note: When using the RMTJOB option with the JOB status (BFXJSTAT) function, the TIMEOUT value should be adjusted downward.

TO and **FROM**

These are parameters (abbreviated **T** and **F**) that specify the symbolic name of the Secure NetEx/IP host that is the other partner in the file transfer. Either **TO** (with a SEND or SUBMIT statement) or **FROM** (with a RECEIVE statement) must be specified.

There is no default for these parameters. One of these parameters is required for control statements supplied to both the BFXTI and BFXTR programs.

RMTJOB

This optional parameter (abbreviated **RJ**) specifies the job name of the remote BFXTR job and causes the BFXJSTAT program to be accessed if the TIMEOUT value occurs.

If this parameter is specified (for jobs on IBM z/OS hosts only), and if the remote BFXTR job does not connect back in the amount of time specified in the TIMEOUT value, then the BFXTI job will connect to the BFXJSTAT function on the remote host to determine whether the BFXTR job failed or is still awaiting execution. If it failed, it will print out a message received from the remote job status function on the remote host. If the job is still awaiting execution, then the BFXTI job will reoffer itself for the TIMEOUT value. This process will continue until the BFXTR job finally executes or leaves the input queues. See the comments on the RMTJOB parameter in "Job Status Function Request Component (BFXJSTAT)" on page 31.

Note that the remote job status function must be started by the user on the remote system in the same way that the BFXJS function is started (entering START BFXSTAT for example).

If this parameter is not supplied, this feature will not be used. Note that not all systems that use Secure BFX have the remote job status function.

- (line continuation character)

This character may be used in the SYSIN parameter file to indicate that more parameters for the same transfer are on the following line.

CLOSE and **NOCLOSE**

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

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

TIME and **NOTIME**

These optional parameters (abbreviated **TIM** and **NOT**) specify if a time of day timestamp is to precede all Secure BFX generated messages (TIME) or not (NOTIME).

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.

UPPER and **UPLOW**

These optional parameters (abbreviated **UPP** and **UPL**) specify if the messages that are to be placed in the SYSPRINT log file are to be printed in uppercase only (UPPER) or in mixed-case (UPLOW).

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

Special Considerations

Record Formats and Record Type Conversion

The Secure BFX default record modules use the QSAM or BSAM to access data on the external storage device. All file organizations and access methods supported by QSAM may be moved with Secure BFX. During the transfer process, logical records are read from the input file in accordance with the DCB parameters specified with the file or in the INDD DD statement. Secure BFX blocks these records for transfer using a variable length record format where all information in the input record is transferred to the other host. On the receiving side, these varying length records can be written using any record format on the other side. This technique has several implications:

- The file can be re-blocked during the transfer process.
- Conversion of file types from F-format to V-or U-format and vice versa is quite possible. F-format records can be expanded or contracted in length during the transfer process. Several rules are followed to adjust the length of records:

  If the incoming record is shorter than the LRECL for an F or FB format file, then the record is padded at the end to fill the record.

  If MODE=CHAR was specified on the receiving Secure BFX, the record is padded with blanks.

  If MODE=BIT was specified on the receiving Secure BFX, the record is padded with binary zeroes.
- If the incoming record is longer than the maximum LRECL on an F, V, or U format output file, then the record is truncated. The Secure BFX default record module will count the number of records truncated and print a warning message at the end of the file transfer.
- Secure BFX will support VS or VBS format records between hosts, and spanned records of arbitrarily large size may be copied by Secure BFX.
- If the receiving file is RECFM=VS or RECFM=VBS, then the JCL for the FILEOUT DD must specify this. If the JCL does not specify this, then the Receiving Record Module will terminate with the BFX213S message (for example, '//FILEOUT DD DSN=xx.xxx,DCB=RECFM=VS,DISP=OLD').
- If the sending file is in 'LRECL=X' format (variable records greater than 32760 bytes), then the JCL for the FILEIN DD must specify this. If the JCL does not specify this, then the Sending Record Module will terminate with the BFX214S message (for example, '//FILEIN DD DSN=xx.xx, DCB=LRECL=X, DISP=SHR).

Transfer to Non-IBM Systems

Secure BFX is designed to support the transfer of file data between incompatible systems in two ways:

- Files containing only character information (program source files, text, line printer output) will be converted to an immediately useful form when sent to a different processor.
- Files containing binary information, floating point numbers, or data structures will be sent to the other computer as a continuous string of bits on a logical record basis. Depending on the type of computer systems involved, the data may be ready for direct use, or some processing of the data may be needed following the Secure BFX run before it is ready for direct use.
- The remote job (RMTJOB) control statements included in the BFXTI program must be written in the control language that is used by the remote host. An example of this is given in "Applications" on page 33.

Job Status Function Request Component (BFXJSTAT)

The Secure BFX for z/OS) contains a remote job status function component. This function is designed to allow remote BFXTI programs to inquire about the status of jobs that have been submitted to a remote host.

Normally, BFXTI uses the value of the offer timeout parameter to determine how long to wait for the remote BFXTR job to connect back. If this time period expires before connection is made, BFXTI will abort the transfer process. This can be a problem when the batch queue on the remote host is heavily loaded or when the remote job is executing at a low priority.

The job status function allows BFXTI to determine whether the remote job is queued for execution, is currently executing, has finished executing, or never started (because of job control errors, and so on).

If the job is queued or is executing, BFXTI will reoffer itself with a chance that the offer will now succeed.

If the remote job fails for some reason, then the BFXTI job will not have to wait for its TIMEOUT value to occur. Instead it will print out a status message received from the remote job status function on the remote host.

The RMTJOB parameter, supplied with the SEND or RECEIVE command by the initiating procedure, keys the use of the remote job status function. This parameter identifies the job to be inquired upon. When RMTJOB is specified, BFXTI will offer with the specified TIMEOUT value. If the offer times out, BFXTI will issue a CONNECT to JOBSTAT for status about the job whose name is specified by RMTJOB. There are three possible outcomes of this request:

- The remote job has died or never was started. In this case, BFXTI will abort the transfer process.
- The remote job is queued for execution or is executing. BFXTI will repeat the offer for the TIMEOUT value as long as the BFXTR job is alive, but not yet connecting.
- An error occurs during communication with BFXJSTAT. (Probable cause: BFXJSTAT is not started on the remote host or does not exist.) In this case, BFXTI win reoffer itself once more using the TIMEOUT value, just as if RMTJOB had not been specified.

It is not a fatal error to use the RMTJOB parameter when the remote host Secure BFX does not have the remote job status function, but the BFXTI job will take longer to complete if the remote job does not start.

When the user knows that the remote job status function is available, the TIMEOUT value can be adjusted downward for the BFXTI job so that execution will terminate sooner if the remote BFXTR job failed to start.

Messages Issued by the Job Status Function

The messages returned by the job status function will be output to the user in error message BFX316. The messages contain a status byte (0-job not good, 1-job good, 2-could not connect), the remote job name (or application name if status code 2), the host name, and the message text.

Status messages/responses may be returned from any one of three levels in the query process:

1. Error response due to request syntax validation; message text is prefaced by ‘IEFSSREQ’ literal;
2. Error response due to request logic error; control block contents incorrect; message text is prefaced by ‘STATUS’ literal;
3. Status response detailing job execution; message text prefaced by ‘JOB’ literal.

If the status information returned is unrecognizable to program BFXJSSRQ, a message will be issued prefaced by ‘BFXJSSRQ’ literal.

IEFSSREQ Messages	Description
REQUEST TO SUBSYSTEM	Request sent to subsystem.
SUBSYSTEM ERROR	Subsystem process error.
FUNCTION NOT SUPPORTED	Request not supported.
SUBSYSTEM NOT ACTIVE	Subsystem not available.
SUBSYSTEM NONEXISTENT	Subsystem not defined.
DISASTROUS ERROR	Disastrous error.
LOGICAL ERROR (SSOB)	Invalid control block.

Status Messages	Description
JOB NAME NOT FOUND	Job name not in system.
INVALID JOBNAME/JOB ID	Job name syntax invalid.
NO CANCEL: DUPLICATE JOBS	Not applicable.
STATUS ARRAY TOO SMALL	Receiving array size invalid.
NO CANCEL: OUTPUT QUEUE	Not applicable.
INVALID SYNTAX	Request format error.
INVALID CANCEL REQUEST	Not applicable.

Job Messages	Description
CURRENTLY ACTIVE	Job running, Secure BFX will re-offer.
WAITING FOR EXECUTION	Job waiting; Secure BFX will re-offer.
QUEUED FOR OUTPUT	Job completed; possible error.
HELD IN CURRENT QUEUE	Queued; Secure BFX will re-offer.
2ND LEVEL MESSAGE	Multiple status responses.
ACTIVE IN NJE	Job in process of transfer between nodes.
STATUS UNKNOWN	JES returned unknown response.
NOT FOUND (JCL ERROR)	Job flushed; possible error.

BFXJSSRQ Message	Description
UNKNOWN STATUS	JES returned unknown response.

Secure BFX messages are described in “Appendix C. Secure BFX Error Messages” on page 103.

Applications

The following examples are provided in this section.

- IBM z/OS to HP NonStop
- IBM z/OS to Linux
- IBM z/OS to Unisys

IBM z/OS to HP NonStop

The example in Figure 9 shows the IBM z/OS job used to perform a Secure BFX transfer from an IBM initiating host to an HP NonStop remote host. The NetEx name of the z/OS 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.

```
//BFXTI1 JOB ,CLASS=A,MSGCLASS=A
//SEND    EXEC BFXTINOS,ID=I101
//F1       DD  DSN=BETATST.DATA101,DISP=SHR
//RMTJOB   DD  DATA,DLM=ZZ
:TRINFILE $AUDIT.BFXJS.TRINFILE
FILE $DSMSCM.BFXSAV.LOTESTX1
ID I101
TO ZOS1
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/
ZZ
//SYSIN    DD  *
SEND TO=TANDEMSE,ID=I101,MODE=B,MS=0,B=30000,NOCLOSE,TIME -
  INDD=F1,TIMEOUT=30
/*
//
```

Figure 9. Example z/OS initiated Secure BFX file transfer from z/OS to HP NonStop

IBM z/OS to Linux

The example in Figure 10 shows the IBM z/OS job used to perform a Secure BFX transfer from an IBM initiating host to a Linux remote host. The NetEx name of the z/OS 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.

```
//BFXTI2 JOB ,CLASS=A,MSGCLASS=Z
//SEND    EXEC PGM=BFXTI,
//      PARM='SEND TO=WASHC ID=A2 MODE=CHAR TIME MSGLVL=0 BLOCK=32000'
//STEPLIB  DD  DSN=BFX.BFX10.BFXLOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*,DCB=BUFNO=01
//SYSUDUMP DD  SYSOUT=*,DCB=BUFNO=01
//FILEIN   DD  DSN=BETATST.DATA5MR2,DISP=SHR
//RMTJOB   DD  DATA,DLM=ZZ
#nesi
#nesi
cd /home/nesi
rm DATA5MR2
/opt/bfx/bin/bfxtr<<EOF
receive from ZOS6 file DATA5MR2 id A2 mode char msglvl 0 timestamp block 32000
EOF
ZZ
/*
//
```

Figure 10. Example z/OS initiated Secure BFX file transfer from Linux to z/OS

IBM z/OS to Unisys

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

```
//BETAUS1  JOB ,CLASS=A,MSGCLASS=A,NOTIFY=&SYSUID
//*    Modify the denoted JCL statements below to meet
//*     your installation requirements.
//* @XQT    H305-R10*BFX.BFXTR
//* @XQT  BFX.BFXTR
//* @XQT  BFX121.BFXTR
//SEND    EXEC PGM=BFXTI,
// PARM='SEND TO=UNISYS    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
//FILEIN   DD  DSN=BETATST.UNISYS.DATA1,DISP=SH
//RMTJOB   DD  DATA,DLM=ZZ
@RUN  BFXIBM,0,NETEX
@ASG,T RECVFILE,F///99999
@XQT    BFX.BFXTR
RECEIVE
FROM = OS390H
FILE = RECVFILE
BLOCK = 2000
MODE = ASCII
ID = BFXU
TIMESTAMP
ZZ
/*
//
```

Figure 11. z/OS initiated Secure BFX file transfer from z/OS to Unisys

Installing Secure BFX

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

Prerequisites for Installation

H215L Secure BFX needs the following prerequisites for successful operation:

- An IBM z/OS system that is running the H214 Secure NetEx/IP software product
- At least one other host with its own Secure NetEx/IP and Secure BFX software.

Release Distribution

H215L Secure BFX for z/OS is distributed as a downloadable file.

Distribution Contents

The H215L distribution consists of one file (h215l.xmit). It is a TSO TRANSMITed file, which when received, results in creating a PDS library that contains two members (BFXINST and BFXL100), which are used to complete the H215L installation. The h215l.xmit distribution file can either be downloaded from Network Executive Software, or it can be copied from physical media, depending on the obtained distribution. To download the distribution file, contact support at support@netex.com for download instructions.

BFXL100

BFXL100 is a member in the PDS that is created when the H215l.xmit distribution file is processed with the TSO RECEIVE command. BFXL100 is another TSO TRANSMITed file that consists of all of the Secure BFX libraries necessary to complete the installation.

BFXINST

BFXINST is a member in the PDS that is created when the H215l.xmit distribution file is processed with the TSO RECEIVE command. BFXINST is a job that extracts the installation libraries from BFXL100, then performs the necessary steps to complete the installation..

Before Installing H215L

Before installing H215L Secure BFX fox z/OS, the H214 NetEx/IP installation must first be completed.

JES3 Considerations

When Secure BFX is used in JES3 environments, the installer must be aware that when setting up the defaults for BFXJS using the BFXJSD macro, JRMOD=JES3JS must be specified.

Installation Process

This section describes the installation procedure for the H215L Secure BFX Release 1.0 distribution.

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 Secure BFX distribution file

Step 2 Upload the distribution file to z/OS

Step 3 TSO RECEIVE the distribution file

Step 4 Execute the BFXINST job on z/OS

Step 5 Check for required updates

Step 6 Execute the BFXJS Program

Step 7 Verify the Secure BFX Installation

Step 8 (Optional) Execute the Remote Job Status Function

Step 9 (Optional) Establish Secure BFX Catalogued Procedures

Step 10 (Optional) Deploy Remote Hostname Substitution Table

Step 11 Customize Authorized Users/Remote Hosts - BFXAUTH File

Step 12 (Optional) Customize Secure BFX

Step 1. Obtain the Secure BFX distribution file

The distribution file 'h215l.xmit' can be downloaded from Network Executive Software. Contact support@netex.com to obtain the download instructions.

Step 2. Upload the distribution file to z/OS

FTP (binary mode) the h215l.xmit file to the z/OS system as follows:

1) Connect via FTP to your z/OS system.
2) Change the directory to your desired high level qualifier:

   cd 'high-level-qualifier'
3) If necessary, change the location of your local directory to the location of the distribution file:

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

   quote site lrecl=80 blksize=3120 recfm=fb prim=5000 sec=200 blocks
5) Transfer the distribution file in bin mode:

   bin

   put h215l.xmit distpkg.xmit
6) Quit your FTP client

Using the above names results in the distribution file residing on z/OS as the following file:

'high-level-qualifier.DISTPKG.XMIT'

Step 3. TSO RECEIVE the distribution file

Issue the TSO PROFILE PROMPT command to be sure prompting is allowed. Then issue the TSO RECEIVE command against the distribution file uploaded in "Upload the distribution file to z/OS" as follows:

RECEIVE INDSNAME('dsn')

where 'dsn' specifies the name of the distribution file that was FTP'd to z/OS.

(for example: RECEIVE INDSNAME('high-level-qualifier.DISTPKG.XMIT')

The RECEIVE command will issue the following prompt:

Enter restore parameters or 'DELETE' or 'END'

Reply with:

DSN('dsn1')

where 'dsn1' specifies the name of a PDS distribution library that will be created from 'dsn'.

(for example: DSN('high-level-qualifier.DFILE')

The resulting 'high-level-qualifier.DFILE' dataset is a PDS distribution library that contains two members:

BFXINST

BFXL100

BFXINST is a job that uses BFXL100 to create the H215L libraries and performs the installation.

The H215L installation scripted job (BFXINST) has been customized with the following installation options:

BLOCK(32767)

BUSYMAX(300)

TIMEOUT(300)

DISCMAX(300)

MSGLVL(4)

MODE(BIT)

JRMOD(BFXTSBRM)

Step 4. Execute the BFXINST job on z/OS

The BFXINST installation job consists of four phases:

LOAD Allocates the H215L base and update installation datasets and loads the datasets from the downloaded distribution.

LINK Links the H215L load modules.

EDIT Automatically tailors the H215L BFXJS startup proc and other Secure BFX JCL.

COPY Copies the tailored BFXJS startup proc to the specified PROCLIB library.

Tailor the installation job in BFXINST.

Warning: DO NOT ISSUE "CHANGE xxx ALL" commands against BFXINST. Change the keyword values on an individual basis only.

a) Change the following to your site requirements:

- The JOB card
- The unit name "UNIT=(SYSALLDA,,DEFER)" on the WORK DD card. Change SYSALLDA to a valid unit name matching your site requirements.

b) Review and tailor the installation parameters contained in BFXINST. At a minimum, the following parameters should be specified:

START(BEGIN)

STOP(END)

TYPE(BASE)

HLQ(hlqname)

DFILE(dfilename)

NTXHLQ(hlqname)

Customers should verify ALL parameters are set appropriately for their site. Tailoring site defaults can be accomplished by making the appropriate parameter changes to the BFXDEF#xxxx and BFXJSD#xxxx parameters prior to submitting the BFXINST installation job.

NOTE: the BFXINST installation job can be rerun as many times as necessary. If a future customer requirement makes it necessary to change an installation default after product installation, this can be accomplished by merely rerunning the BFXINST job with the new default change.

c) Submit the BFXINST job. This job will allocate and load the following H215L distribution libraries:

hlq.BFXCTL	Base control library
hlq.BFXMAC	Base macro library (limited distribution)
hlq.BFXSRC	Base source library (limited distribution)
hlq.BFXOBJ	Base object library
hlq.BFXSAMP	Base sample library
hlq.BFXLKED	Base linkedit statements library
hlq.BFXDLOAD	Base load library

hlq.BFXLOAD	Installation load library
hlq.AUTH.BFXLOAD	Installation authorized load library
hlq.PTFCTL	Update control library
hlq.PTFMAC	Update macro library (limited distribution)
hlq.PTFSRC	Update source library (limited distribution)
hlq.PTFMOD	Update module library
hlq.PTFSAMP	Update sample library
hlq.PTFLKED	Update linkedit statements library

BFXINST Installation Parameters (All Phases):

HELP Default value: NO

Allowed values: YES NO

The HELP parameter is used to produce a description of the H215L z/OS Secure BFX installation parameters and their usage. The value "YES" will only produce the HELP output, and no other installation job phases will be executed.

Example: HELP(YES)

START Default value: BEGIN

Allowed values: BEGIN LOAD LINK EDIT COPY

The START parameter is used to determine which phase the H215L installation job will be started at. It can be used in conjunction with the STOP parameter to cause only a portion of the installation job to be executed.

Example: START(LOAD)

STOP Default value: END

Allowed values: LOAD LINK EDIT COPY END

The STOP parameter is used to determine which phase the H215L installation job will be stopped at. It can be used in conjunction with the START parameter to cause only a portion of the installation job to be executed.

Example: STOP(COPY)

Example: to only execute the LINK phase, code the START and STOP parameters as:

START(LINK)

STOP(LINK)

TYPE Default value: BASE

Allowed values: BASE UPDATE

The TYPE parameter is used to provide the type of installation being performed.

