z/OS
Version 2 Release 3

Bulk Data Transfer File-to-File Transaction Guide

SC14-7583-30

Note

Before using this information and the product it supports, read the information in "Notices" on page 89.

This edition applies to Version 2 Release 3 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2019-02-15

Contents

List of Figures

About This Book

This book describes Bulk Data Transfer (BDT) file-to-file transactions. It explains how to write and submit file-to-file transactions to BDT for execution, and describes the file-to-file transaction parameters.

This book does not explain how to submit SNA network job entry (NJE) jobs to JES3 for execution. For information on submitting SNA NJE jobs, see the JCL manual that is appropriate for your installation.

Who Should Read This Book

This book is for anyone who wishes to submit a BDT file-to-file transaction.

How to Use This Book

If you are unfamiliar with BDT file-to-file transactions, begin with Chapter 1, "Writing and Submitting File-to-File Transactions," on page 1.

This book contains five chapters, an appendix, and a glossary:

- Chapter 1, "Writing and Submitting File-to-File Transactions," on page 1 explains what BDT file-to-file transactions can do as well as how to write and submit transactions.
- Chapter 2, "Storing and Reusing Transaction Definitions — GMJD Libraries," on page 9 explains how to use generic master job definition (GMJD) libraries.
- Chapter 3, "Controlling the Order in Which Transactions Are Processed — DTC Networks," on page 13 explains how to put transactions into dependent transaction control (DTC) networks to control the order in which they are processed.
- Chapter 4, "Sample Transactions," on page 19 contains sample transactions.
- Chapter 5, "File-to-File Transaction Parameter Reference," on page 23 contains a summary of optional and required transaction parameters followed by detailed descriptions of each of the parameters.
- Appendix A, "Parameters That Require Other Parameters," on page 83 is a quick reference to parameters that require other parameters, including those parameters that can be used to allocate new data sets.

Related Reading

Where necessary, this book references information in other books, using shortened versions of the book title. For complete titles and order numbers of the books for all products that are part of z/OS, see *z/OS Information Roadmap*.

How to send your comments to IBM

We invite you to submit comments about the z/OS® product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

Important: If your comment regards a technical question or problem, see instead "If you have a technical problem" on page xv.

Submit your feedback by using the appropriate method for your type of comment or question:

Feedback on z/OS function
If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

Feedback on IBM® Knowledge Center function
If your comment or question is about the IBM Knowledge Center functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Knowledge Center Support at ibmkc@us.ibm.com.

Feedback on the z/OS product documentation and content
If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: z/OS BDT File-to-File Transaction Guide, SC14-7583-30
- The section title of the specific information to which your comment relates
- The text of your comment.

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

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Summary of changes for z/OS BDT File-to-File Transaction Guide for Version 2 Release 3 (V2R3) and its updates

This information contains no technical changes for this release.

z/OS Version 2 Release 1 summary of changes

See the Version 2 Release 1 (V2R1) versions of the following publications for all enhancements related to z/OS V2R1:

- *z/OS Migration*
- *z/OS Planning for Installation*
- *z/OS Summary of Message and Interface Changes*
- *z/OS Introduction and Release Guide*

Chapter 1. Writing and Submitting File-to-File Transactions

This chapter is a general introduction to writing and submitting BDT file-to-file transactions. It explains:

- The kinds of data sets you can copy with BDT
- How to write file-to-file transactions to copy data sets
- How to read the transaction parameter syntax diagrams in this book
- How to submit file-to-file transactions
- How BDT processes file-to-file transactions.

The Kinds of Data Sets You Can Copy with BDT File-to-File Transactions

BDT file-to-file transactions allow you to copy a data set to another data set. The data set you copy from and the data set you copy to can have a variety of locations, organizations, record formats, record lengths, and storage devices.

Locations of the Data Sets

BDT can copy data sets within a network of BDT *nodes*. A BDT node is a point in a BDT address space that is linked to another BDT address space for data transfer. BDT nodes are defined by your system programmer.

Within the network of nodes, BDT can copy:

- A data set located at your node to another data set located at your node. This is called an *internal transfer*.
- A data set located at your node to a data set located at another node.
- A data set located at another node to a data set located at your node.

Organizations of the Data Sets

BDT can copy data sets with partitioned or sequential organization. The data set being copied from and the data set being copied to may have the same organization or different organization. Table 1 on page 1 shows the combinations of sequential data sets and partitioned data sets (PDSs) that BDT can copy.

Table 1: Organizations of Data Sets BDT Can Copy

If the "from" data set is:	**The "to" data set may be:**
A sequential data set	A sequential data set or PDS member
A PDS member	A sequential data set or PDS member
Multiple PDS members	Multiple PDS members

Record Formats of the Data Sets

BDT can copy data sets with any of these record formats:

- Fixed length
- Fixed length and blocked
- Variable length

- Variable length and blocked
- Undefined length

The data set being copied from and the data set being copied to may have different record formats. If the record formats of the data sets are different, BDT reformats the data before writing it to the “to” data set.

Record Lengths and Block Sizes of the Data Sets

BDT can copy a data set with one logical record length and block size to a data set with a different logical record length and block size. However, BDT has some requirements for the logical record lengths and block sizes of data sets being copied. These requirements, which depend on the organization and record format of the data sets being copied, are subsequently shown.

If the organization of the data sets is:	And the record format is:	This must be true of the record length or block size:
Sequential for both	Fixed length, fixed length and blocked, variable length, variable length and blocked, or undefined, for either or both data sets	The logical record length of the “from” data set must be equal to or smaller than the maximum record length defined in the “to” data set's data control block (DCB). If the record length of the “from” data set is larger than the record length of the “to” data set, BDT truncates the records in the “to” data set.
Partitioned for one or both	Fixed length or fixed length and blocked for either data set	The logical record lengths of the “to” and “from” data sets must be equal.
	Variable length or variable length and blocked for either data set	The logical record length of the “to” data set must be equal to or larger than the logical record length of the “from” data set.
	Undefined for either data set	The block size of the “to” data set must be equal to or larger than the block size of the “from” data set.

Storage Devices of the Data Sets

BDT can copy data sets stored on either direct access storage devices (DASD) or on magnetic tape. The data set being copied from and the data set being copied to may be stored on the same type of device or on different types of devices.

How to write a transaction to copy a data set

To copy a data set, you write a transaction and submit it to BDT. The transaction defines the nodes and data sets that will take part in the transfer of data.

