

z/OS
Bulk Data Transfer



Overview

z/OS
Bulk Data Transfer



Overview

Note

Before using this information and the product it supports, be sure to read the general information under "Appendix C. Notices" on page 35.

First Edition (March 2001)

This edition applies to Version 1 Release 1 of z/OS Bulk Data Transfer, 5694-A01, and to all subsequent releases and modifications unless otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405

FAX (Other Countries):

Your International Access Code +1+845+432-9405

IBMLink (United States customers only): IBMUSM10(MHVRCFS)

Internet e-mail: mhvrcfs@us.ibm.com

World Wide Web: www.ibm.com/servers/eserver/zseries/zosqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1986, 2001. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	v
About This Book	vii
Who Should Read This Book	vii
Where to Find More Information	vii
Using LookAt to look up message explanations	vii
Accessing licensed books on the Web	vii
z/OS BDT Publications	ix
Chapter 1. Introduction to BDT	1
Introduction to the Base Feature	1
Introduction to the File-to-File Feature	1
BDT File-to-File Uses SNA	2
BDT File-to-File Copies Data Sets Directly from One Device to Another	2
BDT File-to-File Restarts Failing Copy Operations from a Checkpoint	2
BDT File-to-File Allows Users to Assign Scheduling Priorities	3
BDT File-to-File Runs in Its Own Address Space	3
BDT File-to-File Optionally Compresses Duplicate Data Strings	3
BDT File-to-File Helps Ensure Data Security	3
Introduction to the SNA NJE Feature	3
BDT SNA NJE Uses SNA	4
BDT SNA NJE Can Transmit and Receive Multiple Jobs Concurrently	4
BDT SNA NJE Allows Users to Assign Transmission Priorities to Jobs	4
BDT SNA NJE Can Coexist with JES3 BSC NJE	5
Chapter 2. The System Programmer's View of BDT	7
Coding BDT Initialization Statements	7
Separate File-to-File and SNA NJE Nodes	7
Global and Local Nodes (File-to-File Feature Only)	7
Automatic Session Startup	8
Automatic Session Restart	9
Session Passwords	9
Pacing	9
Number of Concurrent Data Transfers	10
Writing a BDT Start Procedure	11
Writing User Exit Routines	11
Defining BDT to MVS, VTAM, and JES3	12
Defining the Transaction Queuing Integrity (TQI) Facility	12
Creating a System GMJD Library (File-to-File Feature Only)	14
Defining a Poly-BDT Complex	14
Diagnosing Failures in the BDT Address Space	16
Initialization Stream Data Set	16
Storage Dumps	16
Log Data Set	17
Trace Table	17
Collecting System Management Facilities (SMF) Information	17
Chapter 3. The Operator's View of BDT	19
Starting the Transaction Queuing Integrity (TQI) Facility	19
Starting BDT	19
Starting Sessions with Other Nodes	19
Monitoring BDT	19
Changing the Status of a Job or BDT Resource	20

Chapter 4. The User's View of BDT with the File-to-File Feature	21
Submitting a Transaction	21
Example of a Transaction Submitted Interactively	21
Example of a Transaction Submitted in a Batch Job	21
Example of a Transaction Submitted via ISPF	22
Types of Data Sets That Can Be Transferred	22
Record Formats	23
Data Set Organization	23
Logical Record Lengths	23
Determining and Changing a Transaction's Status	23
A Simplified Way to Submit Frequently Used Transactions	24
Controlling the Sequence of Transaction Execution	24
Protecting Data Sets through RACF	25
Chapter 5. The User's View of BDT with the SNA NJE Feature	27
Chapter 6. Hardware and Software Requirements	29
Hardware Requirements	29
Software Requirements	29
Appendix A. Differences between File-to-File and SNA NJE Support	31
Poly-BDT Complexes	31
No Global-Local Relationships with SNA NJE	31
Virtual Logical Units (VLUs)	31
Use of the JES3 Spool	31
Data Compression	31
Data Checkpointing	31
Transaction Queuing Integrity (TQI) Facility	31
Initialization	31
User Exit Routines	32
Commands	32
No Dependent Transaction Control (DTC) Networks for SNA NJE	32
System Management Facilities (SMF) Type 59 Record	32
Appendix B. SNA NJE Flows with BDT	33
SNA NJE Job and Output Flows	33
SNA NJE Message and Command Flows	34
Appendix C. Notices	35
Trademarks	36
	37
Index	41

Figures

1. Example of a Configuration with the File-to-File Feature	2
2. Example of a Configuration with the SNA NJE Feature	4
3. Global-Local Relationship between Nodes	8
4. A Session between File-to-File Nodes Split into 255 Virtual Logical Units (VLUs)	10
5. A Session between SNA NJE Nodes Split into 29 Virtual Logical Units (VLUs)	11
6. An Allowable BDT Transaction Queuing Integrity (TQI) Configuration. Processor A has the BDT Base feature installed. Processor B has the BDT Base and File-to-File features installed.	13
7. Another Allowable BDT Transaction Queuing Integrity (TQI) Configuration. Processor A has the BDT Base feature installed. Processor B has the BDT Base and File-to-File features installed.	13
8. A BDT Transaction Queuing Integrity (TQI) Configuration That Is Not Allowed	14
9. A Poly-BDT Complex with BDT Address Spaces in the Same Processor. The processor has the BDT Base and File-to-File features installed.	15
10. A Poly-BDT Complex with BDT Address Spaces in Different Processors. Each processor has both the BDT Base and File-to-File features installed.	15
11. Poly-BDT Complexes with Separate Test and Production Systems.	16
12. A Simple Dependent Transaction Control (DTC) Network	24
13. A More Complex Dependent Transaction Control (DTC) Network	25
14. SNA NJE Job and SYSOUT Flows with BDT	33
15. SNA NJE Message and Command Flows with BDT	34

About This Book

This book introduces the Bulk Data Transfer (BDT) program product.

Who Should Read This Book

This book is for data processing managers and system programmers who are deciding whether to obtain BDT. It is also for system programmers, system operators, and end users who want to learn about BDT prior to using it.

To understand this book you should be familiar with Systems Network Architecture (SNA), the VTAM program product, and the z/OS JES2 or z/OS JES3 program product (depending on which job entry subsystem your installation uses).

Where to Find More Information

Where necessary, this book references information in other books using shortened versions of the book title.

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for z/OS messages and system abends.

Using LookAt to find information is faster than a conventional search because LookAt goes directly to the explanation.

LookAt can be accessed from the Internet or from a TSO command line.

You can use LookAt on the Internet at:

<http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html>

To use LookAt as a TSO command, LookAt must be installed on your host system. You can obtain the LookAt code for TSO from the LookAt Web site by clicking on **News and Help** or from the *z/OS Collection*, SK3T-4269.

To find a message explanation from a TSO command line, simply enter: **lookat** *message-id* as in the following example:

```
lookat iec192i
```

This results in direct access to the message explanation for message IEC192I.

To find a message explanation from the LookAt Web site, simply enter the message ID. You can select the release if needed.

Note: Some messages have information in more than one book. For example, IEC192I has routing and descriptor codes listed in *z/OS MVS Routing and Descriptor Codes*. For such messages, LookAt prompts you to choose which book to open.

Accessing licensed books on the Web

z/OS licensed documentation in PDF format is available on the Internet at the IBM Resource Link Web site at:

<http://www.ibm.com/servers/resourceLink>

Licensed books are available only to customers with a z/OS license. Access to these books requires an IBM Resource Link Web userid and password, and a key code. With your z/OS order you received a memo that includes this key code.

To obtain your IBM Resource Link Web userid and password log on to:

<http://www.ibm.com/servers/resourceLink>

To register for access to the z/OS licensed books:

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **User Profiles** located on the left-hand navigation bar.
3. Click on **Access Profile**.
4. Click on **Request Access to Licensed books**.
5. Supply your key code where requested and click on the **Submit** button.

If you supplied the correct key code you will receive confirmation that your request is being processed. After your request is processed you will receive an e-mail confirmation.

Note: You cannot access the z/OS licensed books unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

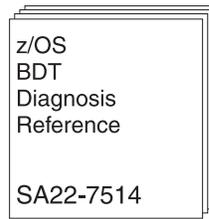
To access the licensed books:

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **Library**.
3. Click on **zSeries**.
4. Click on **Software**.
5. Click on **z/OS**.
6. Access the licensed book by selecting the appropriate element.

z/OS BDT Publications



Introduces BDT.



This book explains how to request and read a BDT formatted dump, explains how to find and read BDT information in an MVS SVS dump, and shows the layouts of BDT data areas.



Explains how to create and submit transactions to copy data sets.



Explains how to define BDT to MVS and ACF/VTAM, allocate BDT and TQI data sets, write BDT initialization statements, write BDT and TQI start procedures, and write BDT user exit routines.



Explains all BDT messages and completion codes.



Describes all BDT commands.

Chapter 1. Introduction to BDT

This chapter provides a brief, high-level introduction to Bulk Data Transfer (BDT).

BDT consists of three features:

- The Base feature, which contains support for the other two features and which provides the transaction queuing integrity (TQI) facility. TQI prevents the loss of file-to-file *transactions*, which are requests to copy data sets, and routes messages from BDT to users.
- The File-to-File feature, which lets OS/390 JES2 and OS/390 JES3 users copy data sets from one computer system to another within a SNA network.
- The SNA Network Job Entry (NJE) feature, which allows OS/390 JES3 users to transmit jobs, output (SYSOUT), commands, and messages from one computer system to another within a SNA network.

The rest of this chapter introduces these three features.

Introduction to the Base Feature

The Base feature of BDT has two purposes:

- It is a prerequisite to the File-to-File feature and the SNA NJE feature. That is, if a processor is to have either of those features, or both, it must also have the Base feature.
- It provides the transaction queuing integrity (TQI) facility. TQI is discussed below.

TQI is provided primarily for users of the File-to-File feature. It ensures that the transactions they submit to copy data sets reach the BDT work queue. Once on the queue, the transactions are scheduled for processing, which results in the copying of the actual data sets.

In order to reach the work queue, a transaction always travels from one address space to another (the user's to TQI's to BDT's) and may travel from one processor to another. Without adequate protection, an error such as a hardware failure could prevent a transaction from being scheduled for processing.

The way that TQI protects a transaction is to record it on the *TQI checkpoint data set* at the processor of origin. Should the transaction fail to reach the BDT address space, BDT recovers it from the TQI checkpoint data set, without user intervention, and attempts the transfer again. Once the transfer reaches the BDT address space, BDT puts it on the work queue.

The secondary purpose of TQI is to ensure that commands and BDT messages are handled properly for both File-to-File users and SNA NJE users. TQI checkpoints commands just like it does transactions to ensure that they reach the BDT work queue. As for BDT messages, TQI makes it possible for users to receive them.

The use of TQI is optional but recommended. Without it, commands and file-to-file transactions that users submit could be lost, and users would not receive messages from BDT.

Introduction to the File-to-File Feature

BDT with the File-to-File feature provides file transfer capability for JES2 and JES3 installations that are part of a SNA computer network. This means that users of a JES2 or JES3 processor in one complex can copy data sets to users of a JES2 or JES3 processor in another complex. (BDT must be installed in both complexes.) Figure 1 on page 2 illustrates this capability.

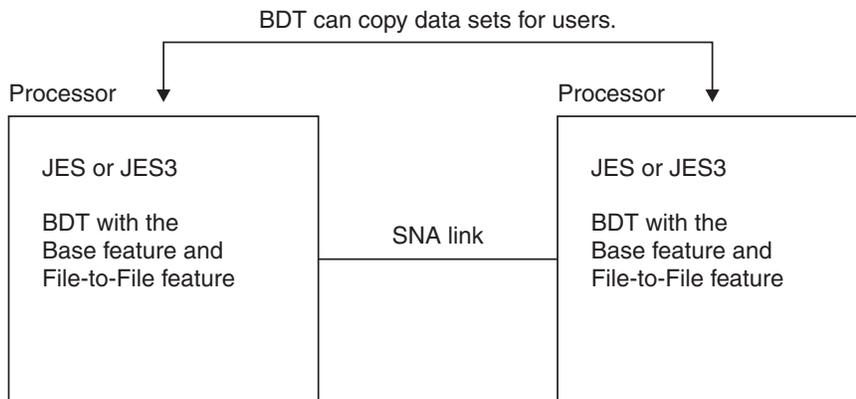


Figure 1. Example of a Configuration with the File-to-File Feature

BDT can also copy data sets from one processor to another within the same complex.