BASE indicates a complete product release installation is being performed. New datasets are allocated, and the steps indicated by the START and STOP parameters are performed against those new datasets.

UPDATE indicates a product update is being performed. Datasets previously allocated during the BASE installation are used, and the steps indicated by the START and STOP parameters are performed against those datasets.

Example: TYPE(BASE)

HLQ Default value: BFX.DIST

Allowed values: Any valid data set name qualifier

The HLQ parameter is used to provide the high level qualifier of the data set names used by the installation job. By default, an HLQ of BFX.DIST is used, which results in all of the datasets identified in step c) on page 38 being created using names in the following format:

BFX.DIST.BFXCTL Base control library

Note: The data sets defined by the HLQ parameter will be deleted and recreated by the LOAD phase when performing a BASE installation.

Example: HLQ(SYSP.BFX)

COMPRESS Default value: NO

Allowed values: YES NO

The COMPRESS parameter is used to determine whether or not the IEBCOPY utility will be invoked to compress the H215L installation data sets before the data set is updated by the installation job. The COMPRESS parameter will only cause a compress of the H215L data sets. Other data sets used by the installation job will not be compressed. System data sets used by the COPY phase will not be compressed prior to the copies.

Example: COMPRESS(NO)

SYSOUT Default value: *

Allowed values: A-Z, 0-9, *

The SYSOUT parameter is used to provide the JES SYSOUT class for utility output (IEBCOPY, etc.). The value "*" will cause the installation job MSGCLASS SYSOUT class to be used for utility output.

Example: SYSOUT(A)

BFXINST Installation Parameters (Load Phase):

DFILE Default value: BFX.DIST.XMITS

Allowed values: Any valid fully qualified data set name

The DFILE parameter is used to provide the name of the dataset that was created when the FTP downloaded distribution file was TSO RECEIVE'd in “TSO RECEIVE the distribution file”

Example: DFILE(SYSP.PROD.DFILE)

VOLUME Default value: '' (none)

Allowed values: Any valid direct access (DASD) volume name

The VOLUME parameter is used to provide the z/OS volume name which will be used to allocate the H215L installation data sets.

Note: If an SMS policy is in effect for the HLQ name specified, the UNIT and VOLUME parameters can be omitted.

Example: VOLUME(PROD01)

UNIT Default value: '' (none)

Allowed values: Any valid direct access (DASD) unit name

The UNIT parameter is used to provide the z/OS unit name which will be used to allocate the H215L installation data sets.

Note: If an SMS policy is in effect for the HLQ name specified, the UNIT and VOLUME parameters can be omitted.

Example: UNIT(SYSDA)

BFXINST Installation Parameters (Link Phase):

ASSEMBLE Default value: ASMA90

Allowed values: Any valid z/OS program name

The ASSEMBLE parameter is used to provide the name of the IBM Assembler program at your site. This name is usually "ASMA90".

Example: ASSEMBLE(ASMA90)

If the Assembler does not reside in data set SYS1.LINKLIB at your site, the ASMLOAD parameter must be added to the installation job following the ASSEMBLE parameter. Its value should be the name of the data set at your site that contains the program specified in the ASSEMBLE parameter.

Example: ASMLOAD(SYSP.LINKLIB)

LINKEDIT Default value: IEWL

Allowed values: Any valid z/OS program name

The LINKEDIT parameter is used to provide the name of the IBM Linkage Editor or Binder program at your site. This name is usually "IEWL".

Example: LINKEDIT(IEWL)

If the Linkage Editor or Binder program does not reside in data set SYS1.LINKLIB at your site, the LINKLOAD parameter must be added to the installation job following the LINKEDIT parameter. Its value should be the name of the data set at your site that contains the program specified in the LINKEDIT parameter.

Example: LINKLOAD(SYSP.LINKLIB)

LOADLIB Default value: hlq.BFXLOAD

(BFX.DIST.BFXLOAD)

Allowed values: Any valid fully qualified z/OS dataset name

The LOADLIB parameter is used to provide the name of the load library where the BFX modules will be linked. This library must be an APF authorized library.

Example: LOADLIB(SYSP.PROD.BFX.LOAD)

AUTHLIB Default value: hlq.AUTH.BFXLOAD

(BFX.DIST.AUTH.BFXLOAD)

Allowed values: Any valid fully qualified z/OS dataset name

The AUTHLIB parameter is used to provide the name of the library where the BFXJSTAT (JOB STATUS) module will be linked. This library must be an APF authorized library.

Example: AUTHLIB(SYSP.PROD.BFX.AUTHLOAD)

NTXHLQ Default value: NETEX.H2140100

Allowed values: Any valid data set name qualifier

The NTXHLQ parameter is used to provide the high level qualifier of the NetEx/IP data set names used by the NetEx/IP installation job. By default, an NTXHLQ of NETEX.H2140100is used, which results in the following Secure NetEx/IP datasets being used in the BFX installation:

NETEX.H2140100.SNXMAC	NetEx/IP macro library
NETEX.H2140100.SNXOBJ	NetEx/IP object library
NETEX.H2140100.SNXLOAD	NetEx/IP load library

Example: NTXHLQ(SYSP.NETEX)

NEWDFLTS Default value: NO

Allowed values: YES NO

The NEWDFLTS parameter is used to specify whether or not new defaults should be built (assembled) during an UPDATE installation.

If NO is specified (or defaulted), the current BFXADEF and BFXAJSD modules will be preserved with the UPDATE. This means all of the parameters that were specified (or defaulted) during the BASE installation (or subsequent UPDATEs) will remain intact.

If YES is specified, new BFXADEF and BFXAJSD modules will be assembled, using the BFXDEF# and BFXJSD# parameters specified (or defaulted) in the BFXINST installation job. If NEWDFLTS(YES) is specified, it is the customer's responsibility to ensure the BFXDEF# and BFXJSD# parameters are specified as desired.

Note: If new BFXADEF and/or BFXAJSD source is distributed as part of an UPDATE, NEWDFLTS(YES) will be forced.

BFXDEF Customized Default Parameters (Link Phase):

The following group of parameters can be used to customize Installation Defaults that will be used when running Secure BFX jobs. However, these parameters can also be overridden at run time. Review these parameters and customize values to your site requirements. In many cases, using the defaults will be sufficient.

Formatting rules when specifying the BFXDEF#xxxxx parameters:

1. A null (or no) parameter cannot be specified as (). This will cause a "Missing keyword value" message to be generated and the installation script will terminate.

   Example: BFXDEF#BLOCK() <= causes the installation to fail

2. Specifying a value of (‘’) will result in the default value being used for the keyword.

   Example: BFXDEF#BUFFERS(‘’) <= results in BUFFERS(2) (default)

3. If a character string is entered as a parameter, and it contains blanks, it must be enclosed in ‘’.

   Example: BFXDEF#BPARM(‘PARM1 PARM2 PARM3’)

BFXDEF#BLOCK

Default value: 8192

Allowed values: up to 65535 (or 64K)

This parameter specifies the default block size to be used to move file data over the NetEx/IP network. This value will be adjusted upwards by the Secure BFX default record module if it is insufficient to contain a logical record in the file. This value can be specified as BLOCK= nnnnn where nnnnn is the number of bytes, or as BLOCK=nnK where nn*1024 byte blocks are selected. The default is 8Kbytes.

Example: BFXDEF#BLOCK(16000)

BFXDEF#BMOD

Default value: STANDARD

Allowed values: any name, up to 8 characters in length

This parameter specifies the alphabetic name of the default block module to be used for transferring the file. The default is BMOD=STANDARD, which is a symbolic name that calls the RBM or SBM Secure BFX default block modules through the Secure BFX MDT table.

Example: BFXDEF#BMOD(STD4)

BFXDEF#BPARM

Default value: "" (none)

Allowed values: any character string, up to 64 bytes in length

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 BMOD. The standard Secure BFX block modules ignore the BPARM. The string must be enclosed in apostrophes.

Example: BFXDEF#BPARM('ABCD EFGH')

BFXDEF#BUFFERS

Default value: 2

Allowed values: 1-255

This parameter specifies the default number of buffers to be used in staging data between the block module and Secure NetEx/IP. The default is 2 buffers.

Example: BFXDEF#BUFFERS(5)

BFXDEF#BUSYMAX

Default value: 300

Allowed values: 1-32000

No longer used. If omitted, the default is 300 seconds (5 minutes).

Example: BFXDEF#BUSYMAX(180)

BFXDEF#CLOSE

Default value: NO

Allowed values: NO YES

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 Secure BFX components to be communicated with (BFXTI, BFXTR, BFXJS) must also use SCLOSE. It is recommended to use SCLOSE if all Secure BFXs support it.

Example: BFXDEF#CLOSE(YES)

BFXDEF#CONTRY

Default value: 5

Allowed values: 1-3600

This optional parameter (abbreviated CON) specifies the time interval (in seconds) between CONNECT attempts by BFXTI or BFXTR if the offering side is reported “busy” or "not present” by the remote Secure NetEx/IP. This parameter is also used as the time interval between OFFER attempts by BFXTI or BFXJS if MAX SESSIONS EXCEEDED. If omitted, the default is 5 seconds.

Example: BFXDEF#CONTRY(60)

BFXDEF#CTLDD

Default value: SYSIN

Allowed values: any valid DD name, up to 8 characters in length

This parameter specifies the DDNAME of the file to be used for the input file containing the Secure BFX control statements. The default is CLTDD = SYSIN.

Example: BFXDEF#CTLDD(STDIN)

BFXDEF#DISCMAX

Default value: 30

Allowed values: 1-3600

This optional parameter (abbreviated DIS) specifies the maximum actual time (in seconds) that the BFXTI or BFXTR program will wait for the offering program to be offered on the remote computer. The connecting program will retry every CONTRY seconds and will receive a not present indication a maximum of DISCMAX/CONTRY times before informing the user that the connection attempt is not possible. If omitted, the default is 30 seconds.

Example: BFXDEF#DISCMAX(60)

BFXDEF#HOST

Default value: ‘’ (none)

Allowed values: any name, up to 8 characters in length

This parameter specifies a default TO or FROM host address to be used if this parameter is omitted by the Secure BFX user. This parameter is particularly useful in the environment where nearly all file transfers are between two processors. The BFXDEF macro does not supply a default HOST parameter. If the user fails to specify a HOST parameter and no default HOST exists, the run will terminate with an error.

Example: BFXDEF#HOST(ZOS5)

BFXDEF#HOSTCHK

Default value: NO

Allowed values: NO YES

This 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 group host names are used
- If a local loopback host name is used (NTXLCLxx)

Example: BFXDEF#HOSTCHK(YES)

BFXDEF#INDD

Default value: FILEIN

Allowed values: any valid DD name, up to 8 characters in length

This parameter specifies the default DDNAME to be used if an input file is needed to send a file to another processor. The default is INDD=FILEIN.

Example: BFXDEF#FILEIN(FILE2)

BFXDEF#JBLOCK

Default value: 800

Allowed values: up to 65535 (64K)

This parameter specifies a default block size to be used when sending the remote job to the remote BFXJS. The default is 800 bytes.

Example: BFXDEF#JBLOCK(12000)

BFXDEF#JBMOD

Default value: STANDARD

Allowed values: any name, up to 8 characters in length

This parameter specifies the alphabetic name of the default job submission block module to be used for transferring the file. The default is JBMOD=STANDARD, which is a symbolic name that calls the JS Secure BFX default job submission module through the BFXMDT table.

Example: BFXDEF#JBMOD(STD4)

BFXDEF#JBPARM

Default value: '' (none)

Allowed values: any character string, up to 64 bytes in length

This parameter (abbreviated JBP) specifies a default string of character parameters to be passed to the invoked sending block module during job submission. This string, if provided, may be up to 64 characters in length. By default, a string of all blanks is passed. The standard Secure BFX sending block module ignores the JBPARM.

Example: BFXDEF#JBPARM('ABCD EFGH')

BFXDEF#JCLDD

Default value: RMTJOB

Allowed values: any valid DD name, up to 8 characters in length

This parameter specifies the default DDNAME to be used for the input file containing the job to be submitted on the remote processor. The default is JCLDD=RMTJOB.

Example: BFXDEF#JCLDD(OUTJOB)

BFXDEF#JID

Default value: BFXJS

Allowed values: any name, up to 8 characters in length

This parameter specifies the process name that is passed to Secure NetEx/IP to establish a connection with the BFXJS module on the remote host. If omitted, the default is JID=BFXJS.

Example: BFXDEF#JID(BFXJST)

BFXDEF#JMOD

Default value: STANDARD

Allowed values: any name, up to 8 characters in length

This parameter specifies the alphabetic name of the default job submission module to be used for transferring the file. The default is JMOD=STANDARD, which is a symbolic name that calls the Secure BFX default job submission module through the MDT table.

Example: BFXDEF#JMOD(STD3)

BFXDEF#JPARM

Default value: ‘’ (none)

Allowed values: any character string, up to 64 bytes in length

This parameter specifies a default string of character parameters to be passed to the invoked job submission 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 Secure BFX job submission module ignores the JPARM. The string may be enclosed in apostrophes.

Example: BFXDEF#JPARM(‘ABCD EFGH’)

BFXDEF#JRMOD

Default value: STANDARD

Allowed values: any name, up to 8 characters in length

This parameter specifies the alphabetic name of the default job submission record module to be used for transferring the file. The default is JRMOD=STANDARD, which is a symbolic name that calls the Secure BFX default job submission module through the MDT table.

Example: BFXDEF#JRMOD(STD6)

BFXDEF#JRPARM

Default value: ‘’ (none)

Allowed values: any character string, up to 64 bytes in length

This parameter specifies a default string of character parameters to be passed to the invoked sending record module during job submission. This string, if provided, may be up to 64 characters in length. By default, a string of all blanks is passed.

The Secure BFX default job module examines this string for the parameter TRANSL (abbreviated TR) or TRSQ. If JRPARM=TR or JRPARM=TRANSL is specified, the string '(.' is translated to a left bracket ([), and the string '.)' is translated to a right bracket (]). For example, the string SAMPLE (.TRANSLATE.) would be translated to SAMPLE [TRANSLATE]. If JRPARM = TRSQ is specified, the left brace ({) is translated to a left bracket ([), and the right brace (}) is translated to a right bracket (]). For example, the string SAMPLE {TRANSLATE} would be translated to SAMPLE [TRANSLATE]. Other than these translation options, the default record module ignores the JRPARM parameter.

Example: BFXDEF#JRPARM(‘ABCD EFGH’)

BFXDEF#LOGDD

Default value: SYSPRINT

Allowed values: any valid DD name, up to 8 characters in length

This parameter specifies the DDNAME of the file to be used for message output by the program. The default is LOGDD=SYSPRINT.

Example: BFXDEF#LOGDD(LOG)

BFXDEF#MAPFN

Default value: ‘’ (none)

Allowed values: any valid dataset name, up to 44 characters in length

This optional parameter specifies the dataset name of the Remote Hostname Substitution Table. This dataset can either be a sequential dataset, or it can be specified as a member of a PDS. BFXTI and BFXTR will dynamically allocate this file and search for a matching Secure BFX application ID (or ID mask) when processing SEND and RECEIVE control statements. Refer to the “Remote Hostname Substitution Table” section on page 8 for a description of this feature.

Example: BFXDEF#MAPFN(BFX.BFXOMAP)

BFXDEF#MODE

Default value: CHAR

Allowed values: CHAR BIT

This parameter specifies if the default MODE for transferring files will be BIT or CHAR. If omitted in the BFXDEF macro, the default is MODE=CHAR.

Example: BFXDEF#MODE(BIT)

BFXDEF#MSGLVL

Default value: 8

Allowed values: 0-16

This parameter specifies the verbosity of the messages to be placed on the SYSPRINT log file. Accepted integer values are 0-16 (0 = most verbose). The default is MSGLVL= 8.

Example: BFXDEF#MSGLVL(6)

BFXDEF#NTXID

Default value: NETX

Allowed values: any 4-character name, beginning with NET

This is ignored in Secure NetEX/IP.

Example: BFXDEF#NTXID(NETZ)

BFXDEF#OUTDD

Default value: FILEOUT

Allowed values: any valid DD name, up to 8 characters in length

This parameter specifies the default DDNAME to be used if an output file is to be generated from the data sent by another processor. The default is OUTDD=FILEOUT.

Example: BFXDEF#OUTDD(FILE1)

BFXDEF#RATE

Default value: 0

Allowed values: 0-100000

This parameter specifies the transmission rate used when sending data, in kilobits per seconds. A value of zero means that no rate control is to be used.

Example: BFXDEF#RATE(10000)

BFXDEF#RMOD

Default value: STANDARD

Allowed values: any name, up to 8 characters in length

This parameter specifies the alphabetic name of the default record module to be used for transferring the file. The default is RMOD=STANDARD, which is a symbolic name that calls the RRM or SRM Secure BFX default record modules through the BFXMDT table.

Example: BFXDEF#RMOD(STD2)

BFXDEF#RPARM

Default value: ‘’ (none)

Allowed values: any character string, up to 64 bytes in length

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 string may be enclosed in apostrophes.

The Secure BFX default record module examines this string for the parameter TRANSL (abbreviated TR) or TRSQ. If RPARM = TR or RPARM = TRANSL is specified, the string '(.' is translated to a left bracket ([), and the string '.)' is translated to a right bracket (]). For example, the string SAMPLE (.TRANSLATE.) would be translated to SAMPLE [TRANSLATE]. If RPARM = TRSQ is specified, the left brace ({) is translated to a left bracket ([), and the right brace (}) is translated to a right bracket (]). For example, the string SAMPLE {TRANSLATE} would be translated to SAMPLE [TRANSLATE]. Other than these translation options, the default Secure BFX record module ignores the RPARM parameter.

Example: BFXDEF#RPARM(‘ABCD EFGHIJKLMNOP’)

BFXDEF#TIME

Default value: YES

Allowed values: YES NO

This parameter specifies if a time of day timestamp is to precede all Secure BFX generated messages. 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 omitted, the default is TIME=NO (no timestamp is printed).

Example: BFXDEF#TIME(NO)

BFXDEF#TIMEOUT

Default value: 1800

Allowed values: 1- 32000

This parameter specifies the maximum amount of time (in seconds) that a BFXTI job will wait for a BFXTR job to be started on the remote host. The default is TIMEOUT=1800 (30 minutes).

Example: BFXDEF#TIMEOUT(300)

BFXDEF#UPPER

Default value: NO

Allowed values: NO YES

This parameter specifies if the messages that are to be placed in the SYSPRINT log file are to be printed in uppercase only (YES) or in mixed uppercase and lowercase (NO).

If this parameter is not supplied, the default is NO.

Example: BFXDEF#UPPER(YES)

BFXDEF#MJSEC

Default value: YES

Allowed values: NO YES

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

Example: BFXDEF#MJSEC(NO)

BFXDEF#MDSEC

Default value: YES

Allowed values: NO YES

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

Example: BFXDEF#MDSEC(YES)

BFXDEF#JSEC

Default value: YES

Allowed values: NO YES

This parameter specifies if job submission to the remote JS will run in secure mode. If this parameter is not supplied, the default is YES.

Example: BFXDEF#JSEC(YES)

BFXDEF#DSEC

Default value: YES

Allowed values: NO YES

This parameter specifies if the job's data transfer will run in secure mode. If this parameter is not supplied, the default is YES.