Figure 1 on page 3 shows the general format of a transaction. The pages following the figure discuss the parts of a transaction in detail.

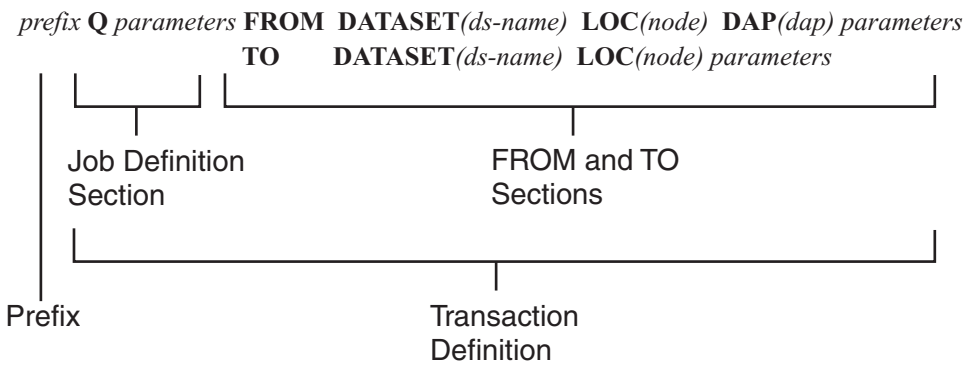


Figure 1: General format of a transaction

A transaction has two major parts: the prefix and the transaction definition. The transaction definition, in turn, is made up of the job definition section, the FROM section, and the TO section.

Prefix

The first part of a transaction is a prefix that tells the system that this is a BDT transaction. The prefix you use depends on the type of terminal or console you are using: TSO, MCS, or JES3; and whether your system programmer has created any shortcuts for you (such as a special symbol or PF keys). Your system programmer can tell you what prefix to use. Table 2 on page 3 shows the IBM-defined prefixes for transactions.

Table 2: IBM-Defined BDT Prefixes

Type of Console	Prefix in a Single-BDT Complex	Prefix in a Poly-BDT Complex
MCS	*bdt-char* or BDT or F [*bdt-proc.*]*id*	same
JES3	*S,BDT	*S,BDT,SY(*node-name*)
TSO	BDT	BDT SY(*node-name*)

In Table 2 on page 3:

- *bdt-char* is a special character (such as +) defined by your system programmer that routes the transaction directly to a specific system. Do not use a delimiter (a comma or a blank) after *bdt-char*. For information on defining *bdt-char* see *z/OS BDT Installation*.
- *bdt-proc* is the name of the cataloged procedure in SYS1.PROCLIB that is used to start BDT. It is optional.
- *id* is the installation-defined identifier used with F to identify BDT.
- SY(*node-name*) is the system parameter that indicates which BDT system should process the transaction. Include the parentheses as shown. There is an installation-defined default for this parameter.

Transaction Definition

The transaction definition contains the information that BDT needs to copy the data set. A transaction definition consists of three sections: the **job definition** section, the **FROM** section, and the **TO** section.

The Job Definition Section

The job definition section begins with a transaction code. The transaction code in Figure 1 on page 3 is "Q". A transaction code of "Q" tells BDT that the transaction definition is wholly contained within the transaction. Other transaction codes tell BDT that the transaction definition is stored in a library. These codes are explained in Chapter 2, "Storing and Reusing Transaction Definitions — GMJD Libraries," on page 9.

After the transaction code, you can add optional job definition parameters.

Some of the things you might do with job definition parameters are:

- Give the transaction a name (with the JOBNAME parameter)
- Put the transaction on hold (with the HOLD parameter)
- Assign the transaction a particular priority (with the PRIORITY parameter)
- Assign the transaction a maximum amount of processor time (with the TIME parameter).

A complete list of the job definition parameters is included in Chapter 5, "File-to-File Transaction Parameter Reference," on page 23.

You can put job definition parameters anywhere in the transaction definition — you do not have to keep them together after the transaction code. For example, you might put some job definition parameters immediately after the transaction code, some with the FROM parameters, and some with the TO parameters. However, you may find it easier to keep track of the job definition parameters if you group them together.

The FROM Section

The FROM section describes the data set to be copied, that is, the "from" data set. It includes information BDT needs to locate, allocate, and deallocate the "from" data set. The FROM section is made up of four essential parameters and then additional parameters as required. The four essential parameters take part in all transactions whether you specify values for them or accept defaults. The additional parameters include optional parameters that you use to meet your needs and parameters that are required by BDT for certain tasks.

Four Essential FROM Parameters

The four FROM parameters that take part in all transactions are shown in Figure 1 on page 3. They are:

- The **FROM** parameter. This begins the FROM section. It tells BDT that the parameters that follow describe the "from" data set.

  BDT assumes that any parameter that is not a job definition parameter and is not in the TO section is a FROM parameter. Therefore, you only have to include the FROM parameter in your transaction if you list parameters describing the "from" data set after the TO section of your transaction.
- The **DATASET** parameter. This parameter identifies the data set to be copied. The format of this parameter is DATASET(*ds-name*). Substitute a data set name for *ds-name*. You must include this parameter in your transaction.
- The **LOC** or **LOCATION** parameter. This parameter identifies the node at which BDT is to find the "from" data set. The format of this parameter is LOCATION(*node-name*). Substitute a node name for *node-name*. You must include this parameter in your transaction unless:
  - You are in a single-BDT system and the data set is at your node
  - You are in a poly-BDT system and the data set is at the node that is defined as the default for your installation.
- The **DAP** parameter. This parameter specifies whether BDT is to use the PDS or SEQ dynamic application program (DAP) to copy the data set. The format of this parameter is DAP(*dap*). SEQ is the default for this parameter.
  - To use the SEQ DAP substitute SEQ for *dap* or omit the DAP parameter.
  - To use the PDS DAP substitute PDS for *dap*.

You may put the DAP parameter in either the FROM or TO section of your transaction.

Other FROM Parameters

Some of the things you might do with other FROM parameters are:

- Request shared or exclusive use of the "from" data set (with the OLD or SHR parameter)
- Request that the "from" data set be deleted, kept, or cataloged (with the DISP parameter)
- Specify the member of a PDS to be copied (with the MEMBER parameter)
- Provide the password for a password-protected data set (with the PASSWORD parameter).

Organizing the FROM Parameters

The FROM parameter, in cases when it is required, must be the first parameter in the FROM section. The other FROM parameters may be in any order.