BDT can copy sequential data sets, partitioned data sets, or selected members of a partitioned data set.

To copy a data set, a user submits a transaction to BDT. The transaction may be submitted from a TSO terminal, an MCS console (by the MVS or JES2 operator), or a JES3 console (by the JES3 operator). It may be submitted at either processor that will take part in the copy operation. The transaction may be submitted interactively (like a command), in a batch job, or via an ISPF panel (from a TSO terminal only).

A transaction may be submitted at either processor that will take part in the copy operation. For example, a user at complex A could submit a transaction to copy a data set from complex A to complex B, or to copy a data set from complex B to complex A.

Once BDT has assigned a job number to the transaction, the user can issue BDT commands to inquire about or alter its processing.

BDT issues messages to report on the progress and completion of each transaction.

BDT File-to-File Uses SNA

BDT uses IBM's Systems Network Architecture (SNA) and runs as a full duplex VTAM application program. (Full duplex operation allows data to be sent and received at the same time.) There are a number of advantages to using SNA and VTAM. For example, it is not necessary to install separate links for different kinds of terminals from which users submit transactions; different kinds of terminals can be attached to the same link.

BDT File-to-File Copies Data Sets Directly from One Device to Another

When BDT copies a data set it writes the data set directly to another data set. In this way BDT avoids using the JES spool. It is advantageous for BDT to avoid the spool because it is a heavily used data set with a limited amount of space. Also, this design results in BDT using fewer machine cycles. If BDT were to use the spool, contention for space with other programs could degrade the performance of BDT. Also, large data sets that BDT wrote to the spool would tie up considerable spool space at the expense of other programs.

BDT File-to-File Restarts Failing Copy Operations from a Checkpoint

If a hardware or software error occurs while BDT is copying a data set, BDT does not have to recopy the entire data set. A checkpoint file enables BDT to automatically resume failing copy operations from the last checkpoint. (Do not confuse this checkpointing of the actual data being copied with the checkpointing of the request to copy the data, which is provided by TQI and was discussed on page 1.) For a partitioned

data set, BDT automatically updates the checkpoint file after each member has been successfully copied. For sequential data sets, BDT updates the checkpoint file at an installation-defined frequency.

BDT File-to-File Allows Users to Assign Scheduling Priorities

Some transactions are more important than others and should be scheduled to run ahead of the less important ones. For this reason BDT allows users to assign scheduling priorities when they submit transactions.

BDT File-to-File Runs in Its Own Address Space

BDT runs in its own address space as an MVS subsystem. It does not share an address space with another subsystem, such as JES2 or JES3. BDT is therefore isolated from failures in other subsystem address spaces, and it does not limit the virtual storage available to other subsystems.

BDT File-to-File Optionally Compresses Duplicate Data Strings

BDT provides two types of data compression. One type compresses into one byte any string of blanks that is two or more characters in length, and compresses into two bytes any string of duplicate nonblank characters that is three or more characters in length. The other type compresses into two bytes any string of duplicate characters (blank or nonblank) that is three or more characters in length. Data compression is optional. Users may select the type of data compression they wish to use when they submit transactions.

BDT File-to-File Helps Ensure Data Security

The system programmer must decide what security measures are necessary to protect both BDT and the installation's data from unauthorized access. Without adequate security measures, unauthorized persons could establish sessions, copy or destroy data to which they should not have access, or issue commands that adversely affect BDT. BDT provides several ways to help ensure data security:

- **Session passwords.** To prevent unauthorized sessions from being established, the system programmer should assign passwords for each node. Thereafter, only a node that provides the correct passwords will be able to establish a session with the password-protected node.
- **Exit routines.** The system programmer can write exit routines to check each transaction or BDT command before BDT executes it, to ensure that the requester is properly authorized. If a user exit routine detects a request that would result in a security violation, the routine can disallow the request. The exit routine could also record the potential violation in an installation-defined data set.
- **RACF-protected data sets.**

When both nodes in a session include the Resource Access Control Facility (RACF) program product, users of that session may RACF-protect data sets they create. Users request RACF protection by coding (or defaulting) security parameters on their copy requests. These parameters include the user ID, group ID, and password under which RACF is to protect the data set.

The system programmer may write exit routines that examine or modify the user's security parameters. BDT uses the system authorization facility (SAF) interface of MVS to request RACF functions. This interface provides an exit to an installation-written exit routine. BDT also provides an exit to one of its own installation-written exit routines. Both exits are invoked before RACF is invoked and both have access to and may modify the parameter list that will be passed to RACF.

RACF provides password encryption. This is important because BDT transfers RACF passwords from one node to another.

Introduction to the SNA NJE Feature

BDT with the SNA NJE feature provides job networking capability for JES3 installations that are part of a SNA computer network. This means that a user at one computer system in a SNA network can:

- Send a job to run on another system in the network
- Send output (SYSOUT) to be printed at another system in the network
- Send messages and commands to another system in the network.

Any of the following systems can participate in the network:

- OS/390 JES3
- OS/390 JES2
- VM/SP (with RSCS Networking)
- VSE/Advanced Functions (with VSE/POWER).

Users use job control language (JCL) and JES3 control statements to submit jobs to JES3. JES3 passes the jobs to BDT, which enters them into the network. If necessary, the network jobs can be routed through intermediatesystems (store and forward) to reach their destination.

Previously, JES3 networking was available using BSC communication links only. Now, with BDT added to a JES3 system, JES3 networking is possible using SNA communication links.

Figure 2 illustrates the function of the SNA NJE feature.

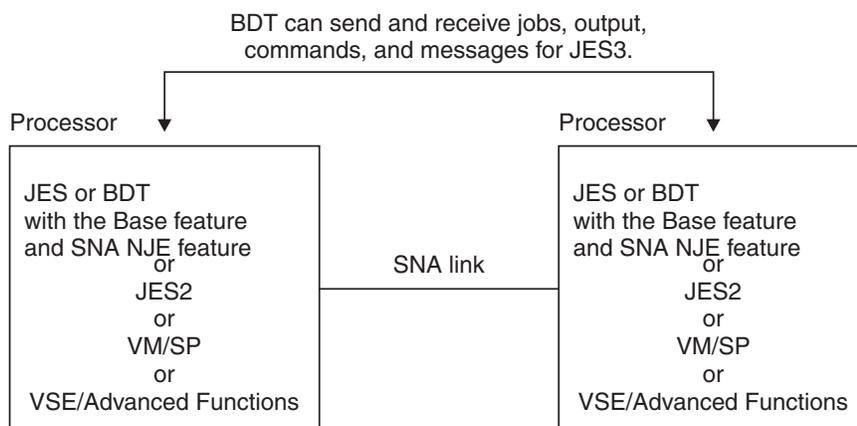


Figure 2. Example of a Configuration with the SNA NJE Feature

BDT SNA NJE Uses SNA

Because BDT uses IBM's Systems Network Architecture (SNA), JES3 job networks can benefit from the many features of SNA that are not available in a BSC network. For example, in a SNA job network, VTAM ensures that transmissions are complete (no data is lost) and that all parts of a transmission are received in the correct order. In a BSC job network the installation must code application programs to ensure this data integrity.

BDT SNA NJE Can Transmit and Receive Multiple Jobs Concurrently

Any two network points between which BDT transfers network jobs are logically connected by sessions. Each session can carry up to 28 SNA NJE transmission streams concurrently. Contrast this with transmission in a BSC network, in which only one stream at a time is possible.

Of the 28 possible concurrent transmission streams in a session, one node can schedule 14 (7 for sending jobs and 7 for sending output) and the other node can schedule 14 (7 for sending jobs and 7 for sending output).

BDT SNA NJE Allows Users to Assign Transmission Priorities to Jobs

With BSC NJE (that is, without BDT), a user can code the PRTY parameter on the MVS JOB statement to assign a priority by which the system will select a job for execution. But it is not possible to assign a priority by which the system will transmit output from the job when it finishes execution; jobs have their output transmitted in the order in which they finish execution.

With SNA NJE (that is, with BDT), the PRTY parameter on the MVS job statement applies to both the scheduling of the job and the transmission of the output produced by the job. This provides installations greater control of their work flow.

BDT SNA NJE Can Coexist with JES3 BSC NJE

The JES3 SNA NJE support through BDT and the JES3 BSC NJE support (through JES3) can coexist in a JES3 complex. This allows a JES3 complex to gradually migrate from BSC NJE to SNA NJE. If the system programmer assigns the same name to the SNA NJE node as to the BSC NJE node, users who submit jobs will not be affected; they can continue to use their job control language (JCL) and JES3 control statements, without change, to submit their work.

Chapter 2. The System Programmer's View of BDT

This chapter discusses tasks that a system programmer must perform during and after installation of BDT. This chapter discusses:

- Coding BDT initialization statements
- Writing a BDT start procedure
- Writing user exit routines that tailor BDT to fit specific needs of the complex
- Defining BDT to MVS, VTAM, and JES3
- Defining the BDT transaction queuing integrity (TQI) facility
- Creating a system generic master job definition (GMJD) library (File-to-File feature only)
- Defining a poly-BDT complex
- Diagnosing failures in the BDT address space
- Collecting system management facilities (SMF) information.

Coding BDT Initialization Statements

To define the environment in which BDT will operate each time it is started, the system programmer must code initialization statements. The system programmer can specify the amount of main storage BDT will use, the names of nodes in the network, the pacing rate for communication between nodes, and many more operational characteristics. BDT initialization statements are similar in function to JES2 and JES3 initialization statements. BDT reads these initialization statements during each warm or cold start of BDT.

The system programmer can code initialization statements from scratch or modify the initialization statements provided in SYS1.SBDTSAMP.

Some of the more significant operational characteristics that can be specified on initialization statements are discussed below.

Separate File-to-File and SNA NJE Nodes

The system programmer must define the node or nodes that will exist at his complex. In addition, the system programmer must define other nodes in the network that will communicate with his node or nodes.

A BDT node is the point in a BDT address space that is linked to another BDT address space for either of two types of data transfer: file-to-file (requiring the File-to-File feature) or SNA NJE (requiring the SNA NJE feature). Thus, an address space can contain one or two nodes, depending on whether it is used for file-to-file communication only, for SNA NJE communication only, or for both.

Each node must have a name. Users and operators use the name to identify the node to which their transfer requests or commands are directed. BDT also uses the node name in messages to identify the node to which the message pertains.

Global and Local Nodes (File-to-File Feature Only)

Within a JES complex, each file-to-file node has a global or local relationship with every other file-to-file node with which it communicates.¹ One node must be defined as the *global node* and the other node as the *local node*. This relationship determines which node schedules transactions to copy data sets between the two nodes, regardless of the node at which the transactions are submitted. The global node always schedules transactions. The global node, therefore, has a BDT address space with a work queue.

1. The terms *global* and *local*, as used here, refer to a relationship between nodes and not to the JES3 global or local processors.

A given node need not have the same global-local relationship with all other nodes. In other words, in one relationship the given node may be the local and in another relationship it may be the global. Figure 3 shows how a node can have one relationship with one node and a different relationship with another node.