Example: BFXDEF#DSEC(NO)

BFXDEF#NSHOST1

BFXDEF#NSHOST2

BFXDEF#NSHOST3

BFXDEF#NSHOST4

BFXDEF#NSHOST5

BFXDEF#NSHOST6

Default value: "(none)

Allowed values: any host name 8 bytes in length

These hosts are excluded from running in secure mode.

BFXDEF#SNXLOGL

Default value: 0

Allowed values: 0-255

This parameter specifies the log level to be set in Secure NetEx/IP.

The supported values (bit values) are:

DEBUG = 1

TRACE = 2

INFO = 4

ERROR = 8

GTRC = 32

LETRC = 64

These values may be added for multiple log levels.

Example: BFXDEF#SNXLOGL(0)

When setting the above parameters with Secure NetEx/IP debug settings, the result will be a logical OR'ing of the settings. Refer to the user manual for the appropriate Secure NetEx/IP product for details on additional debug settings.

BFXJSD Customized Default Parameters (Link Phase):

The following group of parameters can be used to customize Installation Defaults that will be used when running BFXJS jobs. However, these parameters can also be overridden at run time. . Review these parameters and customize values to your site requirements. In many cases, using the defaults will be sufficient.

Note: JES3 environments must code BFXJSD#JRMOD=JES3JS in order to override the standard BFXJS record module.

Formatting rules when specifying the BFXJSD#xxxxx parameters:

1. A null (or no) parameter cannot be specified as (). This will cause a "Missing keyword value" message to be generated and the installation script will terminate.

   Example: BFXDEF#BLOCK() <= causes the installation to fail

2. Specifying a value of ('') will result in the default value being used for the keyword.

   Example: BFXDEF#BUFFERS('') <= results in BUFFERS(2) (default)

3. If a character string is entered as a parameter, and it contains blanks, it must be enclosed in ''.

   Example: BFXDEF#BPARM('PARM1 PARM2 PARM3')

BFXJSD#BUFFERS

Default value: 2

Allowed values: 1-255

This parameter specifies the default number of buffers to be used in staging data between the block module and Secure NetEx/IP. The default is 2 buffers.

Example: BFXJSD#BUFFERS(5)

BFXJSD#BUSYMAX

Default value: 300

Allowed values: 1-32000

No longer used. If omitted, the default is 300 seconds (5 minutes)

Example: BFXJSD#BUSYMAX(120) .

BFXJSD#CLOSE

Default value: NO

Allowed values: NO YES

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

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

Example: BFXJSD#CLOSE(YES)

BFXJSD#CONTRY

Default value: 5

Allowed values: 1-3600

This parameter specifies the time interval between OFFER attempts by BFXJS if MAXSESSIONS is exceeded. The default is 5 seconds.

Example: BFXJSD#CONTRY(60)

BFXJSD#JBLOCK

Default value: 800

Allowed values: up to 65535 (64K)

This parameter specifies the default block size to be used to move the job file over the NetEx/IP network. This value will be adjusted upwards by the Secure BFX default record module if it is insufficient to contain a logical record in the file. The default is 800 bytes.

Example: BFXJSD#JBLOCK(8192)

BFXJSD#JBMOD

Default value: STANDARD

Allowed values: any name, up to 8 characters in length

This parameter specifies the alphabetic name of the default block module to be used for submitting the job. The default is JBMOD=STANDARD, which is a symbolic name that calls the RBM or SBM Secure BFX default block modules through the MDT table.

Example: BFXJSD#JBMOD(STD1)

BFXJSD#JBPARM

Default value: ‘’ (none)

Allowed values: any character string, up to 64 bytes in length

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 block module. The standard Secure BFX block modules ignore the JBPARM parameter.

Example: BFXJSD#JBPARM(‘ABCD EFGHIJKLMNOP’)

BFXJSD#JID

Default value: BFXJS

Allowed values: any name, up to 8 characters in length

This parameter specifies the name that BFXJS is to provide to Secure NetEx/IP to OFFER its job submission services. If omitted, the default is JID=BFXJS.

Example: BFXJSD#JID(BXJST)

BFXJSD#JRMOD

Default value: STANDARD

Allowed values: any character string, up to 8 bytes in length

This parameter specifies the alphabetic name of the default record module to be used for submitting the job. If used in a JES3 environment, specify JRMOD=JES3JS. The default is JRMOD=STANDARD, which is a symbolic name that calls the RRM or SRM Secure BFX default record modules through the MDT table.

Example: BFXJSD#JRMOD(STD1)

BFXJSD#JRPARM

Default value: ‘’ (none)

Allowed values: any character string, up to 64 bytes in length

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 record module. The standard Secure BFX record modules ignore the JRPARM parameter.

Example: BFXJSD#JRPARM(‘ABCD EFGH’)

BFXJSD#LOGDD

Default value: SYSPRINT

Allowed values: any valid DD name, up to 8 character in length

This parameter specifies the DDNAME on the BFXJS procedure or job for messages to be printed. If omitted, the default is LOGDD=SYSPRINT.

Example: BFXJSD#LOGDD(STDOUT)

BFXJSD#MSGLVL

Default value: 8

Allowed values: 0-16

This parameter specifies the verbosity of the messages to be placed on the SYSPRINT log file. Accepted integer values are 0-16 (0 = most verbose). The default is MSGLVL= 8.

Example: BFXJSD#MSGLVL(4)

BFXJSD#NTXID

Default value: NETX

Allowed values: any 4-character name, beginning with NET

This is ignored in Secure NetEx/IP.

Example: BFXJSD#NTXID(NETZ)

BFXJSD#OUTDD

Default value: INTRDR

Allowed values: Any valid DD name, up to 8 character in length

This parameter specifies the DDNAME to be used for BFXJS to write submitted jobs for subsequent execution. If omitted, the default is OUTDD=INTRDR.

Example: BFXJSD#OUTDD(RDRIN)

BFXJSD#TIME

Default value: YES

Allowed values: YES NO

This parameter specifies if the messages logged to the SYSPRINT file by BFXJS will contain a time of day timestamp in the form hh.mm.ss where hh=hours (24-hour clock), mm=minutes, and ss=seconds. If omitted, the default is TIME=YES (a timestamp will be generated).

Example: BFXJSD#TIME(NO)

BFXJSD#TIMEOUT

Default value: 0

Allowed values: 0-32000

This parameter specifies the number of seconds that the BFXJS job will wait for a remote job to arrive from a remote host. If a job does not arrive in that period of time, the BFXJS job will terminate. If 0 is specified, the Secure BFX job will never timeout. The default is TIMEOUT=0.

Example: BFXJSD#TIMEOUT(30)

BFXJSD#MJSEC

Default value: YES

Allowed values: NO YES

This parameter specifies if the user can lessen the secure mode of JS. If this parameter is not supplied, the default is YES.

Example: BFXJSD#MJSEC(NO)

BFXJSD#JSEC

Default value: YES

Allowed values: NO YES

This parameter specifies if JS will run in secure mode. If this parameter is not supplied, the default is YES.

Example: BFXJSD#JSEC(YES)

BFXJSD#MAPJFN

Default value: none

Allowed values: Any valid dataset name, up to 44 characters in length.

This parameter specifies the dataset name of the BFXJS Authorization File

Example: BFXJSD#MAPJFN()

BFXJSD#SOF1= through **SOF6=**

Default value: x

Allowed values: x

These parameters specify the text of a set of JCL statements to be executed before a remote job submitted from a host has begun. As many SOF keywords should be provided as are needed, up to a maximum of six. Statements should be enclosed in apostrophes.

Example: BFXJSD#SOF1('//SAMPJOB JOB CLASS=A')

BFXJSD#SOF2('//TEST EXEC PGM=IEFBR14')

BFXJSD#SOF3('//')

BFXJSD#EOF1 = through **EOF6 =**

Default value: x

Allowed values: x

These parameters specify the text of a set of JCL statements to be executed following the submission of a job from a remote host. As many EOF keywords should be provided as are needed, up to a maximum of 6. Statements should be enclosed in apostrophes (for example, EOF1 ='/*EOF').

Example: BFXJSD#EOF1(‘//SAMPJOB JOB CLASS=A’)

BFXJSD#EOF2(‘//TEST EXEC PGM=IEFBR14’)

BFXJSD#EOF3(‘//’)

BFXJSD#UPPER

Default value: NO

Allowed values: NO YES

This parameter specifies if the messages that are to be placed in the SYSPRINT log file are to be printed in uppercase only (YES) or in mixed uppercase and lowercase (NO). If this parameter is not supplied, the default is NO.

Example: BFXJSD#UPPER(YES)

BFXJSD#SNXLOGL

Default value: 0

Allowed values: 0-255

This parameter specifies the log level to be set in Secure NetEx/IP. The supported values (bit values) are:

DEBUG = 1

TRACE = 2

INFO = 4

ERROR = 8

GTRC = 32

LETRC = 64

These bit values may be added together for multiple log levels.

Example: BFXJSD#SNXLOGL(0)

When setting the above parameters with Secure NetEx/IP debug settings, the result will be a logical OR’ing of the settings. Refer to the user manual for the appropriate Secure NetEx/IP product for details on additional debug settings.

BFXINST Installation Parameters (Edit Phase):

BFXPROC Default value: BFXJS

Allowed values: Any valid JCL procedure name

The BFXPROC parameter is used to provide the name for the z/OS started task used to run BFXJS. This parameter will also be used as the BFXJS member name in the dataset defined by the PROCLIB parameter.

Example: BFXPROC(BFXJSP)

BFXSYSCL Default value: A

Allowed values: A-Z, 0-9

The BFXSYSCL parameter is used to provide the JES output class for SYSOUT datasets in the BFXJS procedure.

Example: BFXSYSCL(H)

BFXRDRCL Default value: A

Allowed values: A-Z, 0-9

The BFXRDRCL parameter is used to provide the JES output class for the INTRDR dataset in the BFXJS procedure.

Example: BFXRDRCL(S)

JSTPROC Default value: BFXJSTAT

Allowed values: Any valid JCL procedure name

The JSTPROC parameter is used to provide the name for the z/OS started task used to run BFXJSTAT. This parameter will also be used as the BFXJSTAT member name in the dataset defined by the PROCLIB parameter.

Example: BFXPROC(BFXJSTP)

JSTSYSCL Default value: A

Allowed values: A-Z, 0-9

The JSTSYSCL parameter is used to provide the JES output class for SYSOUT datasets in the BFXJSTAT procedure.

Example: JSTSYSCL(H)

SAMPJOBI Default value: SAMPJOBI

Allowed values: ‘’or any valid job name prefix or TSO user ID.

The SAMPJOBI parameter is used to provide the job name for the sample BFXTI job.

Example: SAMPJOBI(BFXTI)

SAMPJOBR Default value: SAMPJOBR

Allowed values: ‘’or any valid job name prefix or TSO user ID.

The SAMPJOBR parameter is used to provide the job name for the sample BFXTR job.

Example: SAMPJOBR(BFXTR)

SAMPJOBC Default value: A

Allowed values: A-Z, 0-9

The SAMPJOBC parameter is used to provide the job class for the BFX sample job.

Example: SAMPJOBC(A)

SAMPSYSC Default value: A

Allowed values: A-Z, 0-9, *

The SAMPSYSC parameter is used to provide the JES output class for SYSOUT datasets in the BFX sample job. The value "*" will cause the SAMPMSGC value to be used for the SYSOUT datasets in the BFX sample job.

Example: SAMPSYSC(H)

SAMPMSGC Default value: A

Allowed values: A-Z, 0-9

The SAMPMSGC parameter is used to provide the JES output class for the MSGCLASS parameter in the BFX sample job.

Example: SAMPMSGC(H)

SAMPACCT Default value: ‘’(none)

Allowed values: " or any valid job accounting information

The SAMPACCT parameter is used to provide job card accounting information for the BFX sample job.

Example: SAMPACCT(1234)

SAMPNTFY Default value: " (none)

Allowed values: " or any valid TSO user ID

The SAMPNTFY parameter is used to provide the TSO user ID that will be notified when the BFX sample job completes. To remove the NOTIFY keyword from the sample BFX job card, specify this parameter as:

SAMPNTFY(")

Example: SAMPNTFY(USER1)

BFXINST Installation Parameters (Copy Phase):

PROCLIB Default value: (None)

Allowed values: " or any valid fully qualified data set name

The PROCLIB parameter is used to provide the fully qualified name of the JCL procedure library data set that will contain the BFXJS startup procedure, as specified by the BFXPROC parameter. The EDIT phase creates the started task JCL procedure and the COPY phase copies it into your specified JCL procedure library for later activation. To bypass the copy of the JCL procedure, either do not execute the COPY phase of the installation job, or specify this parameter as:

PROCLIB('')

Example: PROCLIB(SYSP.BFX.PROCLIB)

REPLACE Default value: NO

Allowed values: YES NO

The REPLACE parameter is used to determine whether or not existing members in the dataset specified by PROCLIB will be replaced during the COPY phase.

Note: If the BFXJS proc name specified by BFXPROC already exists in the PROCLIB dataset specified by PROCLIB, be sure to specify REPLACE(YES).

Example: REPLACE(YES)

DISP Default value: OLD

Allowed values: OLD SHR

The DISP parameter is used to determine whether the dataset specified by PROCLIB will be allocated for exclusive use during the COPY phase. To ensure being able to successfully copy into the dataset specified by PROCLIB, specify this parameter as:

DISP(SHR)

Step 5. Check for required updates

Refer to the H215L Memo-to-Users for instructions on checking for product updates at www.netex.com.

Step 6. Execute the BFXJS Program

It is the responsibility of the installation systems programmer to provide the JCL needed to start and run the resident BFXJS program. Sample JCL to do this is shown in Figure 12 on page 62. This sample JCL is contained in the file 'hlq.BFXCTL', and is copied into the library specified by PROCLIB during the installation, if so specified.

The EXEC statement invokes the BFXJS program. BFXJS will remain in execution until cancelled. It should be given a scheduling priority roughly equal to "hot" batch programs. The Secure BFX program is not authorized in any way; when inactive, it will generally be swapped out by z/OS. If Secure NetEx/IP terminates,

BFXJS will remain in execution and issue a message to the user every 30 seconds until Secure NetEx/IP is started again.

```
//BFXJS   PROC SYSCL=A,
//             RDRCL=A
//*
//*    The STEPLIB library will be modified to match
//*    what was specified during installation.
//*
//*    Or, modify the JCL statements below to meet
//*    your installation requirements.
//*
//BFXJS   EXEC PGM=BFXJS,
//             TIME=1440,
//             PARM=''
//*   Sample PARM values and format
//*            PARM='ID=BFXJS NTXID=NETZ MSGLVL=0'
//*
//STEPLIB  DD  DSN=BFX.BFXLOAD,DISP=SHR
//*
//SYSPRINT DD  SYSOUT=&SYSCL
//INTRDR   DD  SYSOUT=(&RDRCL,INTRDR),
//             DCB=(RECFM=FB,BLKSIZE=800,LRECL=80)
```

Figure 12. Sample BFXJS JCL

The following statements appear in Figure 12:

EXEC

This statement invokes the BFXJS program. In normal use, the TIME parameter should be provided to allow Secure BFX to both execute and wait for an extended period of time (1440 allows Secure BFX to continue running until cancelled). Valid execution parameters are specified in "Secure BFX Execution Parameters" on page 21.

STEPLIB

This statement should point to the load module library “hlq.BFXLOAD” that is loaded during the installation process.

SYSPRINT

This statement writes a file of all the informational, error and end of transfer messages produced while Secure BFX is running.

INTRDR

This statement specifies the internal reader. DCB information should be provided with the DD statement.

Step 7. Verify the Secure BFX Installation

Member BFXSAMP in the ‘hlq.BFXSAMP’ dataset contains a sample BFX program to verify that Secure BFX was successfully installed. It consists of a z/OS batch job (BFXSAMPI) that submits a second embedded job (BFXSAMPR) through the BFXJS program on the local host.

Before the sample jobs are run, the JOB cards should be modified to meet local installation conventions. The STEPLIB cards must be changed to refer to the hlq.BFXLOAD load module library. In the FROM= and TO= fields specify the local host name.

If the program is successful, the lines of textual information provided as input to the BFXSAMPI job should be printed on the SYSPRINT file of the BFXSAMPR program. See Figure 13 for the sample BFXSAMPI job.

```
//BFXSAMPI JOB ,'BFX USER',
//            CLASS=A,MSGCLASS=A,REGION=0M
//*
//*    Change the host name (TO/FROM) to specify the
//*    name of a local host name.
//*
//*    Modify the JCL statements below to meet
//*    your installation requirements.
//*
//INIT    EXEC PGM=BFXTI,
//             PARM='SEND TO=<localhost>   ID=BFXSAMP'
//*
//STEPLIB  DD  DSN=BETATST.H215L100.BFXLOAD,DISP=SHR
//         DD  DSN=BETATST.H2140100.SNXLOAD,DISP=SHR
//*
//SYSPRINT DD  SYSOUT=A,DCB=BUFNO=01
//SYSUDUMP DD  SYSOUT=A,DCB=BUFNO=01
//*
//* Optional STDOUT to contain Remote Hostname Substitution Table trace messages
//*
//STDOUT   DD  SYSOUT=A
//FILEIN   DD  *,DCB=BLKSIZE=080
This is a group of card images to be transferred using
the NetEx BFX utility.
If successful, the text of this file should appear
on the 'SYSPRINT' file of the submitted 'BFXSAMPR' job.
/*
//RMTJOB   DD  DATA,DLM=ZZ
//BFXSAMPR JOB ,'BFX USER',
//            CLASS=A,MSGCLASS=A,REGION=0M
//RESPOND EXEC PGM=BFXTR,
//             PARM='RECEIVE FROM=<localhost>   ID=BFXSAMP'
//*
//STEPLIB  DD  DSN=BETATST.H215L100.BFXLOAD,DISP=SHR
//         DD  DSN=BETATST.H2140100.SNXLOAD,DISP=SHR
//*
//SYSPRINT DD  SYSOUT=A,DCB=BUFNO=01
//SYSUDUMP DD  SYSOUT=A,DCB=BUFNO=01
//*
//* Optional STDOUT to contain Remote Hostname Substitution Table trace messages
//*
//STDOUT   DD  SYSOUT=A
//FILEOUT  DD  SYSOUT=A,DCB=(RECFM=F,BLKSIZE=80)
//
ZZ
//
```

Figure 13. Sample Secure BFX verification job

Step 8. (Optional) Execute the Remote Job Status Function

It is the responsibility of the installation systems programmer to provide the JCL needed to start and run the remote job status function. Sample JCL needed to do this is shown in Figure 14. This JCL is included in the file ‘hlq.BFXCTL’ that was created during the installation process.

Note: the BFXJSTAT program must reside in an authorized library. It is the responsibility of the installation systems programmer to ensure this load library is authorized.

If your site uses the dynamic APF list format, the "hlq.AUTH.BFXLOAD" library can be authorized by issuing the following z/OS command:

```
SETPROG APF,ADD,DSNAME=hlq.AUTH.BFXLOAD,{SMS|VOLUME=volser}
```

Specify SMS if the volume is SMS-managed, or specify the volser information if the library is not on an SMS-managed volume.

The SETPROG technique only provides authorization until the next IPL. To make the authorization permanent, add the "hlq.AUTH.BFXLOAD' library to the 'PROGxx' PARMLIB member.