A complete list of FROM parameters, including those required for particular tasks, is included in Chapter 5, "File-to-File Transaction Parameter Reference," on page 23.

The TO Section

The TO section describes the data set that is to receive the data, that is, the "to" data set. It includes information BDT needs to locate, allocate, and deallocate the "to" data set, as well as other information you might require.

Like the FROM section, the TO section is made up of essential parameters that take part in every transaction, and then additional parameters to accomplish particular tasks.

Three Essential TO Parameters

The three TO parameters that are take part in every transaction are shown in Figure 1 on page 3. They are:

- The **TO** parameter. This begins the TO section. It tells BDT that the parameters that follow describe the "to" data set. You must include the TO parameter in your transactions.
- The **DATASET** parameter. This parameter identifies the data set that is to receive the "from" data set. You should include the DATASET parameter in your transactions, unless you use either the DUMMY or INTRDR parameter. The DUMMY parameter specifies a dummy "to" data set, and the INTRDR parameter specifies the MVS internal reader at the receiving node as the "to" data set.
- The **LOC** or **LOCATION** parameter. This parameter identifies the node at which BDT is to find the "to" data set. You must include this parameter unless:
  - You are in a single-BDT system and the data set is at your node
  - You are in a poly-BDT system and the data set is at the node that is defined as the default for your installation.

Other TO Parameters

Things you might do with other parameters in the TO section include:

- Specify that the "to" data set is a new data set to be allocated (with the NEW parameter)
- Request RACF protection for the "to" data set (with the PROTECT parameter)
- Assign an expiration date to the "to" data set (with the EXPDT parameter).

Organizing the TO Parameters

The TO section must begin with the TO parameter. The other parameters in the TO section may be in any order.

A complete list of the TO parameters, including those required for specific tasks, is included in Chapter 5, "File-to-File Transaction Parameter Reference," on page 23.

How to Punctuate a Transaction

You must follow some punctuation rules when writing a transaction:

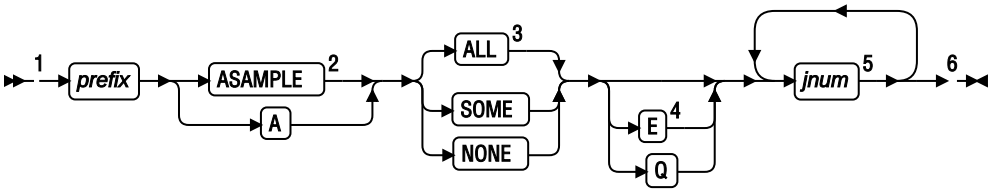
- Use one or more blank spaces to separate the transaction prefix, the transaction code, and transaction parameters. There is one exception to this: If you use the *bdt-char* prefix, do not separate it from the transaction code with a blank space.
- Use either uppercase or lowercase letters.
- Include parentheses when they are shown in the diagrams in this book. Parameters that include parentheses are called *keyword parameters*. Substitute a value for the variable in parentheses. Do not leave a space between the keyword and the first parenthesis. For example, type DATASET(MYDATA).

  In Figure 1 on page 3, DATASET, LOC, and DAP are all keyword parameters.

How to Read the Transaction Parameter Syntax Diagrams

Each BDT transaction parameter is described in this book with a syntax diagram. The syntax diagram shows the parts and punctuation of the parameter. When using a syntax diagram, follow the diagram from left to right, choosing the path that suits your needs.

Figure 2 on page 6 shows a sample syntax diagram and explains how to read it.



Notes:

[1] Start here.
[2] Choose either ASAMPLE or A.
[3] Choose one of the options. The default is always atop of the main line. In this case, ALL is the default.
[4] Choose E, Q, or neither.
[5] Repeat *jnum* any number of times. Variables are always in italics.
[6] End here.

Figure 2: Sample Syntax Diagram for a Transaction Parameter

How to Submit a Transaction

You may submit a transaction interactively, or via ISPF panels (at TSO terminals only), or in a batch job.

In general you may submit a transaction from any TSO terminal, MCS console, or JES3 console. However, your installation may be configured in a way that prevents you from using certain consoles or terminals for file-to-file transactions. Check with your system programmer to find out which consoles or terminals can be used for file-to-file transactions.

How to Submit a Transaction Interactively

To submit a transaction interactively at your terminal or console, simply type in the transaction, following the rules explained in this chapter. The transaction may be complete in itself, or it may refer to a transaction definition stored in a GMJD library. GMJD libraries are explained in Chapter 2, “Storing and Reusing Transaction Definitions — GMJD Libraries,” on page 9.

How to Submit a Transaction via ISPF Panels

TSO users may be able to use ISPF panels to submit transactions. Your BDT system programmer can tell you whether these panels are available to you and, if they are, how to invoke them. Instructions for using the panels are provided by the panels themselves and by a help facility.

Note: When submitting a transaction through an ISPF panel, do not code an ampersand (&) within the transaction's parameter string. A transaction that contains an ampersand will be rejected.

How to Submit a Transaction in a Batch Job

You can submit one or several transactions in a batch job. The transactions may be complete in themselves or may refer to transaction definitions stored in a GMJD library.

Use this format for entering transactions in a batch job:

```
//job card
//  EXEC PGM=BDTBATCH
//SYSPRINT DD DUMMY
//SYSIN DD *
transaction
/EOT
transaction
/EOT
```

When using a batch job:

- Do not use a prefix with the transaction. Start with the transaction code.
- Begin the transaction in any column.
- Continue the transaction on the next line(s) if necessary.
- Do not split a parameter across two statements. Splitting a parameter across statements causes unpredictable results.
- Put each transaction on a separate line, and separate the transactions by an /EOT statement.

Example

Submit two transactions from a batch job. The first transaction (MEM20) refers to a transaction definition in the system GMJD library. The second transaction is complete in itself.

```
//BATCHJOB  JOB CLASS=A
//          EXEC PGM=BDTBATCH
//SYSPRINT  DD DUMMY
//SYSIN     DD *
   MEM20
/EOT
   Q JOB(BDTJOB) ACCT(MY.DEPT,MY.PROJ)
      FROM DATASET(MY.DATA) LOC(MYNODE) SHR DAP(SEQ)
      TO DATASET(OTHER.DATA) LOC(THATNODE) OLD
/EOT
```

How BDT Processes Transactions

After BDT accepts a transaction it creates a BDT job, which it places on the work queue. The job has a name (assigned when the transaction was submitted, or assigned by the BDT default) and a number (assigned by BDT). Once the job has been selected for processing, it is called an *active* job, whether or not the actual copying of data has begun.