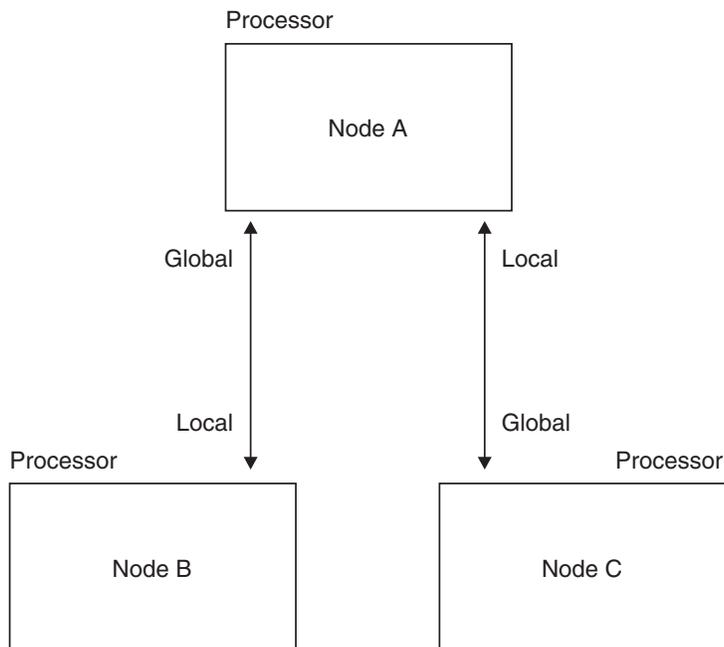


Figure 3. Global-Local Relationship between Nodes

In this figure, node A is global with respect to node B and local with respect to node C. If a user were to submit a transaction to copy data from node A to node B, BDT would schedule the transaction on node A because it is global with respect to node B. It would make no difference from which node, A or B, the user submitted the transaction. If the user submitted a transaction to copy data from node A to node C, BDT would schedule that transaction on node C because node C is global with respect to node A.

Note that, in this example, a user could not copy data from node B to node C, or node C to node B. Also note that if the user at node A wanted to check the status of the job at node C, the user would have to send a command to node C to do so; the user could not issue the command from node A.

The system programmer must define the global-local relationship that exists between his node and other nodes in the network. The system programmers at the other nodes must do the same. Between any given pair of nodes, the relationship defined at one node must agree with the relationship defined at another node. Otherwise the two nodes will be unable to establish a session. For example, if the system programmer at node A defines node A as global with respect to node B, the system programmer at node B must also define node A as global with respect to node B.

Automatic Session Startup

BDT runs as a full duplex VTAM application program. Before BDT can transfer data sets between two file-to-file nodes, or transfer jobs or output between two SNA NJE nodes, a session must exist between the nodes. A session is the logical connection between VTAM application programs. VTAM manages the session; BDT uses the session.

Before VTAM can establish a session between nodes, BDT must be started at both nodes, one of the nodes must request the session, and the two nodes must agree on certain communication parameters. Between a given pair of nodes, BDT allows only one session at a time to be active. Both of those nodes, however, may have sessions active at the same time with as many as 100 other nodes.

The system programmer can request, through BDT initialization statements, a session with one or more nodes. BDT automatically starts these sessions during BDT initialization. There are also BDT commands that enable the operator to start or stop sessions.

VTAM will establish a session between two file-to-file nodes only after the nodes agree on the following communication parameters:

- Session passwords
- The global-local relationship
- The size of data buffers the nodes will use when transferring data
- The maximum number of transactions that can transfer data concurrently
- The data compression option.

The file-to-file nodes must agree exactly on their session passwords, global-local relationship, and data compression option. The other parameters may be negotiated between the nodes. That is, if one node provides a smaller value for a parameter than the other node, the smaller value is used as long as it is acceptable to the other node.

VTAM will establish a session between two SNA NJE nodes only after the nodes agree on the following communication parameters:

- Session passwords
- The size of data buffers the nodes will use when transferring data.

The SNA NJE nodes must agree exactly on their session passwords. The size of data buffers may be negotiated between the nodes.

Automatic Session Restart

Sometimes a system failure may abnormally terminate a session. Although the operator can restart an abnormally terminated session, the operator may be unaware, for some time, that a session has terminated.

To avoid delays in restarting abnormally terminated sessions, the system programmer can select the automatic restart option. This option enables BDT to restart abnormally terminated sessions automatically. When selecting the option the system programmer can also specify how many times or how frequently BDT is to try automatic restart.

Session Passwords

The system programmer should define session passwords for each session with another node. System programmers at other nodes that wish to establish a session with the password-protected node must define the same passwords. Session passwords prevent a node that has not defined the proper passwords from establishing a session with the password-protected node.

Pacing

Each time one node sends data to another node, the node receiving the data must either process it immediately or store it temporarily for processing later. Without some means of protection the receiving node could be overrun by data from the sending node. Overrun occurs when the receiving node receives data faster than it can process it and when the receiving node runs out of temporary storage for the data.

To prevent data overrun BDT uses a technique called pacing. Pacing enables the receiving node to regulate the rate at which the sending node sends it data. When the two nodes establish a session with each other they agree on a pacing rate.

The pacing rate defines the number of data buffers that the sending node can send before the receiving node must return an acknowledgment. The acknowledgment is a signal that the receiving node is ready to

receive more buffers. If the receiving node fails to send an acknowledgment after it receives the agreed-to number of buffers, the sending node stops sending buffers. The sending node resumes sending buffers only after it receives the required acknowledgment.

BDT with the File-to-File feature uses application-level pacing, which means that BDT manages the pacing rate; VTAM pacing is disabled. BDT with the SNA NJE feature uses VTAM pacing.

The system programmer is responsible for defining the pacing rate.

Number of Concurrent Data Transfers

Each node can transfer data concurrently for multiple transfer operations. That is, at any time a node can take part in multiple sessions and each session can support multiple data transfers. When multiple data transfers occur over one session, BDT multileaves the transmitted data. Multileaving ensures that a large data transfer does not monopolize the session. The BDT system programmer defines the maximum number of concurrent data transfers in which a node may take part.

Virtual Logical Units (VLUs)

To achieve concurrent data transfer, BDT multileaves, over a single session, the data for each active transfer by splitting the session into “subsessions” known as *virtual logical units (VLUs)*. A VLU is a logical path between two nodes that represents one user of a session between BDT nodes. Each active transfer is considered to be a user of one VLU.

For each session, BDT always requires one VLU for communicating control information. The other VLUs transfer the data.

For each file-to-file session, 254 data transfer VLUs are possible (that is, BDT can concurrently transfer the data for up to 254 requests). The sending node can schedule any number of them and the receiving node can schedule any number, up to the limit of 254. This is illustrated in Figure 4.

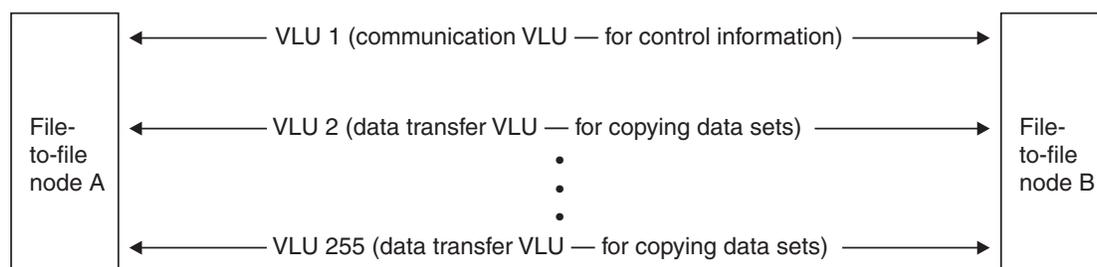


Figure 4. A Session between File-to-File Nodes Split into 255 Virtual Logical Units (VLUs)

For each JES3 SNA NJE session, 28 data transfer VLUs are possible. Of the 28 possible VLUs in a session, one node can schedule 14 (7 for sending jobs and 7 for sending output) and the other node can schedule 14 (7 for sending jobs and 7 for sending output). They are specified in groups of four, as illustrated in Figure 5 on page 11.

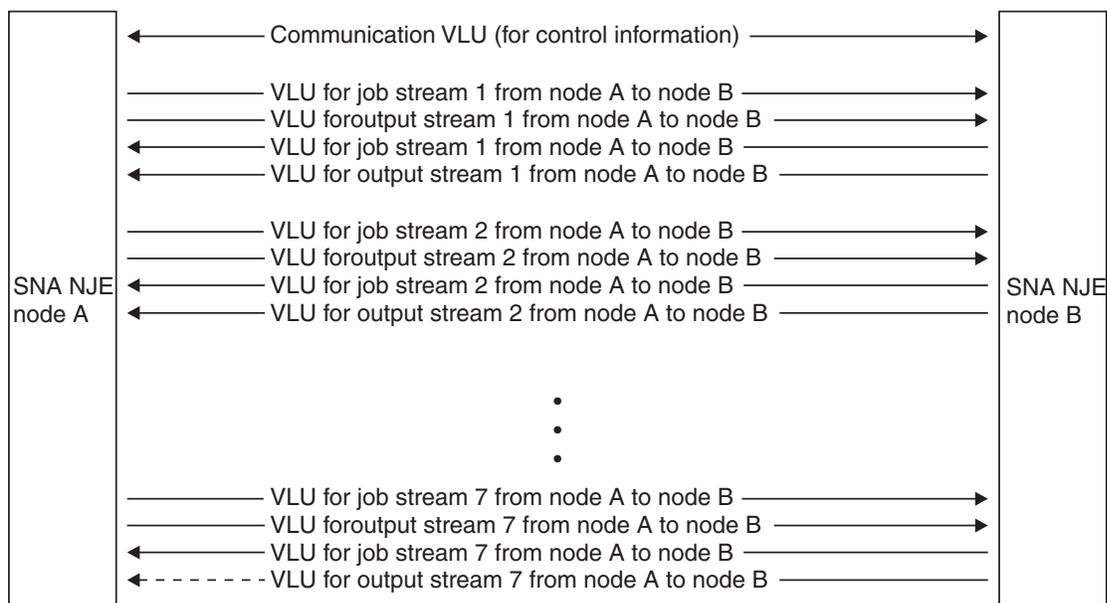


Figure 5. A Session between SNA NJE Nodes Split into 29 Virtual Logical Units (VLUs)

Fenced VLUs (File-to-File Feature Only)

A VLU can transfer data in two directions—either to or from a node. During the time a VLU is allocated to a specific transfer, however, that VLU can transfer data in only one direction. Once that transfer completes and BDT deallocates the VLU, the VLU will again be able to transfer data in either direction.

It is possible for file-to-file transfers traveling in the same direction to tie up all available VLUs. If a transaction to copy data in the other direction is then submitted, it would have to wait for an active transfer to complete, which would free a VLU.

To ensure that all file-to-file VLUs are not tied up in one direction, an installation can *fence* some number of file-to-file VLUs. Fencing a number of VLUs in a particular direction dedicates those VLUs for transfer in one direction only. Either an initialization statement or a BDT command can be used to specify how many VLUs are to be fenced in each direction. Unfenced VLUs are still available to transfer data in either direction.

Writing a BDT Start Procedure

The system programmer must provide a start procedure that the operator can use to start BDT. The procedure must contain JCL statements to start the BDT address space, to identify the data sets BDT is to use, and to identify the data set containing the initialization statements that will establish BDT's operating environment. The system programmer can code the start procedure from scratch or modify the sample start procedure provided in SYS1.SBDTSAMP.

Writing User Exit Routines

BDT provides a number of user exits. A user exit is a place in BDT code where control is passed to a user-written exit routine. User exit routines can provide security for an installation or perform other installation-defined processing. For example, user exit routines can:

- Authorize users to access data sets that are to be copied
- Authorize users to submit transactions or commands
- Enforce installation-defined default values for BDT transactions or commands
- Inspect, modify, or reroute BDT messages

- Process installation-defined initialization statements or parameters
- Gather and process transaction accounting information.

The system programmer must write the authorization-type exit routines. They are required. There are seven of them. IBM provides samples in SYS1.SBDTSAMP. The other exit routines, which do not deal with authorization, are optional and need not be coded. There are 18 of them.