Refer to the IBM publication "*z/OS Initialization and Tuning Guide"* for complete descriptions of the methods used to authorize libraries.

```
//BFXJSTAT PROC SYSCL=A
//*
//*    NOTE: BFXJSTAT must reside in an Authorized Library
//*
//*    The STEPLIB library will be modified to match
//*    what was specified during installation.
//*
//*    Or, modify the JCL statements below to meet
//*     your installation requirements.
//*
//BFXJSTAT EXEC PGM=BFXJSTAT,
//             TIME=1440,
//             PARM='0,$$$$'
//*
//STEPLIB DD DSN=BFX.AUTH.BFXLOAD,DISP=SHR
//*
//SYSUDUMP DD  SYSOUT=&SYSCL
```

Figure 14. Sample BFXJSTAT JCL

Note: Two parameters are allowed to be passed in the form PARM ='timeout,ssn'. Subsystem name is ignored in Secure NetEx/IP. The default TIMEOUT value is zero.

The remote job status function may then be started by specifying START BFXSTAT from the operator's console.

In order to use the remote job status function, the remote job status function must be started on the remote host (where BFXTR will execute) and the RMTJOB parameter must be specified as part of the SEND/RECEIVE command issued in the BFXTI job. See "Job Status Function Request Component (BFXJSTAT)" on page 29 for an explanation of the BFXJSTAT function. BFXJSTAT can be stopped only by an operator cancel.

Step 9. (Optional) Establish Secure BFX Catalogued Procedures

Secure BFX is not distributed with any catalogued procedures to use Secure BFX services. Generally, installations will either want to invoke Secure BFX directly, or they will want to write easy to use procedures to perform very specific end user functions (such as transferring a specific type of file to a specific other host in the network).

Step 10. (Optional) Deploy Remote Hostname Substitution Table

The Remote Hostname Substitution Table can optionally be used to substitute remote hostnames that are specified in the BFX SEND and RECEIVE control statements with different remote hostnames that are defined in this table. The substitution is based on matching BFX application ID's (or application ID masks) that are specified in this table. Refer to the BFXDEF#MAPFN parameter in the BFXINST installation job for the specification of this table, and to the BFXOMAP sample member contained in the &hlq.DISTSAMP distribution library for the format of entries in this table.

Step 11. Customize Authorized Users/Remote Hosts - BFXAUTH File

The BFXAUTH file specifies what user(s) on remote hostname(s) can submit jobs to Secure BFX when secure job submission is performed. The authorization file is configured by the system administrator and secured by file access rights to the user that starts BFXJS. Each line of the file contains an initiating Secure NetEx/IP Hostname and OS specific UserID separated by whitespace. (All trailing characters following the UserID are ignored.) The file is processed from the beginning to the end looking for a match of the initiating Hostname and the UserID submitting the job. If the Hostname or UserID specified in the file is an asterisk (*), it will match any string. Once a match for the Hostname and UserID is met, no other lines are inspected in the file and the job will be submitted. If no match is met, the job will not be submitted and an error is returned to BFXTI. (The Authority file is only inspected for Secure Automatic Job submission.)

Refer to the BFXDEF#BFXJSAUTH parameter in the BFXINST installation job for the specification of this table, and to the BFXJMAP sample member contained in the &hlq.DISTSAMP distribution library for the format of entries in this table.

Step 12. (Optional) Customize Secure BFX

Secure BFX can be put to use by continuing with the installation. In addition, there is a generation facility that allows user-written block and record modules to be added by relinking the BFX programs.

The following paragraphs describe the BFXMOD macro used with this step. If no customization is required, the installation is complete.

Addition of Record and Block Modules

The Secure BFX program supports and uses the Secure BFX default block and record modules to copy sequential files. Secure BFX is designed to allow user-written modules to be added. In order for a user-written module to be added to Secure BFX, one or more entries must be added to the MDT Module Table that is used by all three of the Secure BFX programs. This table associates:

1. An alphabetic name of the module, one to eight characters in length.
2. The type of module, (receiving block, sending record, job submission, and so on).

3. The entry address of the module.
4. The source language of the module. If the module is written in the FORTRAN high level language, Secure BFX will establish the high level language run time environment before calling the module as a high level language subroutine.

Members BFXASM and BFXLINK, contained in "hlq.BFXCTL", can be modified as needed and used to perform the assemblies and links for the record and block modules.

BFXMOD Macro Parameters

In order to add a module, the BFXMOD macro must be coded. Its parameters are shown in Figure 15.

```
         | RBLK |
         | SBLK |                          | ASM  |
BFXMOD   | RREC | ,alpha name,entry addr | FORT |,[initep]
         | SREC |
         | JOBS |
```

Figure 15. BFXMOD Macro

The following parameters appear in the BFXMOD macro. BFXMOD is the keyword name of the macro.

RBLK, SBLK, RREC, SREC, or **JOBS**

These parameters are the five possible module types being generated with this BFXMOD macro. One of the five is selected. Note that the calling user does not distinguish between a send and a receive type module, but that the BFXMOD macro does make that distinction. Two modules may have the same alphabetic name even though their types are different. Thus a user module could be called CUSTMOD as both a send and receive record module; normally a different entry address would be called depending on whether the record module was receiving data or had the responsibility of sending data.

alpha-name

This parameter is the character name that the user will specify on the BMODULE, RMODULE, or JMODULE control parameters to invoke the desired module. This name may be any alphanumeric string 1 to 8 characters in length.

entry-addr

This parameter is the label of the entry point in the module. The macro will generate a V-type constant using this label and branch to the address placed there by the linkage editor.

ASM,FORT

These parameters are the two possible languages used to write the module. One of the two is selected. Secure BFX will call the run time library routine needed to establish the language environment for that routine. When the module is actually invoked, it is called as a subroutine in the high level language. If this parameter is omitted, the default is ASM (no special preparations for the call).

initep

This optional parameter specifies an initialization-required entry point (for example, VFEIN# for FORTRAN record modules).

Modifying the BFXMDT Module Table

To provide the module table, the user should edit the member BFXMDT in the file "hlq.BFXSRC", which was created as part of the installation process.

When edited, the file will take the form shown in Figure 16.

```
BFXMDBEG
BFXMOD  type,name,epaddr,language
.
.
.
BFXMDEND
END
```

Figure 16. BFXMDT Module Table

The BFXMDT statement begins by prefacing the module and generating the seven standard block and record modules (SBM, RBM, SRM, RRM, JES3JS, CMRJS, and SUB) in the module table. Following the BFXMDBEG statement will be zero or more BFXMOD macro entries, one for each type and name of module to be provided by the user or installation systems programmer. The entire block of input is terminated with a BFXMDEND macro and an assembler END statement.

After the BFXMDT module has been updated, then the LKEDBFX member in "hlq.BFXLKED" should be modified so that the user code will be incorporated into the Secure BFX modules. Refer to Figure 17 for the BFXLINK file. The user-written code may be added by adding a new DDNAME to the linkedit step and adding INCLUDE statements following each group of the existing INCLUDE statements in LKEDBFX.

```
//BFXLINK JOB ,BFX.LINK,CLASS=A,MSGCLASS=A,NOTIFY=BFXLINK
//*
//*        LINK-EDIT BFX CREATING SINGLE MODULE W/MULTIPLE EP'S
//*
//BFXLINK  PROC BFXOBJ=BFX.H215L100.BFXOBJ,
//         BFXLOAD=BFX.H215L100.BFXLOAD,
//         BFXLKED=BFX.H215L100.BFXLKED,
//         AUTHLOAD=BFX.H215L100.AUTH.BFXLOAD
//*
//LINKBFX  EXEC PGM=IEWL,PARM='XREF,LET,LIST,NCAL,SIZE=(768K,100K)'
//SYSPRINT DD  SYSOUT=*,DCB=BUFNO=01
//SYSUT1   DD  DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(CYL,(10,2))
//SYSLMOD  DD  DSN=&BFXLOAD,DISP=SHR
//BFXOBJ   DD  DSN=&BFXOBJ,DISP=SHR
//SYSLIN   DD  DSN=&BFXLKED(LKEDBFX),DISP=SHR
//*
//LINKJST EXEC PGM=IEWL,PARM='XREF,LET,LIST,NCAL,SIZE=(768K,100K)'
//*
//*        NOTE: BFXJSTAT MUST RESIDE IN AN AUTHORIZED LIBRARY
//*
//SYSPRINT DD  SYSOUT=*,DCB=BUFNO=01
//SYSUT1   DD  DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(CYL,(10,2))
//SYSLMOD  DD  DSN=&AUTHLOAD,DISP=SHR
//BFXOBJ   DD  DSN=&BFXOBJ,DISP=SHR
//SYSLIN   DD  DSN=&BFXLKED(LKEDJST),DISP=SHR
//         PEND
//*
//LINK   EXEC BFXLINK
//
```

Figure 17. Secure BFX Link JCL

Appendix A: User Modules

Secure BFX provides the capability to perform some record-type conversion and provisions for transfer to non-IBM systems. For applications where custom coding is required, Secure 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 want to write block modules under the following conditions.

- Character sets or assembly/disassembly modes are used which are not provided by Secure BFX.
- Users want to supply logical blocks.

User-written record modules should satisfy most other special needs not provided by Secure BFX.

It should be noted that the NESi block modules are highly specialized in that they call record modules. Most user-written block modules would not necessarily call record modules.

The rest of this section describes the NESi block and record modules. Appendix A, "Secure BFX Internal Summary" on page 91 provides Secure BFX internal information which may be helpful when writing block modules. The following paragraphs describe writing record, block, and job submission 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 Secure 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 18

```
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 18. FORTRAN Record Module Entry Parameters

FORTRAN Record Module Entry Parameters

The following parameters appear in Figure 18

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 Secure 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 Secure 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 re-

turning. 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 Secure 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 Secure 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 Secure NetEx/IP connection to the opposite Secure BFX program. If the level of the message is greater than or equal to the MSGLEV of the remote Secure BFX program, then it will be printed on the remote Secure 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 Secure 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 routines are entered by a CALL macro using the same argument list structure shown in the previous FORTRAN example. This means on entry that register 1 will point to a list of addresses containing the arguments of the FORTRAN program. The resulting parameter list is shown in Figure 19. The meanings of the parameters are described above in the FORTRAN entry section.

```
=====================================================================

  Map of parameter pointers passed to a Secure BFX record module from
  a Secure BFX standard block module.

---------------------------------------------------------------------
   0  (0)  ADDRESS    4  RBUF       Logical record location
---------------------------------------------------------------------
   4  (4)   SIGNED    4  RBUFLEN    Record length
---------------------------------------------------------------------
   8  (8)   SIGNED    4  RBUFLEV    Delimiter level
---------------------------------------------------------------------
  12  (C)  ADDRESS    4  RMSG       -> 128 byte area for message
---------------------------------------------------------------------
  16 (10)   SIGNED    4  RMSGLEN    Length of returned message .
---------------------------------------------------------------------
  20 (14)   SIGNED    4  RMSGLEV    Severity of returned message
---------------------------------------------------------------------
  24 (18)   SIGNED    4  RMODE      File MODE
           X'0'    0     MODEBIT    0 => Bit made
           X'1'    1     MODECHAR   1 => Character mode
---------------------------------------------------------------------
  28 (1C) CHARACTER   8  RDDNAME    DDNAME of file
---------------------------------------------------------------------
  36 (24) CHARACTER  64  RRPARM     64 bytes of RPARM text
=====================================================================

  Work area for use by the record module

---------------------------------------------------------------------
 100 (64)   SIGNED   72  RMSAVE     Save area for rtns called by Rmod
---------------------------------------------------------------------
 172 (AC)   SIGNED  128  RMDCB      Data Control Block space
---------------------------------------------------------------------
 300(12C)           180  -----      Reserved
---------------------------------------------------------------------
 480(1E0)   SIGNED    4  #RECS      Counter for # of records read
---------------------------------------------------------------------
 484(lE4)   SIGNED    4  #BYTES     Counter for # of bytes read
---------------------------------------------------------------------
 488(1E8)   SIGNED    4  #TRUNC     Counter of # records truncated
```

```
------------------------------------------------------------------------
 492(1EC)   SIGNED    2  MAXLRECL   Largest logical record in file
------------------------------------------------------------------------
 494(1EE)            54  -----      Used with spanned records
========================================================================
```

Figure 19. Assembler Record Module Parameters

Block and Record Module Exit and Entry Summary

Table 1 on page 75 through Table 4 on page 77 summarize the use of the BUFLEN, BUFLEV, MSGLEN, and MSGLEV parameters upon exit from the Secure 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. Block module exit parameters

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. Block module exit parameters

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. Block module exit parameters

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 Secure 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 Secure BFX. The module will not be called again.

Table 4. Block module exit parameters

Example of Sending Record Module (FORTRAN)

```
C
C  EXAMPLE OF USER-WRITTEN, SENDING.RECORD MODULE WHICH ACCESSES
C  A VSAM DIRECT FILE. WHEN 'SNDREC' IS CALLED, IT IS TO PROVIDE
C  'UNFORMATTED' RECORDS WHICH ARE 80 BYTES IN LENGTH. THE
C  'DDNAME' IS SUPPLIED BY THE CALLER. (THE STANDARD Secure BFX RECORD
C  MODULE WILL NOT READ VSAM FILES, SINCE IT IS WRITTEN IN ASSEMBLER
C  AND USES 'GETS' AND 'PUTS'.)
C
C  DATA STRUCTURES-PARMS
C
      SUBROUTINE SNDREC (BUF,BUFLEN,BUFLEV,MSG,MSGLEN,MSGLEV,
                         RPARM,MODE,DDNAME)
     +
      INTEGER*4 BUF(20)
      INTEGER*4 BUFLEN,BUFLEV,MSGLEN,MSGLEV,MODE
      CHARACTER*128 MSG
      CHARACTER*64 RPARM
      CHARACTER*8 DDNAME
C
C  RECORD COUNTER (NREC), IOSTAT INDICATOR (IOIND), ERROR MESSAGE
C  (EMSG), AND INTERNAL FILE FOR THE VSAM ERROR CODE (ECODE)
C
      INTEGER*4 NREC
      INTEGER*4 IOIND
      CHARACTER*32 EMSG
      CHARACTER*4 ECODE
C
C  ENTRY POINT:     BUFLEV = -1  -  FIRST CALL
C                   BUFLEV =  0  -  SUBSEQUENT CALLS
C                   BUFLEV = 16  -  FILE XFER ABORT
C
100   IF (BUFLEV .EQ. -1) GO TO 200
      IF (BUFLEV .EQ. 0) GO TO 300
      IF (BUFLEV .EQ. 16) GO TO 400
      GO TO 400
C
C  OPEN THE FILE ON INITIAL ENTRY
C
200   OPEN (11,FILE=DDNAME,ACCESS='DIRECT',FORM='UNFORMATTED',RECL=80)
C
C  THEN PROVIDE PARMS AND INITIAL MESSAGE TO CALLER. INITIALIZE NREC
C
      BUFLEN = 80
      BUFLEV = 1
      MSGLEN = 15
      MSGLEV = 5
      MSG = 'FILE WAS OPENED'
      NREC = 0
      RETURN
C
C  SUBSEQUENT CALLS
C
C  BUMP RECORD COUNTER
```

```
C
300   NREC = NREC+1
C
C  PROVIDE A RECORD OR CLOSE THE FILE
C
      READ (11,REC=NREC,ERR=400,IOSTAT=IOIND) BUF
C
C  TEST FOR EOF
C
      IF (IOIND LT. 0) GO TO 350
C
C  ELSE SEND RECORD AND RETURN USUAL RESPONSE TO CALLER
C
      BUFLEN = 80
      BUFLEV = I
      MSGLEN = 0
      MSGLEV = 0.
      RETURN
C
C  EOF ROUTINE. RETURN EOF RESPONSE, WITHOUT DATA, TO CALLER
C
350   BUFLEN = 0
      BUFLEV = 15
      MSGLEN = 11
      MSGLEV = 5
      MSG = 'EOF REACHED'
      CLOSE (11)
      RETURN
C
C  ABORT ROUTINE. RETURN ABORT RESPONSE
C
400   BUFLEN = 0
      BUFLEV = 16
      MSGLEN = 24
      MSGLEV = 15
C  SET UP MESSAGE
      EMSG = 'TRANSFER ABORTED. IOSTAT = `
      WRITE (ECODE,FMT=450) IOIND
450   FORMAT (14)
      EMSG(29:32) = ECODE(1:4)
      MSG = EMSG
C
      CLOSE (11)
      RETURN
C
C  END
C
500   STOP
      END
```

Example of Receiving Record Module (IBM Assembler)

```
*
*  Example of receiving record module, hence the name 'RCRECMOD'
*  which, in this case, receives a Unisys print record and
*  creates the equivalent IBM print record(s).
*
RCRECMOD CSECT
*
*  General housekeeping
*
*  Entry conditions:
*              R1:  Parm list (in RMPARM DSECT)
*
         SAVE  (14,12),,*    Save caller's regs
         LR    R12,RI5       Load entry address
         USING RCRECMOD,R12  Establish addressability
         LR    R11,RL        Copy address of parm list
         USING RNPARM,R11    Base it with R11
         LA    R2,RMSAVE     Load addr of savearea into R2
         ST    R13,4(,R2)    Save addr of caller's savearea at +4
         ST    R2,8(,R13)    Store addr of my savearea in caller's area
         LR    R13,R2        R13 now contains addr of savearea
*
* Check BUFLEV
*
         ICM   R1,B'1111',RBUFLEV     Get BUFLEV
         BNM   NEXT                   1st time (-1)?  Skip if not
*
*  BUFLEV = -1.  First time entry.
*
*  Insert the passed ddname into the DCB,
*  open the file,
*  send an 'open' message,
*  and return.
*
         LA    R10,PRINT               Set up R10
         MVC   40(8,RIO),RDDNAME       Insert the provided ddname
         OPEN  (PRINT,(OUTPUT))        Open the print file
         MVC   RBUFLEV,=F'134'         RETURN MAX LRECL IN BUFLEV
         MVC   RMSGLEN,=F'19'          Msg is 19 characters long
         MVC   RMSGLEV,=F'5'           Message level is '5'
         MVC   RMSG(15),OPMSG          Give msg to caller as expected
         B     EXIT
*
*  BUFLEV is not negative.  Subsequent entries (or problem).
*
NEXT     CLI   RBUFLEV+3,15   Check for normal data Xfer
         BL    WRITREC        Branch to write routine, if so
         BE    EOF            Branch to EOF routine, if so
*
* Else, fall through to the abort routine.
*
ABORT    MVC   RBUFLEV,=F'16' Indicate abort
         MVC   RMSGLEN,=F'22' Message is 22 characters in length
```

```
         MVC   RMSGLEV,=F'15' Message level is '15'
         MVC   RMSG(22),ABMSG Give message to caller as expected
         B     CLOSE
*
*  Process and write a record.
*        Get Unisys record, strip off control information,
*        and create IBM print record(s).
*        (It is assumed that each Unisys record is prefixed
*        with a 2-character header, which was inserted by the
*        sending record module. This header contains line
*        spacing-information as a signed, zoned number.
*        By previously agreed-upon convention, any negative
*        number means 'top of form'.)
*
WRITREC  MVC   SCOUNT(2),RBUF     Get record header
         PACK  SCOUNT(8),SCOUNT(2) Pack it
         CVB   R6,SCOUNT          Convert it to binary
         C     R6,=F'O'           Compare it with '01
         BL    TOP                If negative, branch to TOP
         BE    SUPPRESS           If zero, branch to SUPPRESS space
*
*  Move the record.
*
MOVEIT   MVC   OUTAREA+1(132),RBUF+2    Get the record
*
*  Write routine.
*
WRITE    PUT   PRINT,OUTAREA       Output the record
         MVC   OUTAREA,OUTAREA-1   Zero out OUTAREA
         BCT   R6,WRITE            Decrement and branch to print blank
*
*  Else, fall through to normal response.
*
         MVC   RBUFLEV,=F'1' Return normal response in BUFLEV
         SR    RO,RO         Zero out RO
         ST    RO,RMSGLEN    Return a '01 for MSGLEN
         ST    RO,RMSGLEV    Return a '01 for MSGLEV
         B     EXIT
*
*  EOF routine.  Return the proper stuff, then fall through
*                to the close routine.
*
EOF      MVC   RBUFLEV,=F'1'  Return normal response in BUFLEV
         MVC   RMSGLEN,=F'15' Message is 15 characters in length
         MVC   RMSGLEV,=F'5'  Message level is '5'
         MVC   RMSG(15),EDMSG Give message to caller as expected
*
*  Close routine.   Close the print file and then fall through
*                   to the exit routine.
*
CLOSE    CLOSE (PRINT)       Close the file
*
*  Exit routine.  Reverse housekeeping and return.
*
EXIT     L     R13,RMSAVE+4  Load R13 with addr of caller's savearea
         RETURN (14,12),,RC=O
*
```