BDT selects jobs from the BDT work queue based on requirements such as the job's priority. BDT job priorities range from 0 to 15, with 15 being the highest. BDT schedules high priority jobs before lower priority jobs.

If the job involves two nodes, it is managed and scheduled at the global node.

The copying of the data is performed by a BDT dynamic application program (DAP). Copying of sequential data sets uses the SEQ DAP; copying of partitioned data sets uses the PDS DAP.

After submitting a transaction, you may issue commands to change the way that BDT processes the transaction, or to inquire about the progress of the transaction. See *z/OS BDT Commands* for more information. BDT issues messages to keep you informed of the status of your transaction. The messages are explained in *z/OS BDT Messages and Codes*

How BDT Protects Transaction Processing

Each time you submit a transaction, the transaction travels from your address space to the BDT address space. In order to reach BDT, the transaction may have to travel from one processor to another or from one node to another. Without adequate protection, a transaction could become lost before reaching BDT. For example, if a transaction had to travel from one processor to another and the receiving processor were disabled, the transaction would be lost.

To reduce the possibility of losing a transaction, BDT uses the transaction queuing integrity (TQI) facility. When you submit a transaction, TQI makes a copy of the transaction in the TQI checkpoint data set. The transaction remains in the checkpoint data set until it has reached the BDT address space. If, for some reason, your transaction does not reach the BDT address space, BDT can recover it from the TQI checkpoint data set.

Chapter 2. Storing and Reusing Transaction Definitions — GMJD Libraries

This chapter is an introduction to GMJD libraries. GMJD libraries make it possible to store and reuse transaction definitions. This chapter explains:

- What GMJD libraries are
- How to use a system GMJD library created by your system programmer
- How to create and use private GMJD libraries
- How to alter transaction definitions stored in GMJD libraries.

What GMJD Libraries Are

GMJD (*generic master job definition*) libraries are data sets in which you can store transaction definitions. You can reuse these stored transaction definitions simply by naming them in your transactions. This can save you from having to type frequently used transaction definitions each time you use them. You can also add to or alter stored transaction definitions when you invoke them. There are two types of GMJD libraries: system GMJD libraries and private GMJD libraries.

How to Use System GMJD Libraries

System GMJD libraries are created for your installation by your system programmer. Your installation may have a system GMJD library; check with your system programmer.

The format for a transaction that uses a transaction definition stored in a system GMJD library is:

prefix name
prefix is the prefix that is appropriate for your console or terminal and installation, and *name* is the name of the transaction definition stored in the system GMJD library. See your system programmer for valid names.

How to Create and Use Private GMJD Libraries

Private GMJD libraries are GMJD libraries you create for your own use.

How to Create a Private GMJD Library

To create a private GMJD library, follow the two steps described subsequently.

1. Allocate and name a data set. The data set may be either sequential or partitioned. A sequential data set can store a single transaction definition. A partitioned data set can store several transaction definitions, each as a member of the data set.

   The data set, whether partitioned or sequential, must have the following characteristics:

   - Record format – fixed length or fixed length and blocked
   - Logical record length – 80 bytes
   - Block size – variable. If you have a system GMJD library, the block size of your data set may be no larger than the block size of the system GMJD library. If you do not have a system GMJD library, the block size of your data set may not exceed 6160 bytes.
2. Enter either transaction parameters or comments into each logical record in your data set, following these rules:

- Start the first parameter in a logical record in any column.
- Do not split a parameter across logical records, that is, do not start a parameter in one logical record and finish it in another.
- Separate parameters with one or more blanks.
- Mark comments by putting an asterisk in the first column of the logical record. Enter anything you wish following the asterisk.
- Use only columns 1 through 71 of each logical record for parameters or comments.
- If your are using a PDS, name the members with any valid PDS member name except Q, GMJD, GMJDLIB, or any BDT command word (C, CALL, CANCEL, DUMP, F, FAIL, I, INQUIRY, J, JES, MESSAGE, MODIFY, R, RESTART, RETURN, S, START, T, V, VARY, X, Z).

How to Use a Private GMJD Library

To use a transaction definition stored in a private GMJD library, enter a transaction with this format:

prefix code `GMJDLIB` (*gmjd-library-name*)

In this transaction:

- *prefix* is the prefix that is appropriate for your console or terminal at your installation.
- *code* is one of the following:
  - For sequential GMJD libraries, any 1-8 character value other than Q
  - For partitioned GMJD libraries, the name of the member of the partitioned data set that contains the stored transaction definition you are requesting.
- GMJDLIB specifies that the transaction refers to a transaction definition stored in a private GMJD library
- *gmjd-library-name* is the name of the GMJD library in which the transaction definition is stored.

Examples

1. Submit the transaction definition stored as member ENDREPT in your private GMJD library, which is a partitioned data set named JOESLIB.

   prefix `GMJDLIB(JOESLIB)`
2. Submit the transaction definition stored in the private GMJD library that is a sequential data set named REPORT.

   prefix `AUG GMJDLIB(REPORT)`

How to Modify Stored Transaction Definitions

When using a stored transaction definition you may add to, remove, or later the stored transaction definition's parameters. You do this by including the changes in the transaction that requests the stored transaction definition. The changes you make are effective only in the current transaction – they are not permanent changes in the stored transaction definition. The next time you invoke the stored transaction definition it will have all of its original parameters.

Figure 3 on page 11 shows how BDT merges transaction parameters.

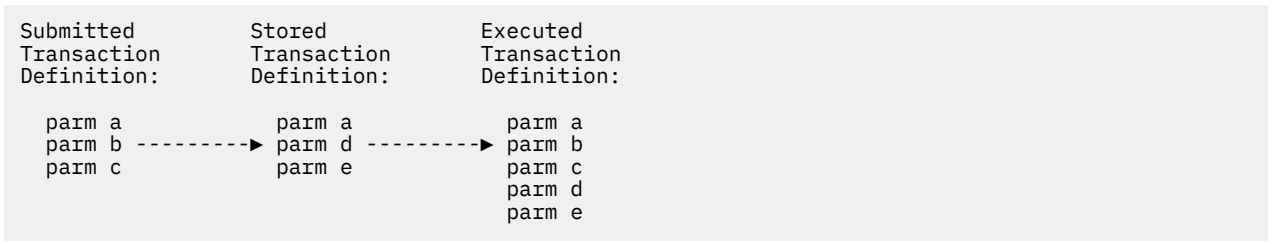


Figure 3: How BDT Merges Parameters

How to Add Parameters

To add parameters to a stored transaction definition, simply include the parameters to be added in the transaction that requests the stored transaction definition.