User exit routines run as an extension of BDT. Therefore, they run in supervisor state under the storage protection key that BDT is using at the time the exit routine receives control. The exit routines must be stored in an authorized program facility (APF) library.

Defining BDT to MVS, VTAM, and JES3

The system programmer must define BDT to MVS as a secondary subsystem by creating an entry in an IEFSSNxx member of SYS1.PARMLIB. If BDT is to participate in a global resource serialization (GRS) complex, the system programmer must specify the GRS parameter on the IEASYSxx entry in SYS1.PARMLIB.

The VTAM system programmer must make several VTAM definitions. The VTAM programmer must define session parameters to VTAM by creating logon mode table entries. The VTAM programmer must define BDT as a VTAM application program on the APPL definition statement. And the VTAM programmer must define each remote node as an VTAM cross-domain resource by coding a CDRSC definition statement for each remote node.

If BDT will be installed in a JES3 complex, the JES3 system programmer will have to code BDT-related information on several JES3 initialization statements: CONSOLE, SYSID, and, if the SNA NJE feature is to be used, NJERMT.

Defining the Transaction Queuing Integrity (TQI) Facility

The transaction queuing integrity (TQI) facility has two functions:

- It prevents the loss of commands and file-to-file transactions that users submit. When a user submits such a request it is recorded on the TQI checkpoint data set in addition to being sent to the BDT work queue to await execution. If a request is lost before it reaches the work queue, BDT recovers it from the TQI checkpoint data set and again attempts to put it on the work queue.
- It routes BDT messages to users. Because message traffic can become heavy, BDT stores messages in one or more message data sets from which TQI periodically gets them to send to users.

TQI runs in its own address space, not in the BDT address space.

The TQI checkpoint data set must be allocated on a direct access device that is attached to the processor from which transactions are submitted and the processor that has the BDT address space.

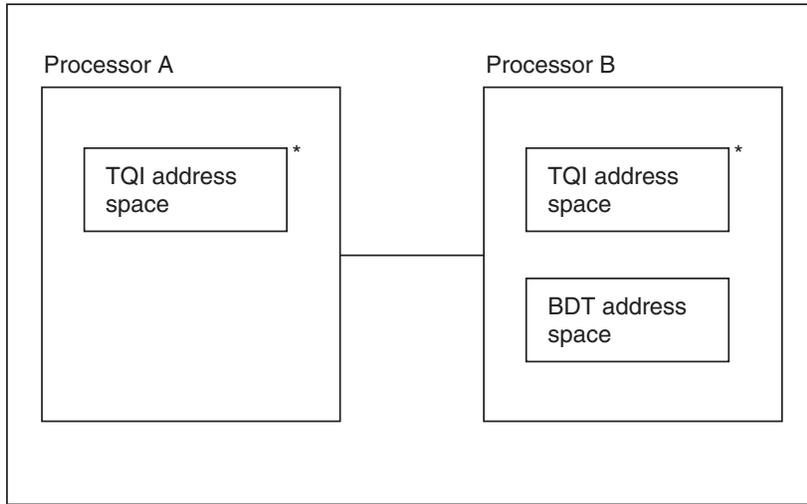
TQI is optional but recommended. Without it user commands and file-to-file transactions could be lost. In addition, messages from BDT would not be sent to users. (Although BDT would write the messages to the log data set, users do not have access to this data set.)

In order to use TQI the system programmer must allocate and format the TQI data sets, define two TQI options to MVS (as parameters in the IEFSSNxx member of SYS1.PARMLIB), and write a start procedure that the operator will use to start TQI (and thus create a TQI address space).

Within a complex, a TQI address space should exist in any processor at which users will enter commands and file-to-file transactions, and must exist in any processor at which users are to receive BDT messages. It is not necessary, however, to have a BDT address space in the same processor as the TQI address

space. Figure 6 and Figure 7 illustrate some allowable configurations of TQI and BDT address spaces. Figure 8 on page 14 illustrates a TQI configuration that is not allowed.

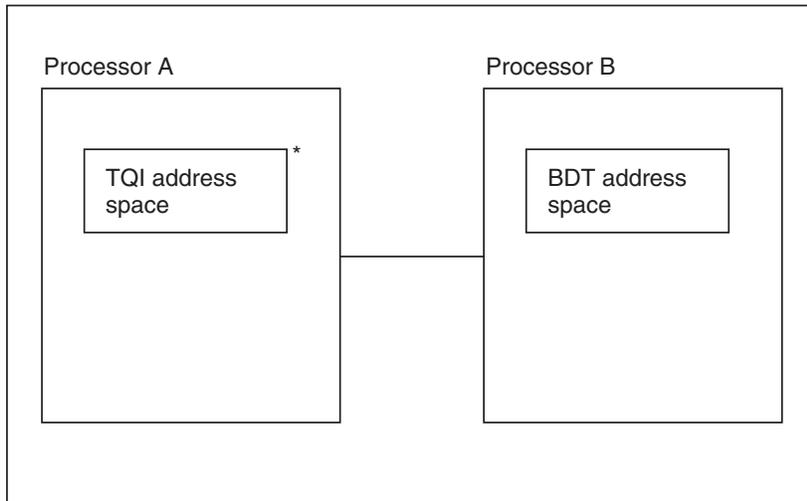
JES complex



* Users can enter commands and file-to-file transactions, and receive messages.

Figure 6. An Allowable BDT Transaction Queuing Integrity (TQI) Configuration. Processor A has the BDT Base feature installed. Processor B has the BDT Base and File-to-File features installed.

JES complex



* Users can enter commands and file-to-file transactions, and receive messages.

Figure 7. Another Allowable BDT Transaction Queuing Integrity (TQI) Configuration. Processor A has the BDT Base feature installed. Processor B has the BDT Base and File-to-File features installed.

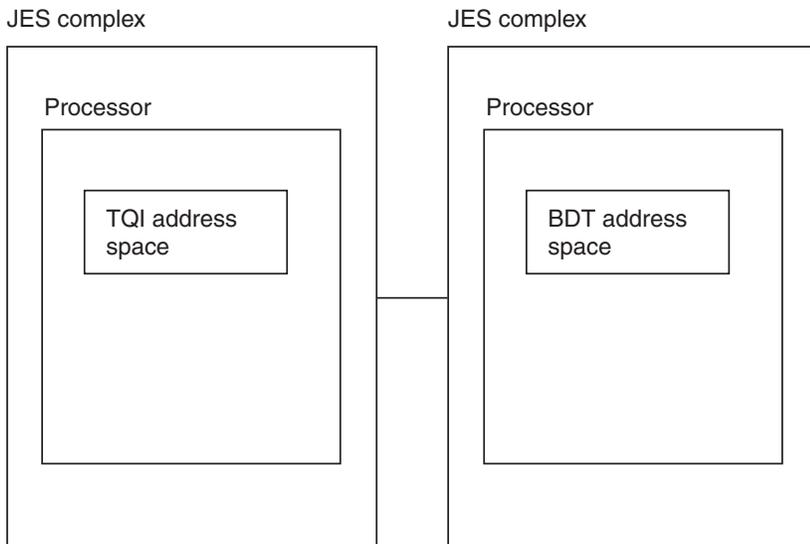


Figure 8. A BDT Transaction Queuing Integrity (TQI) Configuration That Is Not Allowed

Creating a System GMJD Library (File-to-File Feature Only)

A *generic master job definition (GMJD) library* is a data set that contains precoded file-to-file transactions. A GMJD library allows frequently used transactions to be coded just once, stored in the library, and used over and over. Each user can have a private GMJD library and there can be a system GMJD library available to all users.

GMJD libraries are optional. A user who wants to use a private GMJD library must create one. If there is to be a system GMJD library, the system programmer must create it.

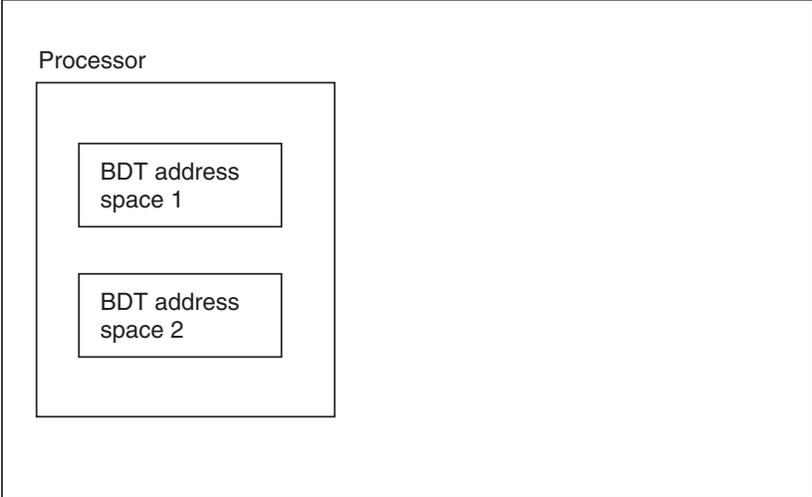
In addition to saving users the time it would take to code transactions of their own, a system GMJD library can provide a way to standardize transactions.

See “A Simplified Way to Submit Frequently Used Transactions” on page 24 for information about the use of GMJD libraries.

Defining a Poly-BDT Complex

BDT runs in its own address space. A JES complex may have one or more BDT address spaces. Each processor in the complex may have one or more BDT address spaces. A complex with multiple BDT address spaces is called a *poly-BDT complex*. Figure 9 on page 15 and Figure 10 on page 15 show examples of poly-BDT complexes.

JES complex



* Users can enter commands and file-to-file transactions, and receive messages.

Figure 9. A Poly-BDT Complex with BDT Address Spaces in the Same Processor. The processor has the BDT Base and File-to-File features installed.

JES complex

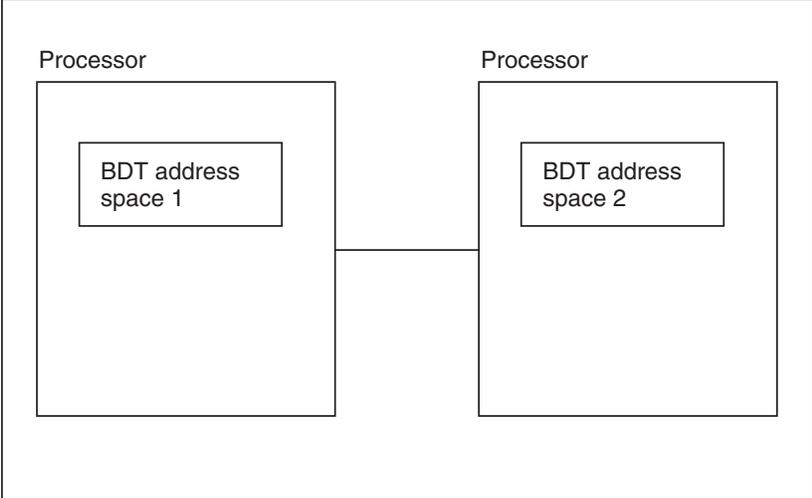


Figure 10. A Poly-BDT Complex with BDT Address Spaces in Different Processors. Each processor has both the BDT Base and File-to-File features installed.

A good use for poly-BDT complexes is to separate test work from production work. Figure 11 on page 16 illustrates such a situation. In the figure, complexes A and C are poly-BDT complexes. They each have an address space for daily production work and another address space for testing (such as testing newly written user exit routines) and for training operators and users. By separating testing and training from the production work, a complex reduces the risk of disrupting its production work.