```
* Routine for ejecting to top of page.
*
TOP     0I    OUTAREA,X'Fl' Insert a '1' for carriage control
        B     MOVEIT
*
* Routine for suppressing line spacing.
*
SUPPRESS 0I   OUTAREA,X'4E' Insert a '+' for no spacing
         B    MOVEIT

*
*  Define the print DCB.
*
PRINT    DCB   BLKSIZW=133,DSORG=PS,LRECL=133,MACRF=PM,RECFM=FA
*
*  Outarea and skip count.
*
         DC    C
OUTAREA  DS    CL133
*
SCOUNT   DS    D                   Skip count (for blank records)
*
* Messages
*
OPMSG    DC    C'THE FILE WAS OPENED'        CL19
*
EDMSG    DC    C'EOF WAS REACHED'            CL15
*
ABMSG    DC    C'THE FILE X-FER ABORTED'     CL22
*
* Equates
*
RO       EQU   0
R1       EQU   1
R2       EQU   2
R6       EQU   6
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*
*  Map of parameter pointers passed to the record module from
*  the standard Secure BFX block module.
*
RNPARM   DSECT
RBUF     DS    A                 Logical record location
RBUFLEN  DS    F                 Record length
RBUFLEV  DS    F                 Delimiter level
RMSG     DS    A                 -> 128 byte area for message
RMSGLEN  DS    F                 Length of returned message
RMSGLEV  DS    F                 Severity of returned message
RMODE    DS    F                 File MODE
MODEBIT  EQU   0                 0 => Bit mode
MODECHAR EQU   1                 1 => Character mode
RDDNAME  DS    CL8               DDNAME of file
```

```
RRPARM   DS    CL64               64 bytes of RPARM text
*
*  Work area for use by the record module
*
RMSAVE   DS    18F                Save area for rtns called by Rmod
*
         END   RCRECMOD
```

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 Secure NetEx/IP. 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 Secure NetEx/IP 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 Secure 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 Secure NetEx/IP supported DATAMODE may be used during the file transfer process.

IMPORTANT: When writing modules that call Secure NetEx/IP, it is important to clear the high byte of register 13 prior to calling Secure NetEx/IP. 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 20.

```
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 20. Block module entry

The parameters in Figure 20 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 Secure NetEx/IP 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 Secure 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 Secure NetEx/IP 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. Secure NetEx/IP 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 Secure NetEx/IP. 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 Secure 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 Secure 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 IBM block module will also begin on a multiple of LCM bytes.

If the system containing the other Secure BFX is an IBM 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 routines are entered by a CALL macro using the same argument list structure as when entering using FORTRAN. This means that on entry, register 1 will point to a list of addresses containing the arguments used. The resulting parameter list is shown in Figure 21. The first part of the figure shows the map of parameters passed to a Secure BFX block module from the BFX send or receive control module. The second part shows the work area used by the block module.

The meanings of the parameters in Figure 21 are described in the FORTRAN entry section.

```
=================================================================
  Map of parameter pointers passed to a Secure BFX block module
-----------------------------------------------------------------
   0  (0)  ADDRESS    4  BBUF       Address of block
-----------------------------------------------------------------
   4  (4)   SIGNED    4  BBUFLEN    Block length
-----------------------------------------------------------------
   8  (8)   SIGNED    4  BBUFUBIT   Unused bit count
-----------------------------------------------------------------
  12  (C)   SIGNED    4  BDTMODE    Datamode used for transfer
-----------------------------------------------------------------
  16 (10)   SIGNED    4  BBUFLEV    Delimiter level
-----------------------------------------------------------------
  20 (14)  ADDRESS    4  BMSG       -> 128 byte area for message
-----------------------------------------------------------------
  24 (18)   SIGNED    4  BMSGLEN    Length of returned message
-----------------------------------------------------------------
  28 (1C)   SIGNED    4  BMSGLEV    Severity of returned message
-----------------------------------------------------------------
  32 (20)   SIGNED    4  BLCM       Negotiated least common mult
-----------------------------------------------------------------
  36 (24)   SIGNED    4  BMODE      File MODE
-----------------------------------------------------------------
  40 (28) CHARACTER   8  BDDNAME    DDNAME of file
-----------------------------------------------------------------
  48 (30) CHARACTER  64  BBPARM     64 bytes of BPARM text
-----------------------------------------------------------------
 112 (70) CHARACTER   8  BRMOD      Alpha name of desired rec mod
-----------------------------------------------------------------
 120 (78) CHARACTER  64  BRPARM     64 bytes of RPARM text
=================================================================
  Start of work area used by block module
-----------------------------------------------------------------
 184 (B8)   SIGNED   72  BMSAVE     Save area used to call rec mod
-----------------------------------------------------------------
 256(100)   DOUBLE    8  BDUBL      Doublework work for CVD
-----------------------------------------------------------------
 264(108)   SIGNED    4  BRECMAX    Max logical record size for file
-----------------------------------------------------------------
 268(10C)   SIGNED    4  BRECNUM    Count of records sent/received.
-----------------------------------------------------------------
 272(110)   SIGNED    4  BHDRLEN    Record protocol header length
-----------------------------------------------------------------
 276(114)    BIT      1  BFLAGS     Work flags
         1... ....      RMODSET    Record module called before
 277(115)    HEX      3             Reserved.
```

```
------------------------------------------------------------------------
 280(118)  ADDRESS    4  RMODADR    Entry address of record module
------------------------------------------------------------------------
 284(11C)  ADDRESS    4  RMODMDTE   MDTE address of record module
========================================================================
```

Figure 21. Block module parameters

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. Secure BFX Internal Summary" on page 89). 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 submission 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. Secure BFX Internal Summary

The following paragraphs briefly summarize the internal structure of Secure BFX as it relates to writing user modules. Table 5 lists and describes the Secure 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.

Module	Description
BFXTIC	BFXTI entry point. Calls PRM to scan control parameters, calls the user- specified JMODULE or the BFX default JMODULE to send the job to the remote host. Calls the SND or RCV module to send or receive the file according to the user's parameters. If batched control statements were provided, it continues the cycle until SYSIN is exhausted.
BFXTRC	BFXTR entry point. Calls PRM to scan control parameters. Calls the SND or RCV module to send or receive the file according to the user's parameters. If batched control statements were provided, it continues the cycle until SYSIN is exhausted.
BFXJSC	BFXJS entry point. Scans JS control parameters and calls SCN to parse the parameters. Calls RCV to receive the batch jobs from remote sources and submit them to the system internal reader. Continues to call RCV until cancelled by the user.
BFXPRM	Parameter scanner for control parameters to the BFXTI, BFXTR, and BFXJS programs. Parses input and fills the TRansfer Option List (TROL) data structure with the user's parameters and/or BFX and installation-provided defaults.
BFXSCN	Parser for control parameters. This breaks down the control parameters into tokens and returns the tokens to the PRM or JSC calling routines.
BFXSUB	Secure BFX default job submission module. BFXSUB builds a new TRansfer Option List and calls the BFXSND file send module to send it to the remote BFXJS program.
BFXRCV	This module directs the process of receiving a file from a remote Secure BFX program. If invoked by TI, it will call CON to complete session negotiation from the OFFERing side. If invoked by TR, it will call CON to connect to TI and negotiate parameters. When the connection is successfully established, it will call the receiving block module to allow the data to be written to the file. During transfer, it processes all generated informational and error messages, and detects and handles error and EOF conditions.
BFXSND	This module directs the process of sending a file to a remote Secure BFX program. If invoked by TI, it will call CON to complete session negotiation from the OFFERing side. If invoked by TR, it will call CON 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.

Module	Description
BFXCON	This module issues an SOFFER or SCONNECT request to allow another Secure BFX program to connect to it. When the connection completes, the module negotiates the file transfer parameters.
BFXMDT	The module table is a non-executable module that contains a list of the currently available record, block, and job modules. The table contains the entry address for each module as well as flags that allow the correct high level language environment to be established before the module is called.
BFXJSD	This is a non-executable module that contains the default parameters to be used by the BFXJS program.
BFXRBM	This module is the Secure BFX default 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.
BFXRRM	The Secure BFX default receiving record module opens the sequential file for output, determines record and record size requirements, and accepts data from 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.
BFXSBM	This module is the Secure BFX default 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.
BFXSRM	The Secure BFX default sending record module opens the sequential file for input, determines record and record size requirements, 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.
BFXPRI	This module prints all internally generated messages on the LOGDD file. If the message is insufficiently severe for the current MSGLVL, then printing of the message is suppressed.
BFXERM	This module formats all Secure BFX error and informational messages, and places variable information in readable form within the message.
BFXERT	This is a non-executable module that contains the text of all Secure BFX error and informational messages.

Table 5. Secure BFX Default modules

Block Diagram

Many modules are used in several or all of the Secure BFX programs. The conceptual flow of control is different for the three BFX programs. Figure 22 is a block diagram for the BFXTI program. Table 5 on page 90 describes each component of the block diagram.

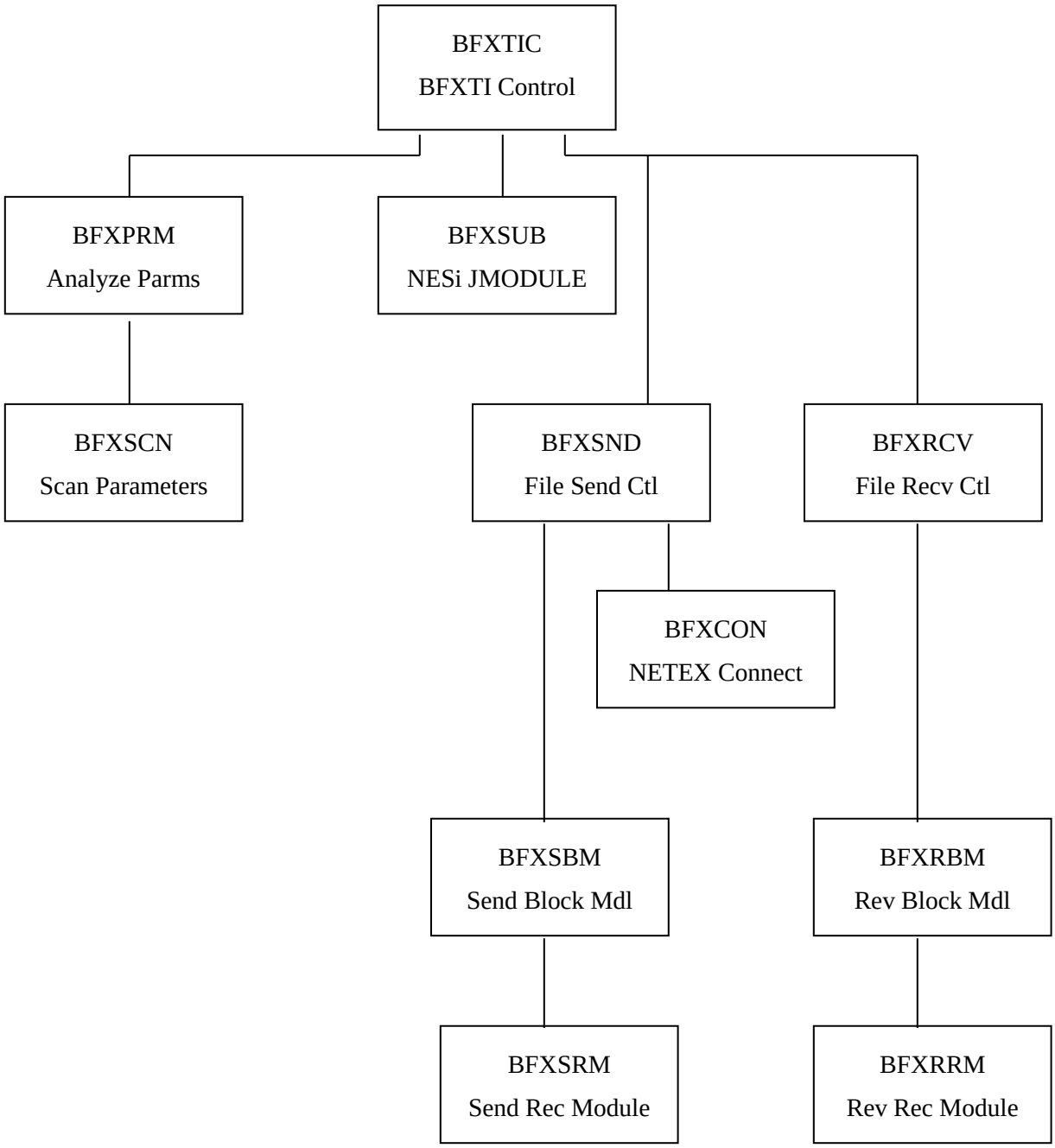

Figure 22. BFXTI Module Block Diagram

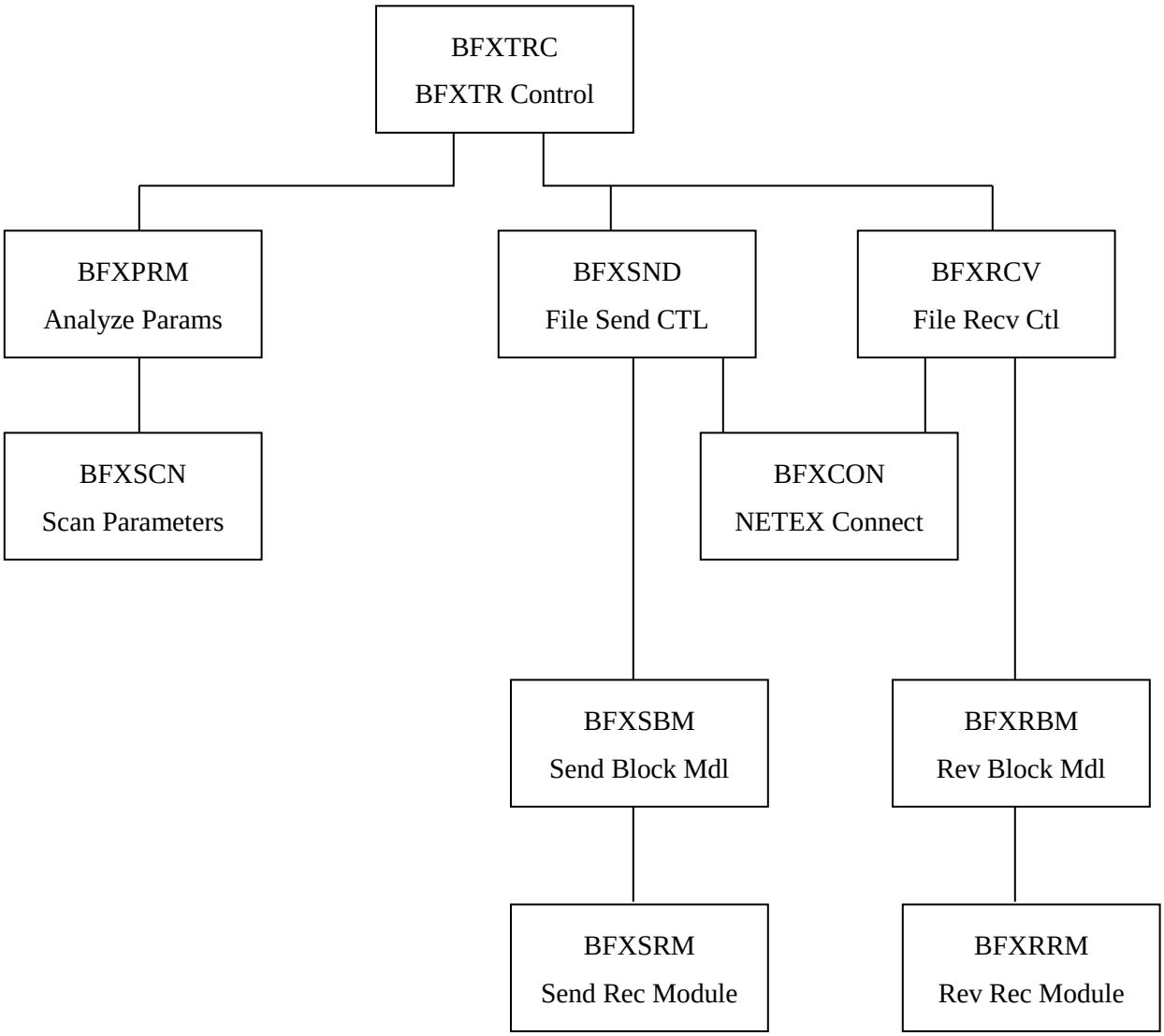

Figure 23. BFXTR Block Diagram

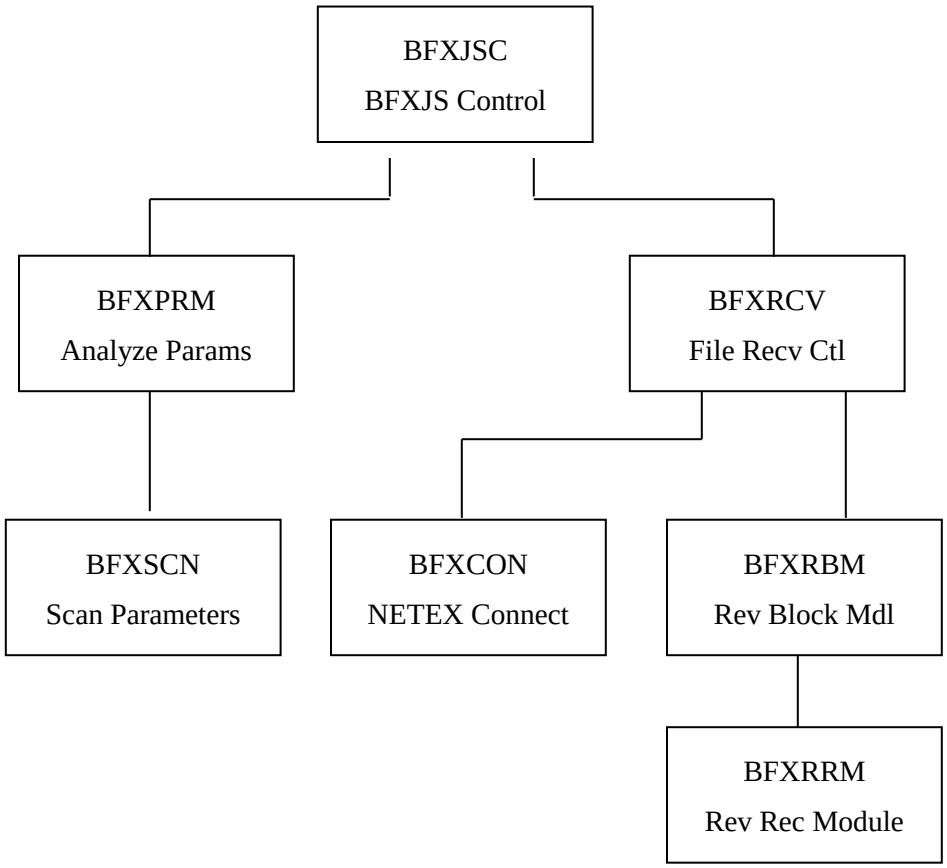

Figure 24. BFXJS Module Block Diagram

Secure BFX Module Descriptions

The following paragraphs describe seven of the default Secure 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.