Example: You have stored a skeletal transaction definition for copying a sequential data set. The transaction definition is stored in a sequential data set named SEQTOSEQ. The definition has all the parameters you need except the data set names and locations. You want to send the data set MONTHEND from your node to the data set REPT06 at the node HDQTR.

You use the TSO prefix (BDT). Because the transaction definition is stored in a sequential data set, you can use any string of up to 8 characters for the code; you use STS.

```
BDT STS GMJDLIB(SEQTOSEQ) FROM DATASET(MONTHEND)
        TO DATASET(REPT06) LOC (HDQTR)
```

Note that since the LOCATION parameter defaults to the node at which the transaction is submitted (your node), you did not need to use the LOCATION parameter in the FROM section.

How to Remove Parameters

To remove parameters that are included in a stored transaction definition, use the / character before the parameter you want to remove.

Example: You want to use the transaction definition that is stored as FASTJOB. However, in addition to the parameters that you want, FASTJOB includes the PRTY parameter, which you don't want. The PRTY parameter assigns a special priority to a job, and you want your job to run at the priority the BDT system would normally assign to jobs. So, when you invoke FASTJOB you remove the PRTY parameter. You use the TSO prefix (BDT).

```
BDT FASTJOB /PRTY
```

How to Alter Parameters

To alter parameters that are included in a stored transaction definition, include the parameters with the new values in the transaction that requests the stored transaction definition.

Example: You want to submit the transaction definition stored as THATJOB, but you want to change the data set named in the TO section of THATJOB. You want the new “to” data set to be the data set named NEWDATA. You use the TSO prefix (BDT).

```
BDT THATJOB TO DTASET(NEWDATA)
```

Note that since the parameter you wanted to changed, DATASET, appears in both the TO and FROM sections of the stored transaction definition, you have to use the TO parameter as well as the DATASET parameter to identify which occurrence of DATASET you want to change.

Chapter 3. Controlling the Order in Which Transactions Are Processed — DTC Networks

This chapter is an introduction to DTC networks. DTC networks make it possible to control the order in which transactions are processed. This chapter explains:

- What DTC networks are
- What parameters you need to create and use DTC networks
- How to create DTC networks
- How to specify the order of transactions in a DTC network
- How to make sure the transactions in a DTC network run in order
- How to tell BDT what to do with a transaction in a DTC network if its predecessor transaction fails
- How to make sure the first transaction in a DTC network doesn't complete before you can submit the second one.

At the end of the chapter is an example of transactions that create and define a DTC network.

What DTC Networks Are

Dependent transaction control (DTC) networks are groups of transactions that you create. These networks allow you to control the order in which the transactions are processed.

Suppose that you have several transactions that copy data to the same data set, and you want to add the data to the receiving data set in a particular order. A DTC network will allow you to do that. Or, suppose you want to break a big transaction into smaller transactions. A DTC network will allow you to do that as well.

It is important to remember that the BDT dependent transaction control (DTC) network facility and the JES3 dependent job control (DJC) network facility are two different network facilities. Do not attempt to use commands intended for one facility with the other facility.

What Parameters You Need

To create and control DTC networks, you need five parameters. You include these parameters in your transactions. The parameters are:

- NETID
- NETREL
- JOBNAME
- NETHOLD
- NETCOND

Each of these parameters are described subsequently.

How to Identify the Transactions That Belong to a DTC Network — the NETID Parameter

To create a DTC network, you identify each transaction that belongs to the network with the NETID parameter. The format of the NETID parmater is NETID(*netid*), where *netid* is the name you assign to the network.

For more information on using the NETID parameter, see Chapter 5, “File-to-File Transaction Parameter Reference,” on page 23.

How to specify the order of transactions — the NETREL and JOBNAME parameters

To control the order in which transactions in a DTC network are processed, you specify the *successor* transactions in each transaction. A successor transaction is a transaction that cannot be processed until it is released by another transaction, called a *predecessor* transaction. You specify the successor transactions to a particular transaction with the NETREL parameter. The format of the NETREL parameter is NETREL(*name*) where *name* is the name or names of the transactions that are successors to this one. You assign names with the JOBNAME parameter. The format of the JOBNAME parameter is JOBNAME(*name*).

Figure 4 on page 14 shows the successor and predecessor transactions in a simple DTC network.

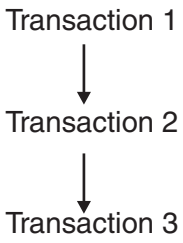


Figure 4: A simple dependent transaction control (DTC) network

In Figure 4 on page 14, transaction 2 is a successor to transaction 1 and a predecessor to transaction 3. Transaction 1 releases transaction 2, which releases transaction 3.

Figure 5 on page 14 shows a more complex DTC network — a network in which transactions are successors and predecessors to more than one transaction.

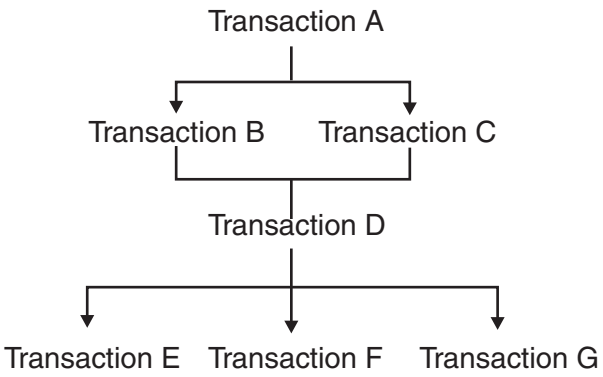


Figure 5: A more complex dependent transaction control (DTC) network

In Figure 5 on page 14, transaction D has two predecessors (transaction B and transaction C) and three successors (transaction E, transaction F, and transaction G). Transaction A releases transaction B and transaction C. It does not matter whether transaction B or transaction C runs next after transaction A, but both transaction B and transaction C must run after transaction A and before transaction D. Transaction D releases transaction E, transaction F, and transaction G.