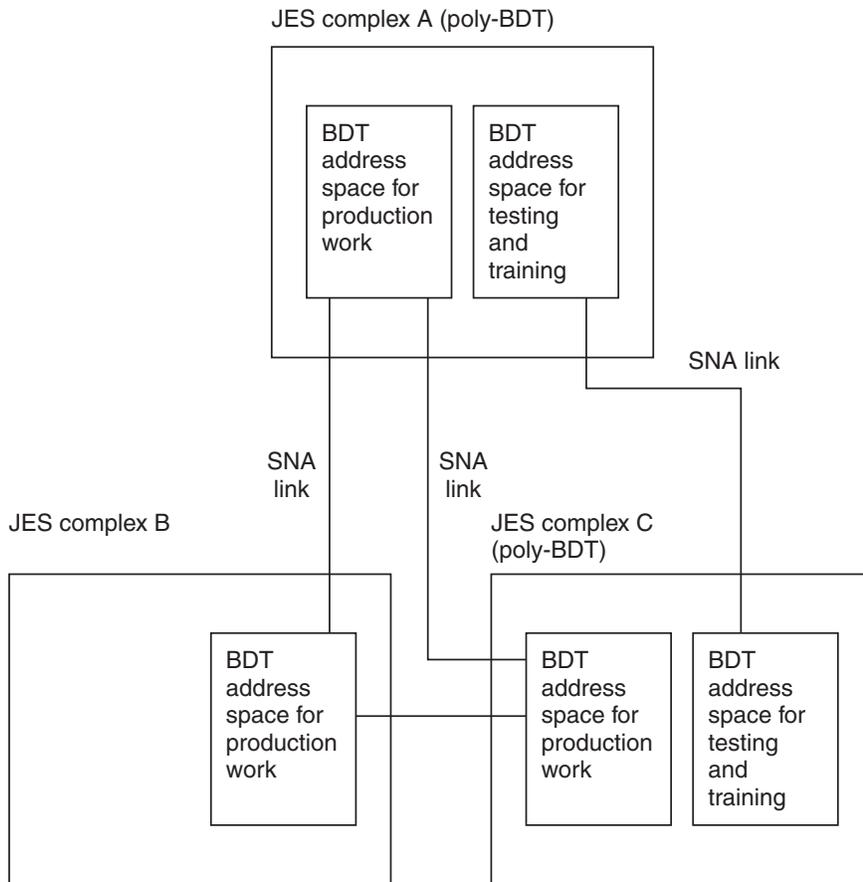


Figure 11. Poly-BDT Complexes with Separate Test and Production Systems.

The system programmer must decide how many BDT address spaces to have in a complex and on which processor each should reside. The system programmer must code a separate set of initialization statements for each address space.

Diagnosing Failures in the BDT Address Space

The system programmer is responsible for analyzing and isolating failures in the BDT address space. If a failure is caused by incorrect initialization statements or by a faulty user-written exit routine, the system programmer must fix it. If a failure is caused by BDT, the system programmer should report it to IBM.

Several debugging tools can help determine the cause of a failure in the BDT address space.

Initialization Stream Data Set

If BDT detects an error in an initialization statement it records the error in a data set. The BDTOUT DD statement of the BDT start procedure identifies the data set.

Storage Dumps

Anytime the BDT address space terminates abnormally, BDT optionally dumps the address space. The system programmer or the operator may request either a formatted dump or an unformatted dump. The system programmer selects the data set to which the formatted dump will be written. The unformatted dump is always written to the SYS1.DUMP data set. The system programmer may then use the AMDPRDMP service aid to format the dump.

The unformatted dump is most useful for complexes that have their error analysis done at another location. If that location is defined as a BDT file-to-file node, the system programmer can submit a BDT transaction to transfer the dump to the other location. Personnel at the other location can then use the AMDPRDMP service aid to format the dump.

Log Data Set

The BDT log data set provides another debugging tool. BDT records all transactions and commands submitted, and all messages issued, in the log data set. This information provides a history of the events that led to the problem.

Trace Table

The BDT trace table provides a chronological history of significant program events that occur within BDT. The data contained in the trace table will vary from entry to entry. For example, an entry might contain:

- The name of the module that caused the entry
- A unique identifier that allows you to distinguish one entry from another
- The contents of important registers
- The TCB address at the time the entry was made
- The time of day the entry was made.

The trace table consists of a number of 64-byte entries. The system programmer defines the number of entries that the table may contain. The table itself is a wraparound table. That is, when BDT has used the last entry in the table, it wraps around to the first table entry and starts over.

Besides tracing BDT events, you may also trace program events that occur in your exit routines. IBM provides a macro that you may use in your exit routines to request an entry in the trace table. The macro also allows you to specify the information that is to be recorded in the entry.

Collecting System Management Facilities (SMF) Information

BDT writes a type 59 SMF record to the SMF data set each time it completes a file-to-file transfer (of a data set) or a JES3 SNA NJE transfer (of a job or SYSOUT). However, BDT does not write a type 59 record for data received, or for commands or messages that an operator sends to another JES3 SNA NJE node.

The information that BDT writes to the SMF data set includes:

- The ID of the user who submitted the transfer
- The name of each node that took part in the transfer
- The name of the node where BDT scheduled the transfer
- A count of the number of bytes of data transferred
- The name of the “from” and the “to” data sets (file-to-file transfers only)
- The original job number and job name and the network account number (JES3 SNA NJE transfers only).

Before BDT writes the type 59 record, a user-written exit routine may receive control. This routine can examine, change, or replace user-defined fields in the record to be written.

Chapter 3. The Operator's View of BDT

This chapter discusses tasks that a system operator is likely to perform to ensure that BDT is available for users. This chapter discusses:

- Starting the transaction queuing integrity (TQI) facility
- Starting BDT
- Starting sessions with other nodes
- Monitoring and changing the status of jobs or the status of BDT resources.

Starting the Transaction Queuing Integrity (TQI) Facility

BDT TQI runs in its own address space as an MVS subsystem. TQI should be started on each processor through which users may submit commands and file-to-file transactions, and must be started on each processor at which users are to receive BDT messages. The system operator is responsible for starting and stopping TQI.

Starting BDT

BDT runs as an MVS subsystem and may be started automatically as part of the MVS IPL procedure. Or the operator may start BDT by issuing the MVS START command. Three types of starts are possible: cold start, hot start, and warm start. Basically, the three types differ in how BDT initializes its address space and whether BDT reformats the work queue.

On a cold start, BDT reads the initialization statements and uses the information to initialize the address space. BDT also reinitializes its work queue. When this happens, any file-to-file jobs that were on the queue are lost and must be resubmitted. Any JES3 SNA NJE jobs that were on the queue, however, are not lost; they are resent from JES3 and put back on BDT's queue.

On a hot start, BDT does not read the initialization stream. Instead, it initializes the address space by using the same values it used the last time it was started. A hot start does not affect the BDT work queue.

On a warm start, BDT reads the initialization stream and uses the information to initialize the BDT address space. A warm start does not affect the BDT work queue.

Starting Sessions with Other Nodes

After starting BDT and VTAM, the operator may start any SNA sessions that were not started automatically. To start a session with another node, the operator must use BDT commands.

If a BDT address space will take part in both file-to-file and JES3 SNA NJE transfers, the operator must start two sessions.

Monitoring BDT

When BDT receives a work request it assigns a number to it and puts it on the BDT work queue to wait until it can be scheduled and processed. The work request, or transaction, can be (1) a data set copy request or (2) a JES3 SNA NJE job or output (SYSOUT) to be sent to another node. Once on the work queue the transaction is considered a job.

BDT's INQUIRY command enables an operator to collect status information about BDT jobs. Some of the job-related information that an operator can collect includes:

- How many BDT jobs are in the system
- The scheduling priority of each job

- The elapsed time since a job began execution
- The amount of processor time a job has used
- The amount of data BDT has transferred for a job
- Which jobs are executing at a specific node.

The INQUIRY command also allows an operator to collect information about BDT resources, including:

- The status of the virtual logical units (VLUs) for SNA sessions (for example, online, offline, or active)
- Statistics concerning BDT's use of virtual storage
- The status of the BDT system log
- The number of file-to-file VLUs that are fenced (dedicated to either sending or receiving data).

After examining the information returned by the INQUIRY command, the operator may decide to change the status of a job or the status of a BDT resource.

Changing the Status of a Job or BDT Resource

To change the status of a job, the operator must use BDT's MODIFY command. The MODIFY command allows the operator to:

- Cancel a job
- Change a job's priority
- Put a job into hold status
- Release a job from hold status.

To change the status of a BDT resource, the operator must use BDT's MODIFY command or VARY command. The MODIFY command enables the operator to:

- Change installation-defined dump options
- Change the primary destination for the BDT system log
- Suspend or resume communication between BDT and JES3 (SNA NJE feature only)
- Change the number of VLUs that are fenced (File-to-File feature only).

Using the VARY command, the operator can change the number of virtual logical units (VLUs) that are available per session. The number of VLUs that are available determines how many data streams may be transmitted concurrently by BDT.

The operator can also change the time interval at which BDT reads the TQI checkpoint data set looking for work.

Chapter 4. The User's View of BDT with the File-to-File Feature

This chapter discusses how users (application programmers, engineers, and others) use the File-to-File feature of BDT. This chapter discusses:

- Submitting a transaction to copy a data set
- The types of data sets that can be copied
- Determining and changing a transaction's status
- Using a generic master job definition (GMJD) library to submit frequently used transactions
- Controlling the sequence of transaction execution
- Protecting data sets through RACF.

Submitting a Transaction

When a user wants to copy a data set, the user submits a transaction. The transaction identifies the nodes and data sets that will take part in the copy operation.

A transaction may be submitted from either node that will take part in the copy operation. For example, a data set can be copied from node A to node B by submitting a transaction at either node A or node B.

A user may submit a transaction from a TSO terminal, an MCS console, or a JES3 console. The transaction may be submitted interactively (like a command), in a batch job, or via an ISPF panel (from a TSO terminal only).

When a transaction is placed on the BDT work queue and is assigned a BDT job number, it is a BDT job.

Example of a Transaction Submitted Interactively

Here is an example of a transaction that could be submitted interactively to copy a data set from one node to another:

```
BDT Q PRIORITY(11) ACCT(DEPT60E)
FROM DATASET(IN.GEO) SHR PASSWORD(Z3F7G) LOC(NODEA) DAP(PDS)
TO DATASET(IN.GEO2) OLD LOC(NODEB)
```

In the first line of the example, the keyword "BDT" specifies that the transaction is being submitted from a TSO terminal. Q specifies that the entire *transaction definition* follows. (When Q is omitted it means that the transaction definition is precoded and stored in a generic master job definition (GMJD) library, which is discussed on page 24.) The job will have a priority of 11 and will be charged to account DEPT60E.

The second line shows the "from" section, which contains information about the data set being copied. In this case we are copying partitioned data set IN.GEO, with a disposition of SHR and a password of Z3F7GG, from node NODEA using the PDS dynamic application program (DAP) of BDT.

The third line shows the "to" section, which contains information about the receiving data set. It is a partitioned data set named IN.GEO2 and it already exists (OLD) at node NODEB.

Example of a Transaction Submitted in a Batch Job

Here is an example of the same transaction submitted in a batch job:

```
//BATCHJOB JOB CLASS=A
//          EXEC PGM=BDTBATCH
//SYSPRINT DD DUMMY
//SYSIN    DD *
```

```

Q PRIORITY(11) ACCT(DEPT60E)
FROM DATASET(IN.GEO) SHR PASSWORD(Z3F7G) LOC(NODEA) DAP(PDS)
TO DATASET(IN.GEO2) OLD LOC(NODEB)
/EOT

```

Example of a Transaction Submitted via ISPF

Here is an example of the first of two panels used to submit the same transaction via ISPF:

```

BDTE04 BDT File to File Transaction - Panel 1 of 2 - - - - -
Enter Command ==>
User ID K8076
Date 85/11/11
Time 10:27

Transaction Name ==> SAMPLE
GMJD(default library) ==>
Comment ==> COPY TO NODE B
FROM Data Set Information
Location ==> NODEA (BDT NODE NAME)
Project ==>
Library ==>
Type ==>
Member ==>

Other Partitioned or Sequential Data Set:
FROM DS Name ==> IN.GEO

FROM DS Organization(SEQ,PDS) ==> PDS
FROM Location SECPWD Keyword ==>
Additional FROM Parameters:
==> SHR PASSWORD(Z3F7G)
==> PRIORITY(11) ACCT(DEPT60E)
==>
Press ENTER for TO data set information.
Enter End to Terminate without Save

```

Types of Data Sets That Can Be Transferred

BDT can copy sequential data sets, partitioned data sets, or members of a partitioned data set.