Secure BFX Receive File Control (RCV)

This module receives control from the TIC, TRC, or JSC main modules when a file is to be transferred. The parameters to be used for the transfer are specified in the TROL that was built by the PRIM module or internally generated in the case of JSC. Its flow of control is as follows:

1. 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 determine the maximum record size.
2. On return from the block module, RCV adjusts the BLOCK= value upwards if necessary. Based on the calling main program (which is specified in the TROL) it calls CON (BFX session offer or BFX session connect) with the TROL as the passed parameter. These modules will allow the connection to complete

and complete the buffer size, least common multiple, minimum byte count, and file mode negotiations as detailed in the BFX general design specification.

3. Upon completion of connection negotiation, RCV allocates the requested number of buffers and begins the transfer process. It uses a multiple buffering technique to overlap Secure NetEx/IP processing of the incoming record with the file writes performed by the block module. Figure 25 on page 95 shows the normal flow of data transfer
4. Whenever the block module is called to write a block received from Secure NetEx/IP, the block module may return with a message. The message will be sent to the SND control module in the other application, and its contents will be placed on the SYSPRINT log files 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 Secure BFX, RCV will call the block and record modules with the EOI indication. Upon return, it will free the buffers allocated in step 2 and will return to the module that called RCV.

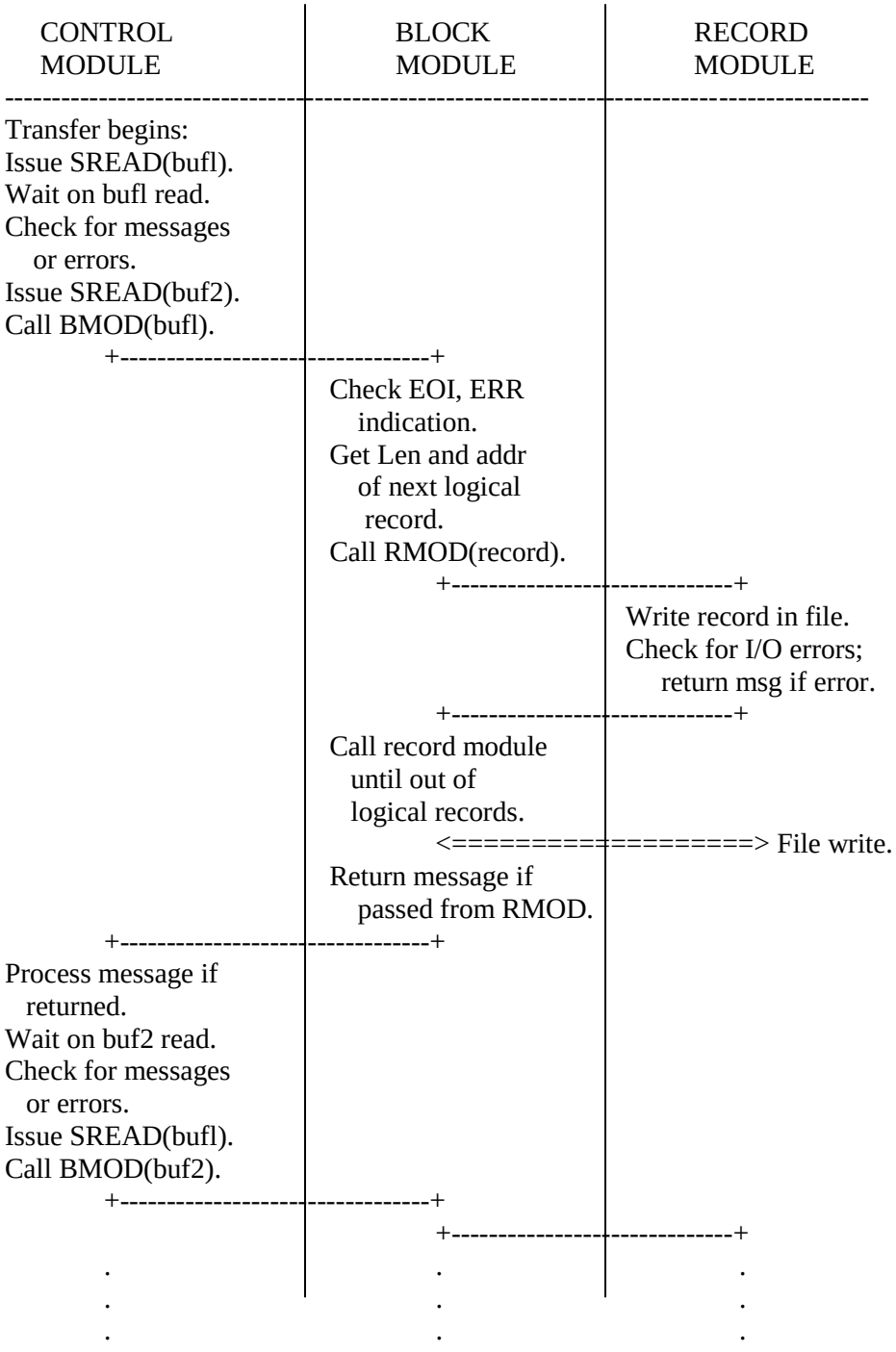

Figure 25. File Receive Data Flow

BFX Send File Control (SND)

This module receives control from the TIC, TRC, or JSC main modules when a file is to be transferred. The parameters to be used for the transfer are specified in the TROL that was built by the PRM module or internally generated as is the case when called by the default JMODULE (SUB). Its flow of control is as follows:

1. The sending 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 input and determine the maximum record size. This initialization logic is illustrated in Figure 26 on page 97.
2. On return from the block module, RCV adjusts the BLOCK= value upwards if necessary. Based on the calling main program (which is specified in the TROL) it calls CON (BFX session offer or BFX session connect) with the TROL as the passed parameter. These modules will allow the connection to complete and complete the buffer size, least common multiple, minimum byte count, and file mode negotiations as detailed in the Secure BFX General Design Specification.
3. Upon completion of connection negotiation, SND allocates the requested number of buffers and begins the transfer process. It uses a multiple buffering technique to overlap Secure NetEx/IP processing of the incoming record with the file writes performed by the block module. Figure 27 on page 98 below shows the normal flow of data transfer.
4. Whenever the block module is called to provide a block to be sent to Secure NetEx/IP, the block module may return with a message. The message will be sent to the RCV control module in the other application, and its contents will be placed on the SYSPRINT log files of both applications. If the message indicates a normal or abnormal end of transfer, SND will send the message and wait for a disconnect indication from the other Secure 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. Upon return from the block module, SND will free the buffers allocated in step 2 and will return to the module that called SND.
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 SYSPRINT file. 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.

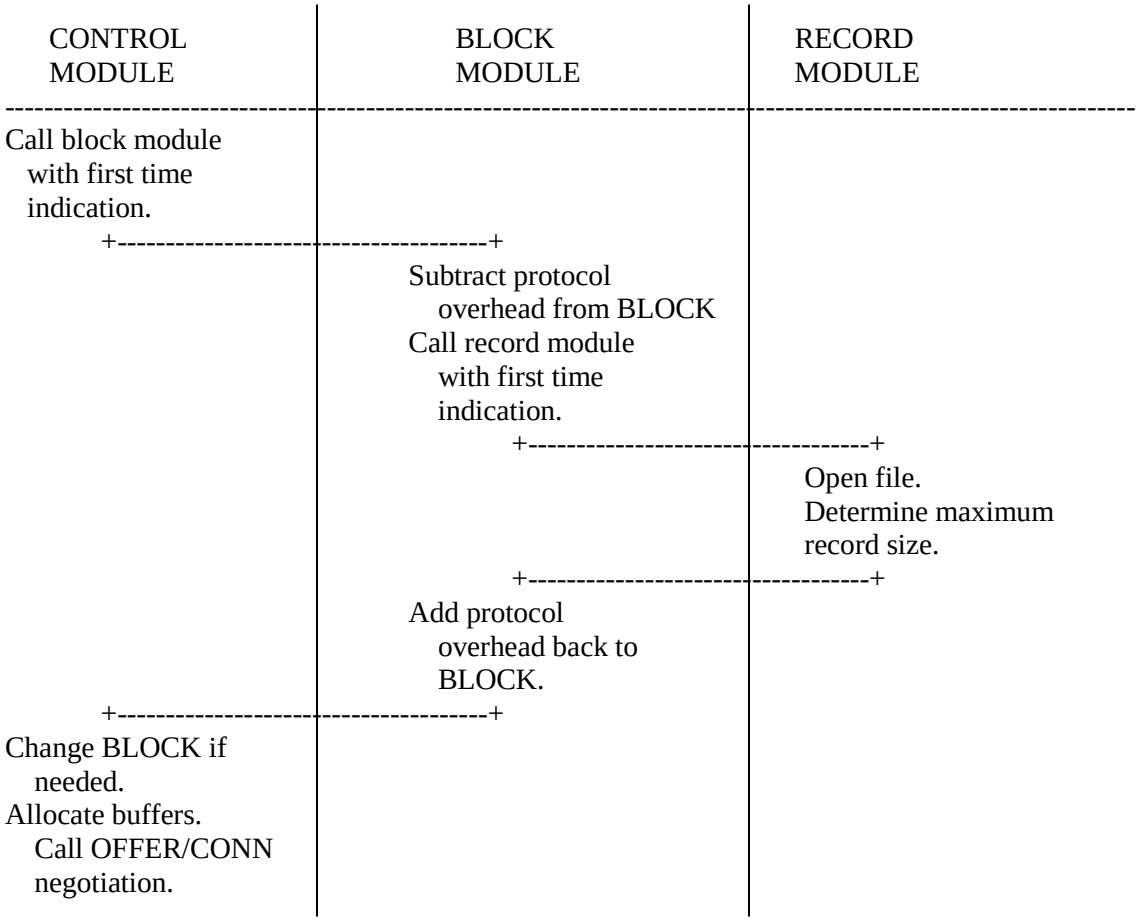

Figure 26. Send Receive Initialization Flow

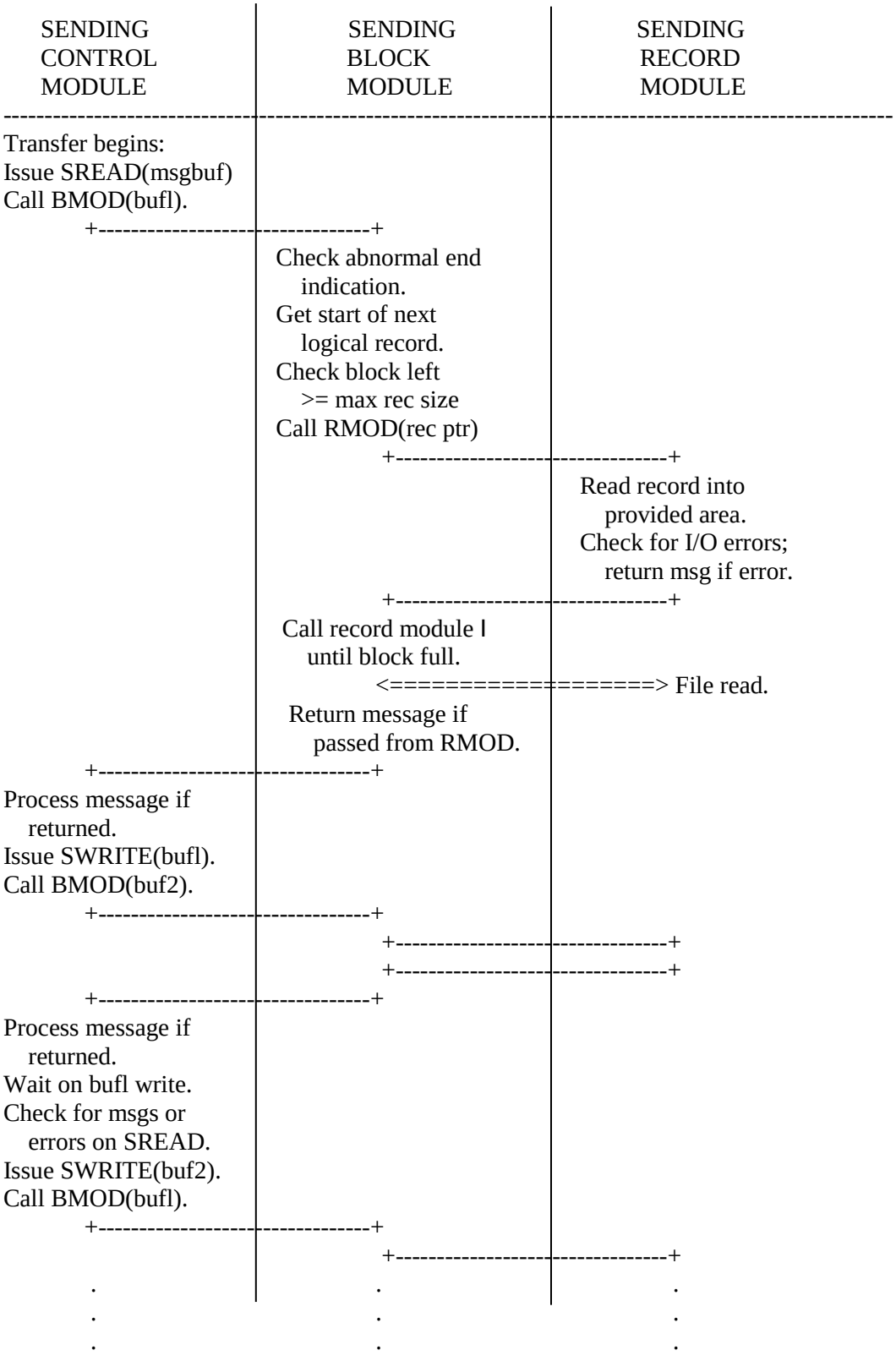

Figure 27. File Send Data Flow

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 Secure NetEx/IP. It is the responsibility of the receiving block module to decode all file transfer protocol information contained within the Secure NetEx/IP data block. The block module will then call the receiving record module once for each logical record contained within the file. Overall logic flow for this module is as follows:

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

Sending Block Module (SBM)

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

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

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

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

Receiving Record Module (RRM)

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

The BFX default receiving record module operates as follows:

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

Sending Record Module (SRM)

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

SRM flow of control proceeds as follows:

1. When the sending record module receives control for the first time, it opens the file whose DDNAME is passed to it from the block module. Using the LRECL parameter produced as a result of the OPEN, it returns the maximum logical record length to the block module. If the specified BLOCK= parameter is not large enough to accommodate the largest logical record, then the SND module will adjust the value of BLOCK upwards. If the open for the file does not succeed, then the record module will return an open failure message to the block module for logging.

2. Subsequent calls to the record module will provide the address and length in which the logical record is to be placed. The record module will GET the record into this block module provided area.

3. If the record module is called with an error or EOF indication, then the record module will close the file and return to the block module.

Secure BFX Standard Job Transmission Module (SUB)

This module is the standard Job Transmission Module (JMODULE). Its logic is as follows:

1. Using the information in the TROL data structure passed to it from the TRC or TIC control module, it constructs another TROL that will be used to do a character file transfer from the RMTJOB DDNAME to the BFXJS ID that is to receive the job on the remote host.
2. Using this constructed TROL, it calls the send control module to send the RMTJOB DDNAME as a remote file. On return from the send control module, the job will have been submitted to the remote host.

Appendix C. Secure BFX Error Messages

Secure BFX generates a variety of messages during the course of execution. A complete list of messages with the suggested action for each is shown below. 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 BFX messages have the following format:

```
BFXnnns message text
```

where:

BFX This indicates that this is a BFX error code.

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

s This indicates the message severity. The following codes are used:

- **I** informational messages
- **W** warning messages
- **E** error messages
- **S** severe error messages
- **F** fatal error messages

message text This is the message text.

The following are the messages issued by the BFX utility.

BFX001F Job Submission failed.

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

BFX002F BFX execution aborted.

Severity: 15 (Fatal error)
Module: BFXTR; BFXTIC in program BFXTI.
Explanation: The BFX program has detected a condition that makes it impossible to successfully continue execution. Generally, this is due to a failure to connect to the matching BFX program on the remote host. The BFX program will terminate.
User Response: The reason for the terminal failure will be indicated in previous BFX messages. Take the corrective action indicated by the previous message description.

BFX003F BFXJS execution aborted.

Severity: 15 (Fatal error)
Module: BFXJSC in program BFXJS.
Explanation: The BFXJS program has detected a condition that makes it impossible to continue offering job submission services. Generally this is due to termination of the Secure NetEx/IP SNXMAP program.
User Response: Restart Secure NetEx/IP SNXMAP or correct the error that caused Secure NetEx/IP SNXMAP to terminate. Restart BFXJS.

BFX004I BFXnn H215L Release x.y.z started.

Severity: 4 (Detailed informational)
Module: BFXJSC in program BFXJS.
BFXTIC in program BFXTI.
BFXTRC in program BFXRR.
Explanation: The BFXJS, BFXTI, or BFXTR programs have been started.
User Response: None

BFX006F "xxxxxxxx" not recognized in control stmt.

Severity: 15 (Fatal error)
Module: BFXPRM in programs BFXTI, BFXTR.
Explanation: When processing the parameters specified in the PARM field or a SYSIN control file, an unrecognizable parameter was found. "xxxxxxxx" contains the eight characters including and following the parameter in error. Generally, BFX execution will stop as indicated in subsequent messages.
User Response: Correct the syntax error and resubmit the BFX jobs.

BFX007W "xxxxxxxx" is only valid for BFXTI jobs -- ignored.

Severity: 9 (Warning)
Module: BFXPRM in programs BFXTI, BFXTR.
Explanation: A parameter (such as "NOSUB") that is applicable to job submission was supplied to the BFXTR program or to a SYSIN input statement other than the first one. The excess parameter is ignored.
User Response: Although processing will continue, the probable cause is an operations or setup error. Verify that the remainder of the BFX run proceeded as intended.

BFX008W "xxxxxxxx" is not valid for a SUBMIT statement -- ignored.

Severity: 9 (Recoverable error)
Module: BFXPRM in program BFXTI.
Explanation: A parameter (such as BLOCK=) that is only used for file transfer was pro- vided as an operand to the SUBMIT statement. The parameter is ignored and processing continues.
User Response: Although processing will continue, the probable cause is an operations or setup error. Verify that the remainder of the BFX run proceeded as intended.

BFX009W "xxxxxxxx" is only valid for a first BFXTI stmt -- ignored.

Severity: 9 (Recoverable error)
Module: BFXPRM in program BFXTI.
Explanation: A parameter (such as JRMODULE=) that is only associated with job submission was supplied in a BFXTI control statement other than the first. Since the BFXTR job has already been submitted, these parameters are not meaningful. The parameter is ignored and processing continues.
User Response: Although processing will continue, the probable cause is an operations or setup error. Verify that the remainder of the BFX run proceeded as intended.

BFX010I TO = or FROM = host name omitted.