NETREL Can Only Release Successors at the Same Node

The NETREL parameter can only release a successor transaction if it is scheduled for processing at the same node as its predecessor. You must be aware of this when constructing DTC networks. In particular, you should be aware of the global-local relationships of the nodes in your network.

Global and local nodes

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. A given node may be the global node in relationships with some nodes and the local node in relationships with other nodes. File-to-file transactions are always scheduled at the global node, regardless of whether they are submitted at the global or local node.

Let's look at a DTC network that does not take the global-local relationship of nodes into account. The complex for this example consists of two BDT nodes, node 1 and node 2. Node 1 is the global node; node 2 is the local node, as shown in Figure 6 on page 15.

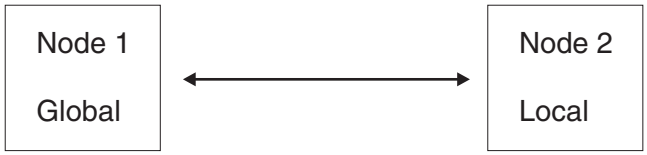


Figure 6: BDT nodes in a sample complex

The following DTC network of two transactions is submitted at node 2, the local node:

```
BDT Q JOBNAME(TRAN1)  NETID(NETA)  NETREL(TRAN2)
      FROM  LOC(NODE2) DATASET(DATA3)  DAP(SEQ)
      TO    LOC(NODE2) DATASET(DATA4)


BDT Q JOBNAME(TRAN2)  NETID(NETA)  NETHOLD(1)
      FROM  LOC(NODE1) DATASET(DATA1) DAP(SEQ)
      TO    LOC(NODE2) DATASET(DATA2)
```

TRAN1 will be scheduled for processing at node 2. The transaction identified by the JOBNAME parameter as TRAN1 copies data within node 2 (as indicated with the LOC parameters); the global node in this complex (node 1) is not involved. When a transaction copies data within a node, that node acts as both the global and the local node for the transaction.

TRAN2 will be scheduled for processing at node 1. The transaction identified by the JOBNAME parameter as TRAN2 copies data from node 1, the global node, to node 2, the local node. TRAN2 will be scheduled for processing at node 1, the global node.

TRAN1 cannot release TRAN2. The NETREL parameter in TRAN1 specifies that TRAN1 is to release TRAN2 when it completes. However, TRAN1 cannot release TRAN2 because they will be scheduled for processing at different nodes.

For more information on using the JOBNAME and NETREL parameters, see Chapter 5, "File-to-File Transaction Parameter Reference," on page 23.

How to Make Sure Transactions Run in Order — the NETHOLD Parameter

To ensure that a transaction with several predecessors runs only after all the predecessors have run, rather than after only one predecessor has run, you give transactions *hold counts*. Hold counts keep track of how many predecessors must be processed before successor transactions can run. You assign hold counts with the NETHOLD parameter. The format of the NETHOLD parameter is NETHOLD(*net-hold*) where *net-hold* is the hold count. Most of the time, the hold count you assign to a transaction will be equal to the number of predecessors for that transaction.

In Figure 5 on page 14, transaction D would be given a hold count of 2 to ensure that it would run only after both transaction B and transaction C. A hold count of 2 means that the transaction requires the completion of 2 predecessor transactions to be released. Each of the other transactions in the network, except transaction A, would be given a hold count of 1, because each requires the completion of only 1 predecessor transaction to be released. As the first transactions in the network, transaction A is not a successor to any other transactions and so would not ordinarily be given a hold count.

For more information on using the NETHOLD parameter, see Chapter 5, "File-to-File Transaction Parameter Reference," on page 23.

How to Tell BDT What to Do If Predecessor Transactions Fail — the NETCOND Parameter

Because transactions in a DTC network depend on each other to be processed, you may want to tell BDT what to do with a successor transaction if a predecessor transaction fails. You do this with the NETCOND parameter. The format of the NETCOND parameter is NETCOND(*option*,*option*). The first option tells BDT what do with this transaction if a predecessor completes normally. The second option tells BDT what to do with this transaction if a predecessor fails. The things you can tell BDT to do with this transaction are:

- Decrease the transaction's hold count by 1 (which will allow it to be released if its hold count was originally 1)
- Leave the hold count of the transaction unchanged (to keep it on hold)
- Flush (cancel) the transaction and its successors.

For more information on using the NETCOND parameter, see Chapter 5, "File-to-File Transaction Parameter Reference," on page 23.

How to Make Sure the First Transaction Doesn't Complete Before You Can Submit the Second One

Unless you construct a DTC network carefully, the first transaction in the network can complete before you submit the second transaction. If that happens, the rest of the network will never run, as the first transaction cannot release its successors.

To prevent this kind of "runaway" first transaction in a DTC network, you can:

- Give the first transaction in the network a hold count of 1, even though it has no predecessors. You can do this with the NETHOLD parameter. Then, to release the first transaction use the F,NET,ID,J,D command or the F,NET,ID,J,R command. See *z/OS BDT Commands* for more information about these commands.
- Put the first transaction in the network into operator hold. You can do this by including the HOLD parameter in the transaction, or by issuing the F,J,H command after you submit the transaction. Then, use the F,J,R command to release the transaction from operator hold. See *z/OS BDT Commands* for more information on the F,J,H and F,J,R commands.
- Submit the transactions in the network "bottoms up", that is, submit the last transaction first and the first transaction last.

Example

Using the NETID and JOBNAME parameters in each of three transactions, create a DTC network named MYNET which consists of MYTRAN1, MYTRAN2, and MYTRAN3.

Make MYTRAN1 the first transaction in the network. Using the NETREL parameter, specify that the successor to MYTRAN1 is MYTRAN2, and that the successor to MYTRAN2 is MYTRAN3.

Using the NETHOLD parameter, assign a hold count of 1 to MYTRAN2 and a hold count of 1 to MYTRAN3. These hold counts are equal to the number of predecessor transactions for MYTRAN2 and MYTRAN3.

Using the NETCOND parameter, specify a conditional treatment of MYTRAN3, depending on the outcome of MYTRAN1 and MYTRAN2. Specify that:

- If MYTRAN2 completes normally, the hold count of MYTRAN3 is to be decreased by 1, releasing MYTRAN3

- If either MYTRAN1 or MYTRAN2 fails, MYTRAN3 is to be canceled (flushed).