If the data set is sequential, BDT will copy the entire data set. If the data set is partitioned, the user may request that BDT:

- Copy the complete data set.
- Copy only selected members of the data set.
- Replace a member in the “to” data set with an identically named member from the “from” data set. (The “from” data set is the data set to be copied. The “to” data set is the data set where BDT will write the copy.)

The “from” and “to” data sets may be attached to the same processor or to different processors. When they are attached to different processors the user may submit the transaction from either processor. For example, if the “from” data set is attached to processor A and the “to” data set is attached to processor B, the user may submit the transaction from either processor A or processor B.

The “from” data set and the “to” data set must be allocated on either a direct access volume or a magnetic tape volume. Both data sets may be on the same volume type or they may be on different volume types. In other words, both data sets could be on tape volumes, both could be on direct access volumes, or one could be on a tape volume and the other on a direct access volume.

When a user submits a transaction, the “from” data set must already exist; the “to” data set may or may not exist. If the “to” data set does not exist, the user’s transaction must provide the information that BDT needs to allocate that data set. Either or both data sets may or may not already be cataloged.

The data set to be copied may contain JCL statements, which may be processed by the internal reader at the receiving complex.

BDT cannot copy partitioned data set members whose directory entries contain user TTRs (relative track addresses) or note lists (pointers to blocks within a member). As a result, users cannot copy programs they have link-edited into an overlay structure.

Record Formats

The “from” and “to” data sets may have any one of the following record formats:

- Fixed
- Fixed-blocked
- Variable
- Variable-blocked
- Undefined.

The record formats of the two data sets need not be the same, however. For example, the “from” data set could have a record format of fixed while the “to” data set has a record format of variable. When the data sets have different record formats, BDT automatically reformats the data before writing it into the “to” data set. The “from” data set remains unchanged.

Data Set Organization

The “from” and “to” data sets may have identical organizations. However, they are not required to. For example, both data sets may have a sequential organization or a partitioned organization. It is also valid for one data set to have a partitioned organization and the other data set a sequential organization.

Logical Record Lengths

Sequential “from” and “to” data sets may have different logical record lengths or block sizes as well as different record formats. When the logical record lengths or block sizes are different, BDT pads or truncates and reformats the data before writing it into the “to” data set.

For partitioned data sets, unlike sequential data sets, certain size relationships must exist between the logical record lengths and block sizes of the “from” and “to” data sets. These relationships depend upon the record format of the data sets as follows:

- Fixed or fixed-blocked — the logical record lengths of the “from” and “to” data sets must be equal
- Variable or variable-blocked — the logical record length of the “from” data set must be equal to or smaller than the logical record length of the “to” data set
- Undefined — the block size of the “from” data set must be less than or equal to the block size of the “to” data set.

Determining and Changing a Transaction’s Status

To determine the status of a transaction, the user may use BDT’s INQUIRY command. Information that this command provides can tell the user:

- The job number BDT assigned to the transaction
- The scheduling priority of the transaction
- The enqueue status of the “from” and “to” data sets
- The elapsed time since the transaction began execution
- The amount of processor time used to process the transaction
- The number of bytes of data copied to the “to” data set.

Depending on the progress of a transaction, the user may wish to change the processing status of the transaction. For example, if a transaction has waited for some time to be scheduled, the user may wish to increase that transaction’s scheduling priority. BDT’s MODIFY command enables a user to:

- Change a transaction’s scheduling priority

- Hold a transaction, that is, prevent BDT from scheduling it
- Release a transaction, that is, allow BDT to schedule a transaction that was being held
- Cancel a transaction.

A Simplified Way to Submit Frequently Used Transactions

Most users will find there are certain transactions that they submit over and over. To avoid coding the full transaction definition each time, the user can code the transaction definition one time, name it, and store it as a member of a generic master job definition (GMJD) library. Each user can have a private GMJD library and there can be a system GMJD library that is available to all users.

To use a transaction definition stored in a system GMJD library, a user would submit a transaction that specifies the name of the GMJD library member. For example, to submit the transaction stored in MYJOB, a user would enter:

```
BDT MYJOB
```

It is not necessary for the entire transaction definition to be stored in a GMJD library. BDT, at the node where the transaction is submitted, will combine parameters from the submitted transaction definition with parameters from the stored definition to create a resulting transaction definition. This makes it possible for a user to add parameters to the stored definition or use a value that is different from what is in the stored definition. For example, if a user would want the transaction stored as MYJOB to be directed to the data set named NEWDATA instead of the data set named in the stored definition, the user would enter:

```
BDT MYJOB TO DATASET(NEWDATA)
```

Controlling the Sequence of Transaction Execution

There may be times when a user wants to ensure that one transaction executes before another. To specify the execution sequence of two or more transactions, the user must define the transactions as part of a *dependent transaction control (DTC) network*. This is done by specifying the appropriate parameters at the time the transactions are submitted. Then, regardless of the order in which the transactions are submitted, they will execute in the specified order.

Figure 12 shows a simple DTC network of three transactions. In this network, transaction 1 will execute first, followed by transaction 2 and then transaction 3, in that order.

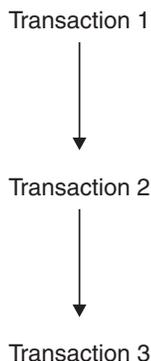


Figure 12. A Simple Dependent Transaction Control (DTC) Network

Figure 13 on page 25 shows a more complex DTC network. In this network, transaction 1 will execute first, followed by transaction 2 or transaction 3. Because transaction 2 and transaction 3 have no dependencies on each other, the order in which they execute, relative to each other, makes no difference. However, transaction 1 must complete execution before transaction 2 or transaction 3 can execute. Transaction 2 and transaction 3 must both complete execution before transaction 4 can execute.

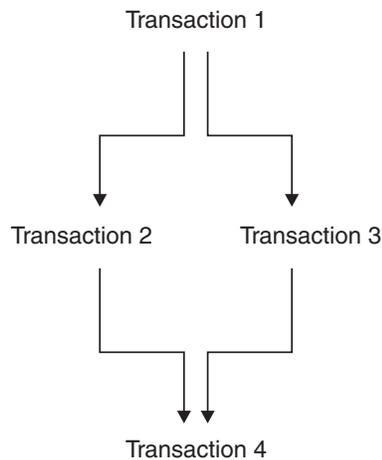


Figure 13. A More Complex Dependent Transaction Control (DTC) Network

Protecting Data Sets through RACF

If both nodes that take part in a transaction include the Resource Access Control Facility (RACF) program product, users can RACF-protect newly created data sets. To request RACF protection or to access a RACF-protected data set, users must submit their transactions through a TSO terminal either interactively or in a batch job. At the time they submit a transaction they may request RACF protection for the “to” data set. The transaction must provide, either explicitly or by default, a user ID and a password. RACF will use this user ID and password to protect the “to” data set.

The “from” data set may also be RACF-protected. If it is, a transaction will have to specify an authorized user or group ID and password (either explicitly or by default) before RACF will allow a user to copy the data set.

Chapter 5. The User's View of BDT with the SNA NJE Feature

This chapter discusses how users (application programmers, engineers, and others) use the SNA NJE feature of BDT.

The SNA NJE feature of BDT allows JES3 to transmit jobs, output (SYSOUT), commands, and messages from one computer system to another within a SNA network. This is called job networking.

The user interface to JES3 job networking in a SNA network is through JES3, not BDT. This user interface is the same as in a BSC network. Users submit job control language (JCL) statements and JES3 control statements to JES3. Users then receive status messages from JES3.

Users who currently submit jobs to a JES3 BSC job network will not need to make any changes to submit jobs to a JES3 SNA job network with BDT, as long as the system programmer has assigned the same names to the SNA nodes as to the BSC nodes. The fact that the network is using a different type of communication link, and the fact that BDT enters jobs into the network for JES3 (rather than JES3 entering them directly) will not be apparent.

The OS/390 JCL manuals describe how to submit jobs to a JES3 job network.

Chapter 6. Hardware and Software Requirements

This chapter describes the hardware and software required to use BDT.

Bulk Data Transfer consists of three features:

- The Base feature which is a prerequisite to the other two features and which provides the transaction queuing integrity (TQI) facility
- The File-to-File feature which lets OS/390 JES2 and OS/390 JES3 users copy data sets from one computer system to another within a SNA network.
- The SNA NJE feature which allows OS/390 JES3 users to transmit jobs, output (SYSOUT), commands, and messages from one computer system to another within a SNA network.

Hardware Requirements

BDT has no special hardware needs. It can use any hardware configuration supported by the OS/390.

Software Requirements

In order for users of a JES2 or JES3 complex to send or receive data sets, at least one processor in the complex must have the Base and File-to-File features installed. Other processors in the complex may have both features or just the Base feature. Processors that have both features allow users to submit commands and file-to-file transactions, checkpoint the commands and transactions, process them, and send BDT messages to users. Processors that have only the Base feature installed do not process commands and transactions but do perform the other three activities; another processor in the complex, which has both the Base and File-to-File features installed, processes the commands and transactions.

In order for users of a JES3 complex to send or receive SNA NJE work, one (and only one) processor in the complex must have the Base and SNA NJE features installed. Other processors in the complex may have just the Base feature installed. The processor that has both features allows users to submit jobs and commands, checkpoints the commands, processes the jobs and commands, and sends output and messages to users. Processors that have only the Base feature installed allow users to submit commands, checkpoint the commands, and send output and messages to users, but do not allow users to submit jobs, nor do they process jobs or commands.

Appendix A. Differences between File-to-File and SNA NJE Support

This appendix describes detailed differences between BDT's file-to-file support, as provided in the File-to-File feature and BDT's JES3 SNA NJE support.

Poly-BDT Complexes

Any file-to-file node in a poly-BDT complex, as designated in the BDT initialization stream, can process file-to-file transactions. Only one SNA NJE node in a poly-BDT complex, as designated in the JES3 initialization stream, can process SNA NJE work.

No Global-Local Relationships with SNA NJE

File-to-file nodes have global and local relationships with each other. Work that takes place between two nodes is scheduled at the global node of the pair. SNA NJE nodes, however, do not have global-local relationships, but rather a peer relationship. Work requests are scheduled at the node where they are submitted.

Virtual Logical Units (VLUs)

Up to 254 VLUs can be specified for transferring file-to-file data. Any number of the VLUs can be fenced in either direction (sending or receiving). Up to 28 VLUs can be specified for transferring SNA NJE jobs, and they can only be specified in groups of four. Each group contains one VLU for handling jobs received, one for jobs sent, one for output received, and one for output sent. Fencing of SNA NJE VLUs is not allowed.

Use of the JES3 Spool

File-to-file transfers go directly from one user data set to another. SNA NJE transfers go from JES3 spool to JES3 spool.

Data Compression

BDT optionally compresses duplicate data strings when copying data sets. BDT always compresses duplicate data strings when transferring SNA NJE jobs.

Data Checkpointing

File-to-file transfers that fail can be retried. BDT uses a checkpoint file that can automatically resume incomplete transfers from the last checkpoint. SNA NJE transfers that fail are retransmitted in their entirety, not from a checkpoint.

Transaction Queuing Integrity (TQI) Facility

For file-to-file users, TQI prevents the loss of transactions and commands, and routes BDT messages to users. For SNA NJE users, TQI prevents the loss of commands and routes BDT messages to users.

Initialization

For SNA NJE nodes, only the BDTNODE statement is used to specify the characteristics of a remote node and control access to the node.