Severity: 15 (Fatal Error)
Module: BFXPRM in programs BFXTI, BFXTR
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.
User Response: Supply the TO= or FROM = parameter required and rerun the job.

BFX011I ID = BFX identifier omitted.

Severity: 15 (Fatal error)
Module: BFXPRM in programs BFXTI, BFXTR.
Explanation: The ID parameter which uniquely identifies the BFX job on the initiating system was not supplied. There is no default for this parameter.
User Response: Supply the ID= parameter and rerun the job.

BFX012F Specified buffer size too large.

Severity: 15 (Fatal error)
Module: BFXPRM in BFXTI, BFXTR.
Explanation: The BLOCK= or JBLOCK= parameter specified a value greater than 60K bytes.
User Response: Correct the BLOCK= or JBLOCK = parameter and resubmit the job.

BFX013F "xxxxxxxx" is not a SEND/RECV/SUBMIT command.

Severity: 15 (Fatal error)
Module: BFXPRM in BFXTI, BFXTR.
Explanation: The first operand in a control statement was not 'SEND', 'RECEIVE', 'SUBMIT' or a suitable abbreviation. Processing is terminated.
User Response: Correct the erroneous -control statement and resubmit the job.

BFX021F NETEX Communications Subsystem is being shut down.

Severity: 15 (Fatal error)
Module: BFXCON in BFXTI, BFXTR, BFXJS. BFXRCV in BFXTI, BFXTR, BFXJS. BFXSND in BFXTI, BFXTR, BFXJS.
Explanation: During the connection process or in the middle of a job or file transfer, a BFX program received an indication that Secure NetEx/IP is abruptly terminating. This can be caused by user cancellation of Secure NetEx/IP or internal Secure NetEx/IP software problems. Execution is terminated as no further data transfer will be possible until Secure NetEx/IP (SNXMAP) is restarted.
User Response: Consult with operations to determine the cause of the Secure NetEx/IP shutdown. Resubmit the job when Secure NetEx/IP is once again active. File cleanup procedures may be needed if a file transfer was in progress at the time of the failure.

BFX023F Remote BFX program did not start.

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

- Job Control Language, errors in the BFXTR job prevented successful execution of the BFXTR program.

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

BFX024F Remote host ceased communicating.

Severity: 15 (Fatal error)
Module: BFXSND in BFXTI, BFXTR, BFXJS. BFXRCV in BFXTI, BFXTR, BFXJS.
Explanation: During the transfer of a file or a job, the Secure BFX program received an indication from Secure NetEx/IP that all communications with the other host have ceased. This is generally caused by a system crash on the remote host, abrupt failure or user cancellation of Secure NetEx/IP on the remote host, or a hardware failure in the physical connection between the two hosts.
User Response: Consult with operations to determine the cause of the failure. Resubmit the job when the connection is once again active. File cleanup procedures may be needed if a file transfer was in progress at the time of the failure.

BFX025F Remote BFX aborted execution.

Severity: 15 (Fatal error)
Module: BFXSND in BFXTI, BFXTR, BFXJS. BFXRCV in BFXTI, BFXTR, BFXJS.
Explanation: During the transfer of a file or a job, the BFX program received an indication from Secure NetEx/IP that the BFX program on the remote host abnormally terminated. Processing is terminated as no further transfer will be possible.
User Response: Examine the output from the job on the remote host to determine the cause of the failure. Correct the error and resubmit the jobs.

BFX026F Remote host NETEX not present.

Severity: 15 (Fatal error)
Module: BFXCON in BFXTI, BFXTR.
Explanation: When an attempt was made to connect to the corresponding BFX program, the Secure NetEx/IP on the local system reported that no Secure NetEx/IP SNXMAP was present on the remote host. Processing is terminated as no data transfer is possible.
User Response: Consult with operations to determine if Secure NetEx/IP (SNXMAP) should have been present on the remote host. Check if paths to the remote host are available (i.e not drained) Rerun the jobs when Secure NetEx/IP, and at least one path, is available on both hosts.

BFX027F Specified host is not on the network.

Severity: 15 (Fatal error)
Module: BFXCON in BFXTI, BFXTR.
Explanation: When BFXTI was attempting to submit a job, or BFXTR is attempting to connect back to the BFXTI program, Secure NetEx/IP reported that the name of the host processor specified is unknown.
User Response: The probable cause of this error is due to an erroneous TO= or FROM= parameter. A second possibility is that the installation has changed the host names used by Secure NetEx/IP. Correct the error and resubmit the job.

BFX028F Access to specified host denied.

Severity: 15 (Fatal error)
Module: BFXCON in BFXTR, BFXTR.
Explanation: When BFXTI was attempting to submit a job, or BFXTR is attempting to connect back to the initiating BFXTI program, Secure NetEx/IP informed the program that access to the specified host has been denied by the local computer operator. The run is terminated as communications between the two hosts cannot take place.
User Response: Computer operations is using a feature of Secure NetEx/IP that can temporarily prohibit access to a host that is undergoing maintenance, performing classified or confidential work, and

so on. Consult with operations to determine when communications with the specified host will once more be permitted. Resubmit the jobs at that time.

BFX029F Not authorized to use NETEX.

Severity: 15 (Fatal error)

Module: BFXCON in BFXTR, BFXTI, BFXJS.

Explanation: When a BFX module attempted to contact Secure NetEx/IP, it was informed by Secure NetEx/IP that the BFX program or user account is not authorized to use Secure NetEx/IP.

User Response: Consult with computer operations, systems, or accounting to determine if the restriction on Secure NetEx/IP can be removed. Resubmit the job when the restriction has been eliminated.

BFX030S NETEX error. NRBSTAT=ssss,NRBIND=iii,NTXID=nnnnnnnn.

Severity: 12 (Severe error)

Module: BFXCON in BFXTI, BFXTR, BFXJS. BFXSND in BFXTI, BFXTR, BFXJS.

Explanation: Secure NetEx/IP has reported an error to the BFX program that is not an expected condition. "ssss" is the four digit status code returned by Secure NetEx/IP; "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.

User Response: Refer to Secure NetEx/IP 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.

BFX040F NETEX Communications Subsystem terminated.

Severity: 15 (Fatal error)

Module: BFXSND in BFXTI, BFXTR, BFXJS. BFXRCV in BFXTI, BFXTR, BFXJS.

Explanation: During the process of transferring a file or batch job, BFX discovered that the local Secure NetEx/IP is abruptly terminating. This could be caused by user cancellation of Secure NetEx/IP (SNXMAP) or Secure NetEx/IP internal software problems.

User Response: Consult with operations to determine when the local Secure NetEx/IP (SNXMAP) will once more be available. Resubmit the jobs when Secure NetEx/IP is running.

BFX042S BFX program timed out to NETEX.

Severity: 15 (Severe error)

Module: BFXSND in BFXTI, BFXTR, BFXJS. BFXRCV in BFXTI, BFXTR, BFXJS.

Explanation: The BFX program suspended execution for a sufficiently long time that Secure NetEx/IP terminated the connection between the two BFX programs. The transfer of this particular file is aborted. If batched execution is being performed, then BFX will attempt to transfer the remaining files.

User Response: This is generally due to difficulties in system tuning, or exceptionally long delays in such activities as mounting tape volumes other than the first in a multivolume file. If the problem was not caused by operational errors, the Secure NetEx/IP system programmer may have to adjust the Secure NetEx/IP READTO= parameter upward to accommodate the long delay.

BFX043S BFX Protocol error -- premature disconnect.

Severity: 15 (Severe error)

Module: BFXSND in BFXTI, BFXTR, BFXJS. BFXRCV in BFXTI, BFXTR, BFXJS.

Explanation: The remote BFX program terminated the connection at a time that was not anticipated by the local BFX program.

User Response: This is an internal BFX error. It should be brought to the attention of installation BFX support personnel.

BFX044S Remote BFX program timed out.

Severity: 15 (Severe error)

Module: BFXRCV in BFXTI, BFXTR, BFXJS.
Explanation: The remote BFX program suspended execution for a sufficiently long time that the remote Secure NetEx/IP broke the connection between the two BFX programs. The transfer of this particular file is aborted. If batched execution is being performed, then BFX will attempt to transfer the remaining files.
User Response: This is generally due to difficulties in system tuning, or exceptionally long delays in such activities as mounting tape volumes other than the first in a multivolume file. If the problem was not caused by operational errors, the Secure NetEx/IP system programmer may have to adjust the Secure NetEx/IP READTO = parameter on the remote host to accommodate the long delay.

BFX050F BFX Protocol error -- run aborted.
Severity: 15 (Fatal error)
Module: BFXSND in BFXTI, BFXTR, BFXJS. BFXRCV in BFXTI, BFXTR, BFXJS.
Explanation: One of the two BFX control modules detected invalid protocol in the messages exchanged between them. The BFX program will abend.
User Response: This is an internal BFX error that should be brought to the attention of BFX support personnel.

BFX051F BFX Abend Unnn in Module $$$$$$.
Severity: 15 (Fatal error)
Module: BFXRBM, in BFXTI, BFXTR, BFXJS.
Explanation: BFX encountered an internal fatal error condition and cannot continue processing. The message contains the abending module name and the user abend code.
User Response: This is an internal BFX error that should be brought to the attention of BFX support personnel. A dump is generated with the message. See the appropriate BFX manual for a list of user abend codes.

BFX052F Invalid BFX Header Data at Record nnnnn; Buf Addr=aaaaaaaa; Data=dddddddd; RC=rr.
Severity: 15 (Fatal error)
Module: BFXRBM in BFXTI, BFXTR, BFXJS
Explanation: The BFX header record received from the sending side was in error. Processing cannot continue. The message contains the failing record number (nnnnn) received at the time of the error, the location (aaaaaaaa) within the BFX buffer, the BFX header data (dddddddd), and the reason code (rr) where:

rr = 1 Character sequence number(modulo 100) is not equal to the record number.

rr = 2 Character record length field is not numeric.

rr = 3 Character delimiter level is not numeric.

rr = 10 Binary record number is invalid.

User Response: This is an internal BFX error that should be brought to the attention of BFX support personnel. If this is BFXJS, the session will be disconnected, otherwise this message will be issued by WTO and an Abend taken.

BFX084F BFX protocol error -- data received when sending.
Severity: 15 (Terminal error)
Module: BFXSND in BFXTI, BFXTR
Explanation: The file transfer information received from the remote BFX was incorrect. The remote BFX issued message BFX220I to indicate that it was sending data when the local BFX was also sending data.
User Response: The BFXTI/BFXTR pair cannot both send at the same time. Correct the transfer information so that one side specifies “Receive”.

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

Severity: 12 (Terminal error)

Module: BFXCON in BFXTI

Explanation: HOSTCHK is specified on the control statements (or is the installation-defined default). A CONNECT arrived from BFXTR, and the hostname it is trying to connect to does not match the HOST specified on the 'SEND TO hostname' or 'RECEIVE FROM hostname' BFXTI statements. The connection is rejected.

User Response: Determine the cause of the error. If dissimilar NCTs are used on both sides of the connection, or group host names are used, or a local loopback hostname (NCTLCLxx) is used, HOSTCHK will not work in any of these configurations. If that is the case, specify 'NOHOSTCH' and rerun the job. If none of these conditions are true, then look for a configuration or procedural error when running these jobs.

BFX101I File ffffffff done; nnnn records sent.

Severity: 6 (Informational)

Module: BFXSRM in BFXTI, BFXTR.

Explanation: The BFX default standard sending record module has detected normal end of file on the input file specified by DDNAME "ffffffff". The total number of QSAM logical records sent for this particular file is "nnnn". At the time the message was issued, the last record of the file will already have been sent to the receiving BFX.

User Response: None.

BFX102S File ffffffff Permanent I/O error -- < SYNAD info>.

Severity: 12 (Severe error)

Module: BFXSRM in BFXTI, BFXTR. BFXRRM in BFXTI, BFXTR, BFXJS.

Explanation: The QSAM access method reported a permanent I/O error reading or writing a file during transfer of the job or transfer of the file. The DDNAME of the file is indicated by "ffffffff". The SYNAD information associated with the error follows the text of the message. Transfer of this file is aborted. If batched transfer of files is being performed, BFX will attempt to transfer the rest of the specified files.

User Response: Determine the cause of the I/O error. If the error can be corrected, rerun the BFX jobs.

BFX103S Cannot find record module mmmmmmmm.

Severity: 12 (Severe error)

Module: BFXSBM in BFXTI, BFXTR. BFXRBM in BFXTI, BFXTR, BFXJS.

Explanation: The record module specified by the RMODULE= 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.

User Response: This can be caused because the module does not exist, or if the module does exist but was generated as being the wrong type.

BFX104S Record module "mmmmmmmm" language unsupported.

Severity: 12 (Severe error)

Module: BFXSBM in BFXTI, BFXTR. BFXRBM in BFXTI, BFXTR, BFXJS.

Explanation: The module specified by the RMODULE= operand was found in the module table, but it is written in a source language that is not currently supported by BFX.

User Response: This is principally a diagnostic message for incorrect module table generation during BFX installation. If the error is genuine, then the module needs to be rewritten.

BFX105S Cannot find block module mmmmmmmm.

Severity: 12 (Severe error)
Module: BFXSND in BFXTI, BFXTR. BFXRCV in BFXTI, BFXTR, BFXJS.
Explanation: The block module specified by the BMODULE= parameter was not installed in this copy of the BFX program. Transfer of this file is aborted; if batched transfer of files is being performed, BFX will attempt to transfer the rest of the specified files.
User Response: This can be caused because the module does not exist, or if the module does exist but was generated as being the wrong type.

BFX106S Block module "mmmmmmmm" language unsupported.

Severity: 12 (Severe error)
Module: BFXSND in BFXTI, BFXTR. BFXRCV in BFXTI, BFXTR, BFXJS.
Explanation: The module specified by the BMODULE= operand was found in the module table, but it is written in a source language that is not currently supported by BFX.
User Response: This is principally a diagnostic message for incorrect module table generation during BFX installation. If the error is genuine, then the module needs to be rewritten.

BFX110S Cannot open input file ffffffff.

Severity: 12 (Severe error)
Module: BFXSRM in BFXTI, BFXTR.
Explanation: The standard sending record module attempted to open a file with a DDNAME (or FILEDEF) of "ffffffff". The open did not succeed. Often a CMS or z/OS message will accompany this message that indicates the cause of the error. Transfer of this file is aborted. If batched transfer is used, then BFX will attempt to transfer the remaining files.
User Response: Correct the reason for the open failure. The most probable cause is a missing FILEIN DD or FILEDEF statement. Transfer the file that was unsuccessfully sent.

BFX111S Record module initialization failed.

Severity: 12 (Severe error)
Module: BFXSBM in BFXTI, BFXTR. BFXRBM in BFXTI, BFXTR, BFXJS.
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.
User Response: Correct the condition in the user-written record module.

BFX112S Block module initialization failed.

Severity: 12 (Severe error)
Module: BFXSND in BFXTI, BFXTR. BFXRCV in BFXTI, BFXTR, BFXJS.
Explanation: A user-written block 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.
User Response: Correct the condition in the user-written block module.

BFX113S Sending record module aborted transfer.

Severity: 12 (Severe error)

Module: BFXSBM in BFXTI, BFXTR.

Explanation: During the transfer of a file or a batch job, 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.

User 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)

Module: BFXRBM in BFXTI, BFXTR, BFXJS.

Explanation: During the transfer of a file or a batch job, 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.

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

BFX115S Sending block module aborted transfer.

Severity: 12 (Severe error)

Module: BFXSND in BFXTI, BFXTR.

Explanation: During the transfer of a file or a batch job, 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.

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

BFX116S Receiving block module aborted transfer.

Severity: 12 (Severe error)

Module: BFXRCV in BFXTI, BFXTR, BFXJS.

Explanation: During the transfer of a file or a batch job, a user-written receiving block module returned an abort transfer indication (BUFLEV= 16.) The user module did not supply a message with the abort code, so this default message is printed. Transfer of this file is aborted; if batched execution is in use, BFX will attempt to transfer the remaining files.

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

BFX117S Sending record module abort processed.

Severity: 12 (Severe error)

Module: BFXSBM in BFXTI, BFXTR.

Explanation: Due to a loss of communications or an error condition reported by the BFX program, a user-written sending record module was called with the abort condition (BUFLEV= 16). Upon return, the user module did not supply a message to acknowledge the abort, so this default message is printed. Processing will continue as indicated by the messages that preceded this one.

User Response: Correct the condition indicated in the messages previous to this one.

BFX118S Sending block module abort processed.

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

BFX119S Receiving record module abort processed.

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

BFX120S Receiving block module abort processed.

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

BFX121S RECSZ nnnnn too large for BFX block mmmmm

Severity: 12 (Severe error)
Module: BFXSND in BFXTI, BFXTR, BFXJS
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. This message may also be issued if one of the two BFX programs failed to open its file and the other program had a maximum logical record length greater than 250 bytes. In that case, the real error message will follow the BFX121S message.
User Response: Adjust the BLOCK= parameter in one of the two BFX programs so it is sufficient to transfer the file; or correct inability of one BFX program to open a file.

BFX122F Invalid file transfer protocol information.

Severity: 15 (Terminal error; ABEND will follow)
Module: BFXRBM in BFTI, BFXTR, BFXIS
Explanation: The file transfer information sent to the receiving BFX was found to be incorrect. This may be due to an internal BFX or Secure NetEx/IP integrity error, or due to a user-written block module that is incorrectly sending data to the BFX default receiving block module. As this error is very serious,, the BFX program will unilaterally ABEND.
User Response: If the error was caused by a user-written block module, correct the coding error that caused incorrect data to be sent. If only standard BFX code was used, bring the error to the immediate attention of NESi Support .

BFX123F File transfer protocol sequence error

Severity: 15 (Terminal error; ABEND will follow)
Module: BFXRBM in BFXTI, BFXTR, BFXJS.

Explanation: The file transfer information sent to the receiving BFX was found to be incorrect. A check of record numbers indicated that transferred records are either missing or duplicated. This may be due to an internal BFX or Secure NetEx/IP integrity error, or due to a user-written block module that is incorrectly sending data to the standard BFX receiving block module. As this error is very serious, the BFX program will unilaterally ABEND.
User Response: If the error was caused by a user-written block module, correct the coding error that caused incorrect data to be sent. If only standard BFX code was used, bring the error to the immediate attention of NESi Support.

BFX124S MODE= parameters inconsistent for both BFX jobs.
Severity: 12 (Severe error)
Module: BFXRCV in BFXTI, BFXTR, BFXJS. BFXSND in BFXTI, BFXTR
Explanation: In a BFX program pair, one side had MODE=BIT specified and the second had MODE=CHAR. 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.
User Response: Correct the erroneous specification and transfer the files that were not sent.

BFX125F Record number nnnnnnnn exceeds Blksize of file.
Severity: 12 (Fatal error)
Module: BFXSRM in BFXTI, BFXTR
Explanation: The Sending Record Module determined that the current record number (nnnnnnnn) exceeded the maximum Blksize of the file while attempting to move this record to BFX. This would result in the new record overrunning the allocated area for the block.
User Response: Determine the maximum logical record size of the file before transfer and adjust the BLOCK parameter accordingly.

BFX200I File transfer complete.
Severity: 7 (Informational)
Module: BFXRCV in BFXTI, BFXTR, BFXJS.
Explanation: This message is issued when the receiving BFX has received an EOI message. The user record module did not supply a message to acknowledge the EOI, so this default message is printed.
User Response: None.