Submit the transactions in reverse order to keep MYTRAN1 from completing before MYTRAN2 and MYTRAN3 are submitted.

```
prefix Q JOBNAME(MYTRAN3) NETID(MYNET) NETHOLD(1) NETCOND(D,F)
           FROM DATASET(DDATA) LOCATION(ANODE) DAP(SEQ)
           TO DATASET(BDATA) LOCATION(BNODE) MOD

prefix Q JOBNAME(MYTRAN2) NETID(MYNET) NETHOLD(1) NETREL(MYTRAN3)
           FROM DATASET(CDATA) LOCATION(ANODE) DAP(SEQ)
           TO DATASET(BDATA) LOCATION(BNODE) MOD

prefix Q JOBNAME(MYTRAN1) NETID(MYNET) NETREL(MYTRAN2)
           FROM DATASET(ADATA) LOCATION(ANODE) DAP(SEQ)
           TO DATASET(BDATA) LOCATION(BNODE) MOD
```

Chapter 4. Sample Transactions

This chapter contains samples of several useful transactions. These samples include the required parameters and a few optional parameters.

The transaction prefixes and the FROM, DATASET, LOCATION, DAP, and TO parameters that are used in the samples are explained in Chapter 1, “Writing and Submitting File-to-File Transactions,” on page 1. Brief explanations of all other parameters used in the samples are included with the samples. For more information on the parameters shown in the samples, see Chapter 5, “File-to-File Transaction Parameter Reference,” on page 23.

If you use these samples, substitute values for the variables shown in italic type.

Copy a Data Set at Your Node to Another Data Set at Your Node

```
prefix Q FROM DATASET(ds-name) DAP(dap) OLD
TO DATASET(ds-name) OLD
```

Because the LOCATION parameter defaults to your node, you can omit the LOCATION parameter in both the TO and FROM sections.

The OLD parameter gives BDT exclusive use of an existing data set. It is optional.

Copy a Data Set from Your Node to Another Node

```
prefix Q FROM DATASET(ds-name) OLD DAP(dap)
TO DATASET(ds-name) LOCATION(node-name) OLD
```

Because the LOCATION parameter defaults to your node, you can omit the LOCATION parameter in the FROM section.

The OLD parameter gives BDT exclusive use of an existing data set. It is optional.

Copy a Data Set from Another Node to Your Node

```
prefix Q FROM DATASET(ds-name) LOCATION(node-name) OLD
DAP(dap)
TO DATASET(ds-name) OLD
```

Because the LOCATION parameter defaults to your node, you can omit the LOCATION parameter in the TO section.

The OLD parameter gives BDT exclusive use of an existing data set. It is optional.

Copy a Sequential Data Set to Another Sequential Data Set

```
prefix Q FROM DATASET(ds-name) LOCATION(node-name) SHR
TO DATASET(ds-name) LOCATION(node-name) OLD
```

Because the DAP parameter defaults to SEQ, you can omit the DAP parameter.

The SHR parameter gives BDT shared use of an existing data set. It is optional.

The OLD parameter gives BDT exclusive use of an existing data set. It is optional.

Copy a Partitioned Data Set to Another Partitioned Data Set

```
prefix Q FROM DATASET(ds-name) LOCATION(node-name) SHR DAP(PDS)
TO DATASET(ds-name) LOCATION(node-name) OLD
```

The SHR parameter gives BDT shared use of an existing data set. It is optional.

The OLD parameter gives BDT exclusive use of an existing data set. It is optional.

Copy a Sequential Data Set to the End of an Existing Sequential Data Set

```
prefix Q FROM DATASET(ds-name) LOCATION(node-name) SHR
TO DATASET(ds-name) LOCATION(node-name) MOD
```

Because the DAP parameter defaults to SEQ, you can omit the DAP parameter.

The SHR parameter gives BDT shared use of an existing data set. It is optional.

The MOD parameter specifies that the copied data is to be added to the end of an existing data set.

Copy a Data Set to a New Sequential DASD Data Set

```
prefix Q FROM DATASET(ds-name) LOCATION(node-name) SHR
TO DATASET(ds-name) LOCATION(node-name)
NEW SPACE(primary,secondary) TRACKS BLKSIZE(blk-size)
LRECL(length) RECFM(rf) UNIT(3380)
DISP(CATLG,DELETE) VOLUME(111111)
```

Because the DAP parameter defaults to SEQ, you can omit the DAP parameter.

The SHR parameter gives BDT shared use of an existing data set. It is optional.

The NEW, SPACE, TRACKS, BLKSIZE, LRECL, and RECFM parameters define the new “to:” data set.

The UNIT parameter specifies the device on which the data set is to be stored.

The DISP parameter specifies that the data set is to be cataloged if the transaction completes normally, and deleted if the transaction completes abnormally. It is optional.

The VOLUME parameter specifies the serial number of the volume on which the data set is stored.

Copy a Data Set to a New Partitioned DASD Data Set

```
prefix Q FROM DATASET(ds-name) LOCATION(node-name) SHR DAP(PDS)
TO DATASET(ds-name) LOCATION(node-name)
NEW SPACE(primary,secondary) TRACKS BLKSIZE(blk-size)
LRECL(length) RECFM(rf) DSORG(PO) UNIT(3380)
DIR(blocks) DISP(CATLG,DELETE) VOLUME(111111)
```

The SHR parameter gives BDT shared use an existing data set. It is optional.

The NEW, SPACE, TRACKS, BLKSIZE, LRECL, RECFM, DSORG and DIR parameters define the new “to:” data set.

The UNIT parameter specifies the device on which the data set is to be stored.

The DISP parameter specifies that the data set is to be cataloged if the transaction completes normally and deleted if the transaction completes abnormally.

The VOLUME parameter specifies the serial number of the volume on which the data set is stored.

Copy a Tape Data Set to a New Tape Data Set

```
prefix Q FROM DATASET(ds-name) LOCATION(node-name) DAP(SEQ)
TO DATASET(ds-name) LOCATION(node-name)
LRECL(length) RECFM(rf) BLKSIZE(blk-size)
UNIT(TAPE) VOLUME(vol-ser) LABEL(SL) NEW
```

The NEW, LRECL, RECFM, and BLKSIZE parameters define the new “to:” data set.

The UNIT parameter indicates the device on which the data set is to be stored.

The VOLUME parameter identifies the tape volume on which the data set is stored.

The LABEL parameter supplies a label for the data set. It is optional.