The BDTNODE statement for defining a remote file-to-file node has several parameters that do not apply to SNA NJE definitions.

The SYSID statement has several new parameters for SNA NJE.

SNA NJE definitions require one more CELLPOOL statement than file-to-file definitions.

The BDTNODE statement for defining a home node is not used when defining SNA NJE nodes.

User Exit Routines

Some user exit routines are invoked for file-to-file transfers but not for SNA NJE transfers.

Commands

The MODIFY,JES3 command applies to SNA NJE only. This command lets an operator suspend or resume the transfer of SNA NJE jobs from JES3 to BDT.

Three commands apply to file-to-file systems only:

- MESSAGE
- SEND
- MODIFY,NODE,FENCE

With the SNA NJE feature, an operator who wants to change a job's status sometimes has to deal with both JES3's queue and BDT's queue. To display status, the operator can issue the INQUIRY command to JES3. If the job is on JES3's job queue, the operator uses JES3 commands to hold the job, change its priority, let it run, or cancel it. If JES3 no longer has the job but has sent the work to BDT, a message will say so. The operator can then issue a BDT command to hold the job, change its priority, or let it run; but to cancel it, the operator can use either a JES3 command or a BDT command.

No Dependent Transaction Control (DTC) Networks for SNA NJE

Dependent transaction control (DTC) networks provide a way to control the order in which file-to-file transactions are executed. There is no such facility for SNA NJE jobs.

System Management Facilities (SMF) Type 59 Record

For file-to-file transfers, BDT creates a type 59 SMF record after copying a data set. For SNA NJE transfers, BDT creates a type 59 SMF record after sending a JES3 SNA NJE job or output (SYSOUT).

Appendix B. SNA NJE Flows with BDT

This appendix describes the flow of work in a network that has BDT with the SNA NJE feature. It is provided for those who would like detailed information about the way that JES3 and BDT work together to manage SNA network transfers.

SNA NJE Job and Output Flows

Figure 14 illustrates the flow of a JES3 SNA NJE job or output (SYSOUT) from the time it is submitted until the time it is received at the receiving node. Although the figure shows JES3 and BDT at both nodes of the transfer, one of the nodes could be JES2, RSCS Networking, or VSE/POWER.

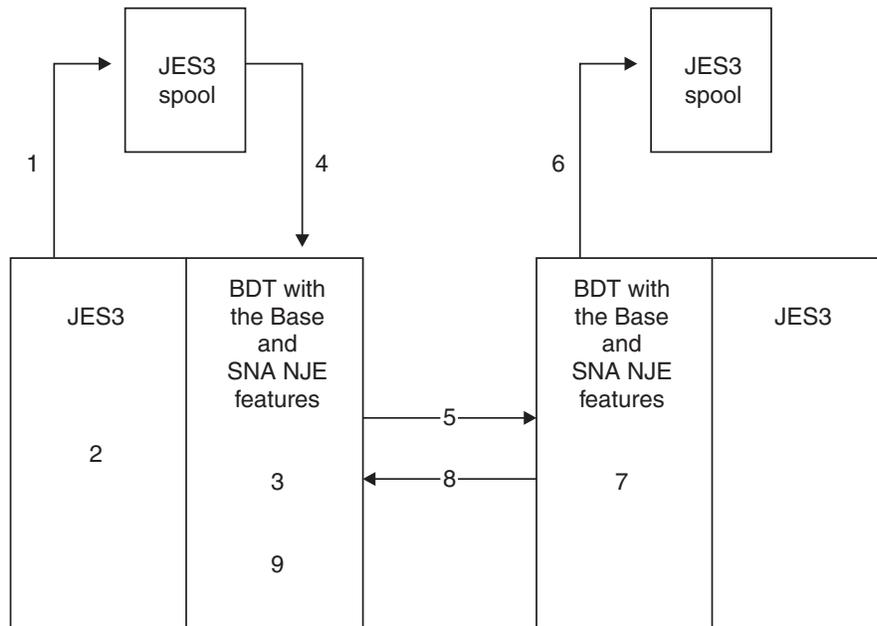


Figure 14. SNA NJE Job and SYSOUT Flows with BDT

The following explanations apply to Figure 14:

1. JES3 creates a job or SYSOUT data set on spool for a request that a user submits to JES3. The request is to send a job or output (SYSOUT) to another node.
2. JES3 notifies BDT that there is work to be sent to another node.
3. BDT puts the job on BDT's work queue, assigns a BDT job number to it, and notifies JES3 of the job number. (A JES3 command can display the JES3 and corresponding BDT job number.)
4. BDT retrieves the NJE data set from spool.
5. BDT compresses and transmits the NJE data.
6. At the other node, BDT decompresses the data and stores it on spool. This continues until all the data is retrieved.
7. BDT notifies JES3 that the data has been stored.
8. BDT notifies the sender that the data has been stored.
9. BDT notifies JES3 that the NJE work is complete. JES3 deletes the data from the spool.

SNA NJE Message and Command Flows

Figure 15 illustrates the flow of a JES3 SNA NJE message or command from the time it is created until the time it reaches its destination. Although the figure shows JES3 and BDT at both nodes of the transfer, one of the nodes could be JES2, RSCS Networking, or VSE/POWER.

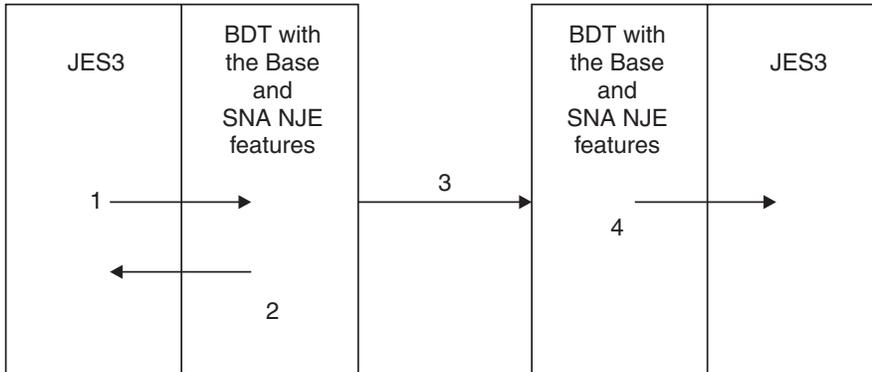


Figure 15. SNA NJE Message and Command Flows with BDT

The following explanations apply to Figure 15:

1. JES3 creates and passes the message or JES3 command to BDT.
2. If the session is not active, BDT returns the message or JES3 command to JES3.
3. If the session is active, BDT sends the message or JES3 command.
4. BDT passes the message or JES3 command to JES3.

Appendix C. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- ACF/VTAM
- BookManager
- CICS
- DFSMSdfp
- DFSMSdss
- DFSMSHsm
- DFSMSrmm
- DFSMS/MVS
- DFSORT
- ESCON
- GDDM
- Hardware Configuration Definition
- IBMLink
- IBM
- IMS
- MVS
- MVS/ESA
- OS/390
- RACF
- Resource Link
- RMF
- VisualLift
- VTAM
- z/OS

—

This glossary defines important terms and abbreviations used in this book. If you do not find the term you are looking for, refer to the index or to the *IBM Dictionary of Computing* New York: McGraw-Hill, 1994.

ACF/VTAM. Advanced Communication Function for the Virtual Telecommunications Access Method.

Advanced Communication Function for the Virtual Telecommunications Access Method (ACF/VTAM). A program product that provides single-domain network capability, and optionally, multiple-domain capability.

Base feature. A feature of BDT that is a prerequisite for the File-to-File feature and the SNA NJE feature, and that contains the transaction queuing integrity (TQI) facility.

BDT. See *Bulk Data Transfer*.

Binary synchronous communication (BSC). A uniform procedure, using a standardized set of control characters and control character sequences, for synchronous transmission of binary-coded data between stations.

BSC. Binary synchronous communication.

Bulk Data Transfer. An IBM program product that (1) copies sequential or partitioned data sets from a JES2 or JES3 computer complex to another JES2 or JES3 computer complex within a SNA network, and (2) allows JES3 computer complexes to participate in SNA NJE networks.

fencing. In BDT, a method by which an installation can restrict the direction in which a set of VLU's can transmit file-to-file data.

File-to-File feature. A feature of BDT that lets JES2 and JES3 users copy data sets from one computer system to another within a SNA network.

full duplex. Pertaining to a simultaneous two-way independent transmission in both directions.

generic master job definition (GMJD) library. In BDT, a data set that contains predefined transaction definitions.

global node. In BDT, the node that schedules and manages all file-to-file transactions involving itself and a local node and responds to commands issued against those transactions. The global node has a BDT address space, which contains the BDT work queue.

GMJD library. See *generic master job definition library*.

initialization stream. In BDT, a set of installation-written statements that define a node and its operating environment.

Interactive System Productivity Facility. A program product that provides menus and data entry panels for using system functions.

ISPF. Interactive System Productivity Facility.

JES2. Job entry subsystem 2.

JES3. Job entry subsystem 3.

job entry subsystem 2 (JES2). A component of MVS that receives jobs into the system and processes all output data produced by the jobs. JES2 exerts decentralized control over multiple processor complexes.

job entry subsystem 3 (JES3). A component of MVS that receives jobs into the system and processes all output data produced by the jobs. JES3 exerts centralized control over multiple processor complexes.

job networking. The transmission of jobs, in-stream data sets, operator commands and messages, system output data sets, and job accounting information from one computer complex to another across a telecommunication link. Synonym for *network job entry (NJE)*.

local node. In BDT, the node that receives file-to-file transactions and commands submitted by users and sends them to the global node for processing.

multileaving. The synchronized bidirectional transmission of a variable number of data streams between two nodes.

network. In BDT, two or more BDT nodes that are joined by SNA sessions.

network job entry (NJE). The transmission of jobs, in-stream data sets, operator commands and messages, system output data sets, and job accounting information from one computer complex to another across a telecommunication link. Synonymous with *job networking*.

NJE. Network job entry.

node. In BDT, the point in a BDT address space that is linked to another BDT address space for either file-to-file communication or SNA NJE communication.

padding. In SNA, a technique by which a receiving component controls the rate of transmission of a sending component to prevent overrun or congestion.

poly-BDT complex. A JES complex that has more than one BDT address space.

POWER. Priority Output Writers, Execution Processors, and Input Readers.

RACF. Resource Access Control Facility.

Remote Spooling Communications Subsystem (RSCS). A program product that transfers spool files between VM/SP users, remote stations, and remote and local batch systems via telecommunication facilities.

Resource Access Control Facility (RACF). A program product that provides for access control by identifying and verifying users to the system, authorizing access to DASD data sets, logging detected unauthorized attempts to enter the system, and logging detected accesses to protected data sets.

RSCS. Remote Spooling Communications Subsystem.

session. In SNA, a logical connection between two network addressable units. The connection can be activated, deactivated, or tailored to provide different protocols.

session password. A character string that a node must provide when it attempts to establish a session with another node. Both nodes must provide the same password in order to establish the session.

SNA. Systems Network Architecture.

SNA NJE feature. A feature of BDT that allows JES3 to transmit jobs, output (SYSOUT), commands, and messages from one computer system to another within a SNA network.

spooling. (Simultaneous peripheral operation online.) The reading of input data streams and the writing of output data streams on auxiliary storage devices, concurrently with job execution, in a format convenient for later processing or output operations.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

TQI. Transaction queuing integrity.

transaction. In BDT, (1) a request to copy a data set, transmit a SNA NJE job, or transmit SNA NJE output (SYSOUT), and (2) the work that BDT does to process the request. Requests to copy data sets are submitted to BDT by users. Requests to transmit SNA NJE jobs and output are submitted to BDT by JES3.

transaction definition. In BDT, a character string that identifies the data set BDT is to copy, the data set into

which BDT is to write the copy, and parameter values that BDT is to use while processing the transaction definition.

transaction queuing integrity (TQI) facility. In BDT, a program that records commands and file-to-file transactions on a data set at the submitting node, thus allowing the transfers to be resubmitted automatically should they not reach the BDT work queue. TQI also allows users to receive messages.

user exit. A point in an IBM-supplied program at which a user exit routine may be given control.

user exit routine. A routine written by a user to take control at a user exit of a program supplied by IBM.

virtual logical unit (VLU). In BDT, data and program logic that represent one user of a SNA session. The virtual logical unit enables more than one user to concurrently use a session.