BFX201I File ffffffff done; nnnn records received.
Severity: 6 (Informational)
Module: BFXRCV in BFXTI, BFXTR, BFXJS.
Explanation: This message is issued when the receiving BFX processes the last record of the file. When issued, it indicates that the last record was received and the output file was successfully closed. "ffffffff" is the DDNAME or FILEDEF of the file used for output; "nnnn" is the number of records that were written to the output file.
User Response: None

BFX202W File ffffffff done; nnnn records received, tttt truncated.
Severity: 9 (Warning)
Module: BFXRRM in BFXTI, BFXTR, BFXJS.
Explanation: This message is normally issued when the receiving BFX program processes the last record of the file. It indicates that the last record was received and that the output file was successfully closed. However, the specified LRECL of the output file was not large enough to hold all data sent from the receiving file; the long records have been truncated. "ffffffff" is the DDNAME or FILEDEF of the file used for output, "nnnn" is the number of records written to the file; "tttt" is the number of records shortened because their length exceeded the LRECL of the output file. If batched execution is being used, BFX will continue to transfer the batched files.

User Response: If the truncation was an expected condition, then the message may be ignored. If it was not expected, then the DCB information for either the input or output file should be corrected and the file transferred once again.

BFX203E File ffffffff transfer aborted; nnnn records received.

Severity: 10 (Error)
Module: BFXRRM in BFXTI, BFXTR, BFXJS.
Explanation: This message is issued by the receiving BFX code when the file transfer process is aborted either due to the loss of Secure NetEx/IP communication or some error detected by the sending BFX. "ffffffff" contains the DDNAIME or FILEDEF of the output file; "nnnn" contains the number of records written to the output 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 depending on the condition that caused transfer to abort. The original error will be reported by other BFX messages.
User Response: Correct the error that that caused the abort in the first place. Transfer the file again.

BFX204E File ffffffff transfer aborted; nnnn records received, tttt truncated.

Severity: 11 (Error)
Module: BFXRRM in BFXTI, BFXTR, BFXJS.
Explanation: This message is issued by the receiving BFX code when the file transfer process is aborted either due to the loss of Secure NetEx/IP communication or some error detected by the sending BFX. "ffffffff" contains the DDNAME or FILEDEF of the output file; "nnnn" contains the number of records written to the output file before the abort caused transfer to stop. In addition to the abort, a number of records were truncated as described in message BFX101I. The transfer of this file is always aborted; subsequent files in a batched run will be transferred depending on the condition that caused transfer to abort. The original error will be reported by other BFX messages.
User Response: Correct the error that caused the abort in the first place. Transfer the file again.

BFX205I File ffffffff done; nnnn records sent.

Severity: 6 (Informational)
Module: BFXSRM in BFXTI, BFXTR.
Explanation: This message is issued after sending BFX transmits the last record of the file. When issued, it indicates that the last record was sent, the input end of file condition was detected, and the file was successfully received. "ffffffff" is the DDNAME or FILEDEF of the file used for input; "nnnn" is the number of records that were read from the input file.
User Response: None.

BFX206E File ffffffff transfer aborted; nnnn records sent.

Severity: 10 (Error)
Module: BFXSRM in BFXTI, BFXTR.
Explanation: This message is issued by sending the 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. "ffffffff" contains the DDNAME or FILEDEF of the input file; "nnnn" 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 depending on the condition that caused transfer to abort. The original error will be reported by other BFX messages.
User Response: Correct the error that caused the abort in the first place. Transfer the file again.

BFX210S Cannot open input file ffffffff.

Severity: 12 (Severe error)
Module: BFXSRM in BFXTI, BFXTR.
Explanation: The sending BFX program was unable to open the input file whose DDNAME is specified by "ffffffff". Transfer of this particular file is aborted. BFX will attempt to transfer subsequent

files in a batched run following this error. Generally VM or Z/OS will issue a message indicating the reason for the open failure.

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

BFX211S Cannot open output file ffffffff.

Severity: 12 (Severe error)

Module: BFXRRM in BFXTI, BFXTR.

Explanation: The receiving BFX program was unable to open the output file whose DDNAIME is specified by "ffffffff". Transfer of this particular file is aborted. BFX will attempt to transfer subsequent files in a batched run following this error. Generally VM or Z/OS will issue a message indicating the reason for the open failure.

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

BFX213S RECFM must be specified for file ffffffff.

Severity: 12 (Severe error)

Module: BFXRRM in BFXTI, BFXTR.

Explanation: The output file already exists and is a VS or VBS file. However, the DD card does not specify the record format of the file. Therefore, BFX will not know that it is a spanned file and will not correctly open it for BSAM processing.

User Response: Add DCB = (RECFM = VBS) to the DD statement for the file ffffffff

BFX214S LRECL = X must be specified for file ffffffff.

Severity: 12 (Severe error)

Module: BFXSRM in BFXTI, BFXTR

Explanation: The input file for the sending record module is "LRECL=X" format. However, the DD card does not specify the LRECL = X in the JCL. Module BFXSRM did not get the proper information before opening the file and hence fails this transfer.

User Response: Add DCB=(LRECL=X) to the JCL before initiating the SEND function for the file.

BFX215F Fatal I/O Error Sccc-rr; ffffffff Transfer Aborted; Records Received nnnnnnnn.

Severity: 15 (Fatal error)

Module: BFXRRM in BFXTI, BFXTR, BFXJS.

Explanation: During the process of writing a file, a fatal I/O error occurred. The message displays the System code (ccc) and reason code (rr) from the operating system. The transfer of this file ("ffffffff") is aborted. The number of records received (nnnnnnnn) is displayed. Subsequent files in a batched BFX execution will not be transferred.

User Response: Analyze the System code in the message, fix the JCL or the file, and rerun the job.

BFX220I Sending file ffffffff.

Severity: 4 (Informational)

Module: BFXSRM in BFXTI, BFXTR.

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 DDNAME or FILEDEF of the file is "ffffffff".

User Response: None

BFX221I Receiving file ffffffff.

Severity: 4 (Informational)

Module: BFXRRM in BFXTI, BFXTR, BFXJS.

Explanation: The receiving BFX has successfully opened the output file and is ready to receive file data. The DDNAME or FILEDEF of the file is "ffffffff".

User Response: None

BFX300I Offering ssssssss; Block out size nnnn; NTXID = nnnnnnnn.

Severity: 2 (Diagnostic)
Module: BFXCON in BFXTI.
Explanation: The BFXTI transfer initiate program has issued a NETEX SOFFER to wait for the BFXTR program to connect to it. The name offered is "ssssssss" which will be the specified ID= of the user's input parameters. The expected transfer is a receive operation; the incoming block size that the receiving BFX would like to used is nnnn.
User Response: None

BFX301I Offering ssssssss; Block in size nnnn, NTXID=nnnnnnnn.
Severity: 2 (Diagnostic)
Module: BFXCON 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. The expected transfer is a receive operation; the incoming block size that the receiving BFX would like to used is nnnn.
User Response: None

BFX302I Connecting to ssssssss on host hhhhhhhh; Block in size nnnn; NTXID = nnnnnnnn.
Severity: 2 (Diagnostic)
Module: BFXCON in BFXTI, BFXTR.
Explanation: When BFXTR is connecting back to it starting BFXTI, it has issued a NETEX SCONNECT to establish communications. "ssssssss" is the name to connect to as specified in the ID = or JID = user parameters; "hhhhhhhh" is the host name specified in the TO= or FROM= parameters. The direction of file transfer will cause this program to receive the file; the block size that is program is prepared to receive is nnnn.
User Response: None

BFX303I Connecting to ssssssss on host hhhhhhhh; Block out size nnnn; NTXID = nnnnnnnn.
Severity: 2 (Diagnostic)
Module: BFXCON in BFXTI, BFXTR.
Explanation: When BFXTI is submitting a job to BFXJS, or when BFXTR is connecting back to its starting BFXTI, the BFX program has issued a NETEX SCONNECT to establish communications. "ssssssss" is the name to connect to as specified in the ID= or JID = user parameters; "hhhhhhhh" is the host name specified in the TO= or FROM= parameters. The direction of file transfer will cause this program to send a file; the block size that is program is prepared to send is nnnn.
User Response: None

BFX304I Connect complete.
Severity: 2 (Diagnostic)
Module: BFXCON in BFXTI, BFXTR.
Explanation: A previously issued NETEX SCONNECT (see messages BFX302I and BFX303I) has successfully completed.
User Response: None

BFX305W Connect failed; ssssssss busy.
Severity: 1 (Diagnostic)
Module: BFXCON in BFXTI, BFXTR.
Explanation: A previously issued NETEX SCONNECT (see messages BFX302I and BFX303I) did not succeed because the OFFER'ed application (hopefully BFXJS) was currently in use by some other network application. Secure NetEx/IP will retry the connection a number of times and at intervals determined during BFX generation.
User Response: None

BFX306W Connect failed; ssssssss not offered.

Severity: 1 (Diagnostic)
Module: BFXCON in BFXTI, BFXTR
Explanation: A previously issued NETEX SCONNECT (see messages BFX302I and BFX303I) did not succeed because the OFFER'ed application was not currently available. This is not always a terminal error; it can be caused by connecting to BFXJS during a small window between batch job submissions; or by BFXTI if a job is submitted from a heavily overloaded system to a very responsive one. Secure NetEx/IP will retry the connection a number of times and at intervals determined during BFX generation.
User Response: None

BFX307I Offer complete.
Severity: 2 (Diagnostic)
Module: BFXCON in BFXTI, BFXJS.
Explanation: A previously issued NETEX SOFFER request (see messages BFX300I and BFX301I) has completed successfully.
User Response: None

BFX308I Confirm issued.
Severity: 2 (Diagnostic)
Module: BFXCON in BFXTI, BFXJS.
Explanation: A NETEX SCONFIRM is being issued in response to a previously completed SOFFER.
User Response: None

BFX309I Confirm complete.
Severity: 2 (Diagnostic)
Module: BFXCON in BFXTI, BFXJS.
Explanation: A previously issued NETEX SCONFIRM (see message BFX308I) has completed successfully.
User Response: None

BFX310I Connect confirm read issued.
Severity: 2 (Diagnostic)
Module: BFXCON in BFXTI, BFXTR.
Explanation: Following a successful SCONNECT request, the BFX program has issued an SREAD to obtain the SCONFIRM response from the other program.
User Response: None.

BFX311I Connect confirm complete.
Severity: 2 (Diagnostic)
Module: BFXCON in BFXTI, BFXTR.
Explanation: The SREAD issued to accept a SCONFIRM message from the remote BFX has successfully completed. The Secure NetEx/IP session establishment process is now complete.
User Response: None.

BFX312I Block size bbbb, datamode dddd, LCM ll.
Severity: 2 (Diagnostic)
Module: BFXCON in BFXTI, BFXTR, BFXJS.
Explanation: The diagnostic message is issued by all BFX modules when the session negotiation process is complete. "bbbb" is the actual Secure NetEx/IP block size (in bytes) that will be used to transfer the file or job. "dddd" consists of four hexadecimal digits that give the Secure NetEx/IP 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 is being sent to non-IBM processors with more than one character per word.

User Response: None

BFX313W NETEX down-BFXJS will wait $$ sec and reoffer.

Severity: 12 (Severe)
Module: BFXJS
Explanation: This message is issued by BFXJS when it has attempted to offer itself but received a return code of 500 or 505 (Secure NetEx/IP unavailable). "$$" is the number of seconds that it will wait before attempting the offer again
User Response: Cancel BFXJS, start Secure NetEx/IP, or do nothing - JS will continue to offer itself.

BFX314I BFXJS: Job Submitted.

Severity: 12 (Detailed information, but should be sent to the operator)
Module: BFXJSC in program BFXJS.
Explanation: The BFXJS program has submitted a job.
User Response: None

BFX315W Char Mode Record Truncated To 9999 Bytes.

Severity: 9 (Warning)
Module: BFXSBM in program BFXTI or BFXTR.
Explanation: A record greater than 9999 bytes in length was encountered during a char mode transfer. The record is truncated.
User Response: Transfer the file in BIT mode or allow the truncation to occur.

BFX316I sb rjobname rjobid rhost – "message from remote job status responder (BFXJSTAT)."

Severity: 9 (Highly visible)
Module: BFXCON in program BFXTI
Explanation: BFXTI was not connected to by BFXTR. The user specified the "RMTJOB" parameter in order to use the remote job status facility on the Z/OS remote host to determine the status of the BFXTR job. The status may be as follows:

sb = 0 Job not executing or queued for output.
sb = 1 Remote job is executing or queued for execution.
sb = 2 Could not Connect to 'Job Status Responder; no job status available.

The "rjobname" is the BFXTR jobname from the "RMTJOB" parameter (JOBSTAT if sb = 2). The "rjobid" is the JES jobid returned from B FXJSTAT. The "rhost" is the remote host name for the BFXTR job. The following messages may be returned by BFXJSTAT.

CURRENTLY ACTIVE	Job is running.
WAITING FOR EXECUTION	Job is waiting to run.
QUEUED FOR OUTPUT	Job has already run; possible JCL error.
HOLDING	Job is queued for execution.
ACTIVE IN NJE	Job is in the process of being transferred to another node.
STATUS UNKNOWN	JES returned unknown condition.
JOB NOT FOUND	Possible JCL error; probably already flushed.
DISASTROUS ERROR	JES is terminating for unknown reason. This should be shown to the systems programmer.
RCxxxx SCONNECT to BFXJSTAT FAILED	Interrogate the NRBSTAT in Rcxxxxx to determine the cause of the problem.

User Response: If the message indicates that the remote job will never connect to BFXTI, determine the cause of the problem, correct it and run the job again. Refer to "Messages Issued by the Job Status Function" on page 32 for more messages and descriptions.

BFX317W Offer failed; maximum sessions exceeded.

Severity: 1 (Diagnostic)
Module: BFXCON in BFXTI, BFXJS.
Explanation: A previously issued NETEX SOFFER (see messages BFX300I and BFX301I) failed because the Secure NetEx/IP maximum sessions limit had already been reached. Secure NetEx/IP will retry the OFFER a number of times at intervals determined during BFX generation or at run time.
User Response: None

BFX318W Connect failed; maximum sessions exceeded.
Severity: 1 (Diagnostic)
Module: BFXCON in BFXTI, BFXTR.
Explanation: A previously issued NETEX SCONNECT (see messages BFX302I and BFX303I) failed because the Secure NetEx/IP maximum sessions limit had already been reached. Secure NetEx/IP will retry the CONNECT a number of times at intervals determined during BFX generation or at run time.
User Response: None.

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

BFX320I $$
Severity: 9 (Informative)
Module: BFXPRM
Explanation: Echo control statement parameters
User Response: None.

BFX350I General Purpose Msg for Use by user written record modules.
Severity: 8 (Informative)
Module: ?
Explanation: User Defined 80 Character Msg area.
User Response: Indeterminate.

BFX351F C Environment could not be created.
Severity: 15 (Fatal)
Module: BFXTIC
Explanation: Could not create C environment for calls to C
User Response: Indeterminate.

BFX352I SNX log level set to x'$$'.
Severity: 8 (Informative)
Module: BFXCON
Explanation: Log level for Secure Netex set
User Response: Indeterminate.

BFX353I Host $$$$$$$$ excluded from secure transfer.
Severity: 8 (Informative)
Module: BFXCON

Explanation: Host is in excluded list
User Response: Indeterminate.

BFX354I Connection will be secured.
Severity: 8 (Informative)
Module: BFXCON
Explanation: Connection will be a secure connection
User Response: Indeterminate.

BFX355I Connection will be non-secured.
Severity: 8 (Informative)
Module: BFXCON
Explanation: Connection will be a non secure connection
User Response: Indeterminate.

BFX356E Invalid host/userid for job submission.
Severity: 10 (Error)
Module: BFXCON
Explanation: Secure job submission can't find host/userid in table
User Response: Indeterminate.

BFX357E No host/userid mapping for job submission.
Severity: 10 (Error)
Module: BFXCON
Explanation: Secure job submission has no host/userid mappings
User Response: Indeterminate.

BFX358E NOJSEC cannot be specified once JSEC has been set – will run secure.
Severity: 10 (Error)
Module: BFXPRM
Explanation:
User Response: Indeterminate.

BFX359E NODSEC cannot be specified once DSEC has been set – will run secure.
Severity: 10 (Error)
Module: BFXPRM
Explanation:
User Response: Indeterminate.

Lloyds Messages

TSB692I $$
Severity: 7 (Informative)
Module: BFXSTAPE and BFXRTAPE
Explanation: User Defined 80 Character Msg area for copy of UHL1
User Response: None.

TSB695I $$

Severity: 7 (Informative)
Module: BFXSTAPE and BFXRTAPE
Explanation: User Defined 80 Character Msg area for copy of UTL1
User Response: None.

TSB697I INFO ONLY – VOL1 DETECTED BY TAPESEND

Severity: 0 (Debug)
Module: BFXSTAPE
Explanation: Ensures a short block is transmitted. This allows the BFXRTAPE receive module to issue TSB698I if required.
User Response: None.

TSB698I INFO ONLY – VOL1 DETECTED BY TAPESEND

Severity: 7 (Informative)
Module: BFXRTAPE
Explanation: The volser received (FROM) will not be written in the tape label. The original volser (TO) will be written.
User Response: None.

TSB911I End of file $$: Transferred $$$$$$$$$ records

Severity: 7 (Informative)
Module: BFXRTAPE **and BFXSTAPE**
Explanation: Reports the total of all records on the tape – this will be an accumulation of the record counts in the 101 and 201 messages
User Response: None.

TSB444F OPEN OF TAPE FILE FOR INPUT FAILED

Severity: 15 (Fatal)
Module: BFXRTAPE
Explanation: Tried to open the first file for INPUT to get the VOLSER but the OPEN failed for some reason
User Response: None.

TSB555F RPARM VR but unable to find a volser

Severity: 15 (Fatal)
Module: BFXRTAPE
Explanation: RPARM (RP) VR (volser-retain) specified but no volser on tape or filedef/jcl
User Response: None.

TSB920I $$

Severity: 4 (ABEND)
Module: TSBCMR
Explanation: Job submit handled by tsbcmr – message for automation to pickup on
User Response: None.

Appendix D. H215L User Abend Codes

The following chart describes possible BFX abends.

Abend Code	Module Name	Cause of Abend	Action
0	BFXERM	Invalid conversion type.	This should not occur. Contact support at support@netex.com
U011	BFXSRM	Buffer overlap from QSAM	Collect generated dump. Contact support at support@netex.com
U050	BFXJSC	INTRDR open failure.	This should not occur. Contact support at support@netex.com
U102	BFXRBM/ SND	BFX protocol error; probably data corruption.	Contact support at support@netex.com
U103	BFXRBM	BFX protocol error; invalid sequence number.	Contact support at support@netex.com

Appendix E. Secure BFX Condition Codes

After execution of BFX, the user may check for condition codes. These condition codes indicate the status of the job. The user may construct the JCL to continue, restart, or terminate Secure BFX on the basis of the returned codes.

The condition codes are in the form of one or two-digit numbers and have the following meanings:

Code	Meaning
0	The job was completed.
4	The job was completed, but informational messages were issued. Access the messages in the BFX job log.
8	The job was completed, but warning messages were issued. Access the messages in the BFX job log.
12	A fatal error occurred, and the job was not completed. Consult the job log for causes.
16	A fatal error occurred, and the job was not completed. Usually the connection failed.
20	A fatal error occurred, and the job was not completed. Secure NetEx/IP may be unavailable, or the user may not be authorized to use Secure NetEx/IP.
24	A condition occurred (for example, a HALT SREF command was issued) which caused the BFX job to terminate.