Virtual Memory/System Product. A program product that (1) manages the resources of a computing system in such a way that multiple remote terminal users have a functional simulation of a computing system (a virtual machine) at their disposal, and that (2) provides general time sharing, program development, and problem solving facilities.

Virtual Storage Extended/Priority Output Writers, Execution Processors, and Input Readers (VSE/POWER). A program product that serves as the spooling system for the VSE operating system.

VLU. Virtual logical unit.

VM/SP. Virtual Memory/System Product.

VSE/POWER. Virtual Storage Extended/Priority Output Writers, Execution Processors, and Input Readers.

work queue. In BDT, a queue whose elements represent work that BDT must do on behalf of transactions.

Readers' Comments — We'd Like to Hear from You

z/OS
Bulk Data Transfer
Overview

Publication No. SA22-7510-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>				

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>				
Complete	<input type="checkbox"/>				
Easy to find	<input type="checkbox"/>				
Easy to understand	<input type="checkbox"/>				
Well organized	<input type="checkbox"/>				
Applicable to your tasks	<input type="checkbox"/>				

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? Yes No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.



Fold and Tape

Please do not staple

Fold and Tape



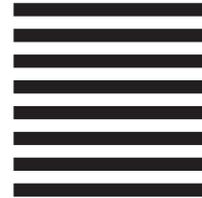
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY
12601-5400



Fold and Tape

Please do not staple

Fold and Tape

Index

Numerics

59, SMF record type 17

A

address space i
BDT runs in own 3, 14
diagnosing failures in BDT 16
dump of BDT 16
how start type affects initialization of 19
more than one BDT 14
relationship to BDT node 7
starting BDT 19
starting TQI 19
TQI runs in own 12
writing start procedure for BDT 11
writing start procedure for TQI 12
allocation of data sets for copying 22
AMDPRDMP service aid to format BDT dump 16
APF (authorized program facility) library 12
authorized program facility (APF) library 12

B

Base feature of BDT i
hardware required to use 29
introduction to 1
relationship to other BDT features 1
batch submission of transactions 2
example 21
BDTNODE statement 31
block sizes for data set copy 23
books, BDT ix
buffers, data 9

C

catalog requirement for copying data sets 22
checkpointing i
difference between file-to-file and SNA NJE 31
of data being copied 2
of transactions and commands (by TQI) 1, 12
coexistence i
of JES3 BSC NJE and BDT SNA NJE 5
cold start of BDT 19
commands i
can be sent through job network 3
differences between file-to-file and SNA NJE 32
flow through SNA NJE system 34
preventing loss of 1
recorded in log data set 17
SMF record not written for 17
use of INQUIRY by operator 19
use of INQUIRY by user 23
use of MODIFY by operator 20
use of MODIFY by user 23
use of VARY by operator 20
communication parameters i

communication parameters i (*continued*)
agreement between nodes 9
how they are negotiated 9
role in establishing a session 9
communication VLU 10
compression i
of file-to-file data 3
of SNA NJE data 31
concurrent transmission i
of file-to-file data 10
of SNA NJE data 4, 10
CONSOLE statement of JES3 12
consoles for submitting transactions 2

D

data buffers 9
data compression i
of file-to-file data 3
of SNA NJE data 31
data overrun, preventing 9
data security i
password-protected nodes 3, 9
RACF-protected data sets 3, 25
user exit routines 3, 11
data set i
checkpoint 2
dump 16
initialization stream 16
log 17
system GMJD library 14
system management facilities (SMF) 17
TQI checkpoint 12
transaction queuing integrity (TQI) checkpoint 1
work queue 19
data sets that can be copied, types of 22
data transfer VLUs 10
debugging BDT 16
dependent job control (DJC) network 24
dependent transaction control (DTC) network 24
diagnosing BDT failures 16
differences i
between file-to-file and SNA NJE support 31
DJC (dependent job control) network 24
DTC (dependent transaction control) network 24
dump, BDT 16
duplex operation of BDT, full 2

E

errors in BDT, diagnosing 16
exit routines, user i
differences between Versions 1 and 2 32
introduction to 11
required and optional 11

F

fencing i
introduction to 11

fencing *i (continued)*
not possible with SNA NJE 31
file *i*
File-to-File feature of BDT *i*
hardware required to use 29
how different from SNA NJE support 31
introduction to 1
relationship to the Base feature 1
user's view of 21
flows, SNA NJE 33
FMIDs 29
format of records for data set copy 23
formatted dump of BDT 16
full duplex operation of BDT 2

G

generic master job definition (GMJD) library *i*
creation of 14
use of 24
global-local node relationship *i*
introduction to 7
not applicable to SNA NJE 31
global resource serialization (GRS) complex 12
GMJD library *i*
GRS (global resource serialization) complex 12

H

hardware requirements, BDT 29
hot start of BDT 19

I

IEASYSxx member of SYS1.PARMLIB 12
IEFSSNxx member of SYS1.PARMLIB 12
initialization statements *i*
BDT 7
JES3 12
initialization stream data set 16
INQUIRY command *i*
used by operator 19
used by user 23
ISPF panels for copying data sets *i*
example of use 22
introduction to 21

J

JCL (job control language) *i*
can be copied by BDT 22
for starting BDT address space 11
for submitting SNA NJE jobs 27
JES3 control statements for submitting SNA NJE jobs 27
job control language *i*
job flow through SNA NJE system 33
job networking 3
job number *i*
assigned to file-to-file transaction 2, 21
determining with INQUIRY command 23

job number *i (continued)*
when assigned to SNA NJE job 33
written to SMF data set 17
job submission to JES3 27
jobs, monitoring 19

L

lengths of records for data set copy 23
link-edit restriction when copying data sets 23
local-global node relationship *i*
introduction to 7
not applicable to SNA NJE 31
log data set, BDT 17
logical record lengths for data set copy 23
logon mode table 12
lost transactions, recovery of 1

M

manuals, BDT *ix*
MCS console for submitting transactions 2
messages *i*
can be sent through job network 3
ensuring delivery to users 1
flow through SNA NJE system 34
recorded in log data set 17
rerouting via a user exit routine 11
SMF record not written for 17
migration *i*
from BSC to SNA NJE can be gradual 5
MODIFY command *i*
used by operator 20
used by user 23
monitoring BDT 19
multileaving 10
multiple BDT address spaces 14
MVS/ESA *i*
MVS secondary subsystem, BDT as 12

N

network, dependent transaction control (DTC) 24
networking, job 3
NJERMT statement of JES3 12
nodes, BDT *i*
defining 7
global-local relationship between 7
password-protected 3
separate for file-to-file and SNA NJE 7
starting sessions with 19
note list restriction when copying data sets 23

O

operator's view of BDT 19
organization, data set 23
output (SYSOUT) *i*
assigning priority to 4
can be sent through job network 3
flow through SNA NJE system 33

output (SYSOUT) i *(continued)*
SMF record for 17
virtual logical units (VLUs) for sending 10
overlay restriction when copying data sets 23
overrun of data, preventing 9

P

pacing i
glossary definition 37
use by BDT 9
panels for copying data sets, ISPF i
example of use 22
introduction to 21
partitioned data sets i
allowable copy operations 22
required size relationships 23
what BDT copies 22
passwords for nodes, session 3, 9
poly-BDT complex i
examples 14
introduction to 14
prerequisite programs 29
priority, scheduling i
of file-to-file transactions 3
of SNA NJE jobs 4
production and test system separation 15
PRTY parameter of OS/390 JOB statement 4
publications, BDT ix

Q

queue, BDT work 1, 19

R

RACF protection for data sets 25
record formats for data set copy 23
record lengths for data set copy 23
recovery i
of data being copied 2
of transactions and commands (by TQI) 1
requirements, hardware and software 29
restart, automatic i
of data being copied 2
of sessions 9
of transactions and commands (by TQI) 1, 12
RSCS Networking program product i
participation in a job network 4

S

SAF (system authorization facility) interface 3
SAMPLIB i
scheduling priority. i
of file-to-file transactions 3
of SNA NJE jobs 4
security, data i
password-protected nodes 3, 9
RACF-protected data sets 3, 25
user exit routines 3, 11

sequence of transaction execution, controlling the 24
sequential data sets i
logical record length 23
record format 23
what BDT copies 22
sessions i
automatic restart of 9
automatic startup of 8
concurrent 8
passwords for 9
requirements to establish 8
separate for file-to-file and SNA NJE 19
starting 19
SMF (system management facilities) 17
SNA (Systems Network Architecture) i
advantages with File-to-File feature 2
advantages with SNA NJE feature 4
SNA NJE feature of BDT i
from the user's view 27
hardware required to use 29
how different from file-to-file support 31
introduction to 3
relationship to the Base feature 1
SNALINE statement 31
software requirements i
software requirements, BDT 29
spool data set i
diagram showing role in SNA NJE flow 33
not used with File-to-File feature 2
used with SNA NJE feature 31
start procedure, writing a i
for BDT 11
for TQI 12
starting i
BDT 19
sessions with other nodes 19
transaction queuing integrity (TQI) facility 19
VTAM 19
store and forward 4
subsessions 10
supervisor state 12
SYS1.DUMP data set 16
SYS1.PARMLIB 12
SYS1.SAMPLIB i
BDT start procedure in 11
initialization statements in 7
user exit routines in 11
SYSID statement of JES3 12
SYSOUT i
assigning priority to 4
can be sent through job network 3
flow through SNA NJE system 33
SMF record for 17
virtual logical units (VLUs) for sending 10
system authorization facility (SAF) interface 3
system management facilities (SMF) 17
system requirements 29
Systems Network Architecture (SNA) i
advantages with File-to-File feature 2
advantages with SNA NJE feature 4

T

- table, BDT trace 17
- terminals for submitting transactions 2
- test and production system separation 15
- TQI (transaction queuing integrity) facility i
 - defining 12
 - introduction to 1
 - sample configurations 12
 - starting 19
- trace table, BDT 17
- transaction definition 21
- transaction queuing integrity (TQI) facility i
 - defining 12
 - introduction to 1
 - sample configurations 12
 - starting 19
- transactions, file-to-file i
 - aid to frequent submission (GMJD library) 24
 - batch example 21
 - controlling execution sequence 24
 - interactive example 21
 - introduction to 1
 - ISPF example 22
 - monitoring the status of 23
 - preventing loss of 1
 - recorded in log data set 17
 - submitting 21
 - use of RACF with 25
 - user-assigned priorities for 3
 - when considered jobs 19
- TSO terminal for submitting transactions 2
- TTR restriction when copying data sets 23
- type 59 SMF record 17

U

- unformatted dump of BDT 16
- user's view of BDT 21
- user exit routines. i
 - differences between Versions 1 and 2 32
 - introduction to 11
 - required and optional 11

V

- VARY command 20
- virtual logical units (VLUs) i
 - displaying information about 20
 - fencing of 11
 - introduction to 10
 - number per session 10
- VLUs i
- VM/SP program product i
 - participation in a job network 4
- volume requirements for data sets 22
- VSE/Advanced Functions program product i
 - participation in a job network 4
- VSE/POWER program product i
 - participation in a job network 4
- VTAM i
 - benefit to file-to-file users 2

VTAM i (continued)

- benefit to SNA NJE users 4
- defining BDT to 12
- pacing used by BDT 10
- role in session establishment 9
- starting 19

W

- warm start of BDT 19
- work queue, BDT 1, 19



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SA22-7510-00