Copy a DASD Data Set to a New Tape Data Set

```
prefix Q FROM DATASET(ds-name) UNIT(3350) VOLUME(vol-ser)
LOCATION(node-name) DAP(SEQ)
TO DATASET(ds-name) LOCATION(node-name)
NEW UNIT(TAPE) VOLUME(111111) LABEL(label)
RECFM(rf) LRECL(length)
BLKSIZE(blk-size)
```

The NEW, LRECL, RECFM, and BLKSIZE parameters define the new “to:” data set.

The UNIT parameter indicates the device on which the data set is stored.

The VOLUME parameter identifies the volume on which the data set is stored.

The LABEL parameter supplies a label for the data set. It is optional.

Copy a Cataloged Tape Data Set to a DASD Data Set

```
prefix Q FROM DATASET(ds-name) LOCATION(node-name)
DAP(SEQ) UNIT(TAPE)
TO DATASET(ds-name) LOCATION(node-name)
UNIT(3350) OLD
```

The SHR parameter gives the transaction shared use of an existing data set. It is optional.

The OLD parameter gives BDT exclusive use of an existing data set. It is optional.

The UNIT parameter indicates the device on which the data set is stored. It is optional.

Chapter 5. File-to-File Transaction Parameter Reference

This chapter describes the parameters you use to write transactions. It includes:

- A summary of the transaction format and parameters.
- A list of the transaction prefixes.
- A list of parameters required in particular circumstances.
- A description of each of the transaction parameters. The parameters are in alphabetic order. Each parameter begins a new page. The description of each parameter explains the purpose, rules for use, and format of the parameter. In addition, each description includes brief usage notes, examples, and defaults for the parameter, if any.

  Note: The defaults described in this book are those provided by IBM. Your installation may change these defaults or provide additional defaults. Check with your system programmer for more information.

If you need more general information about transactions, or do not understand terms used in this chapter, see Chapter 1, “Writing and Submitting File-to-File Transactions,” on page 1.

Transaction format and parameter summary

The preceding section contains the parts of a transaction.

Prefix	Job Definition	FROM Parameters	TO Parameters
prefix			
(See “Transaction Prefixes” on page 25.)	Q ACCT CSOPT GMJDLIB HOLD JOBNAME MSGCLASS NETCOND NETHOLD NETID NETREL PRIORITY PROGRAMMER SYSTEM TIME	FROM DAP DATASET LOCATION BDTENQ BLKSIZE DEN DISP DUMMY LABEL LRECL MEMBER OLD PARALLEL PARMS PASSWORD POSITION SECGROUP SECPSWD SECUSER SHR TRTCH UNIT VOLREF VOLSEQ VOLUME	TO DATASET LOCATION ALX BDTENQ BLKSIZE BLOCK BUFL CONTIG CYLINDERS DAP DCBDS DEN DIAGNS DIR DISP DSORG DUMMY EXPDT INTRDR LABEL LRECL MAXVOL MEMBER MOD MSGVP MXIG NEW OLD PARALLEL PARMS PASSWORD POSITION PROTECT RECFM RELEASE RETPD ROUND SECGROUP SECUSER SECPSWD SHR SPACE TRACKS TRTCH UCOUNT UNIT VOLREF VOLSEQ VOLUME

Required Parameters

The parameters in "Transaction format and parameter summary" on page 23 take part in every transaction, although you can accept defaults for FROM, DAP, and LOCATION.

Other parameters are required in particular circumstances, as explained in the following table. Your installation may provide defaults which allow you to omit some of the parameters shown in the table.

If the Transaction Involes a Data Set That Is:	Then the Transaction Must Include These Additional Parameters:
New	NEW, BLKSIZE, LRECL, RECFM, UNIT, VOLUME; also, SPACE, and either BLOCK, CYLINDERS, or TRACKS for DASD data sets; also, DIR for partitioned data sets
Without an explicit block size	BLKSIZE
Labeled with other than a standard IBM label	LABEL
Unlabeled	BLKSIZE, LRECL
Stored on more than five volumes	MAXVOL
Stored on a mass storage system	MSVGP
Protected with a password	PASSWORD
Uncataloged	RECFM and UNIT; also, POSITION for data sets stored on a tape that contains several data sets

Transaction Prefixes

Table 3: Transaction Prefixes

Type of Console	Prefix in a Single-BDT Complex	Prefix in a Poly-BDT Complex
MCS	*bdt-char* or BDT or F [*bdt-proc.*]*bdt-id*	same
JES3	*S,BDT	*S,BDT,SY(*node-name*)
TSO	BDT	BDT SY(*node-name*)

In Table 3 on page 25:

- *bdt-char* is a special character (such as +) defined by your system programmer that routes the transaction directly to a specific system. Do not use a delimiter (a blank space or a comma) after *bdt-char*.
- *bdt-proc* is the name of the cataloged procedure in SYS1.PROCLIB that is used to start BDT. It is optional.
- *bdt-id* is the installation-defined identifier of BDT.
- SY(*node-name*) is the system parameter. See "SYSTEM — Specify the BDT Node in a Poly-BDT Complex" on page 74.

ACCT — Supply Accounting Information

Use this parameter to supply accounting information with your transaction.

Rules

Optional or Required: Optional

Section: Job definition, FROM, TO

DAP: SEQ, PDS

Format

acct-info
is the accounting information you want to supply. This information may be no more than 142 characters in length. Any commas used count in the total number of characters.

Usage Note

Accounting information is read beginning with the first column. If there are leading blanks in the accounting information, the blanks are read as part of the information.

Example

Supply your ID, department number, and project number in your transaction. Use the TSO prefix (BDT).

```
BDT Q ACCT(BRK,73A,35748)
      FROM DATASET(ADATA) LOCATION(KGN01) DAP(SEQ)
      TO DATASET(DATAFILE) LOCATION(KGN02)
```

ALX — Allocate Space in Several Contiguous Areas

Use this parameter with the SPACE parameter when allocating a new “to” data set to be stored on a direct access storage device (DASD). ALX specifies that MVS may satisfy the primary allocation request of the SPACE parameter by allocating up to five different contiguous areas of space. The size of each area must be equal to or greater than the amount of primary space requested.

Rules

Optional or Required: Optional

Also Requires: The NEW and SPACE parameters, and either the BLOCK, CYLINDERS, or TRACKS parameter

Section: TO

DAP: SEQ, PDS

Format

Usage Notes

1. Specify the units of space to be allocated with the BLOCK, CYLINDERS, or TRACKS parameter.
2. This parameter is invalid with the CONTIG or MXIG parameters.

3. If you code this parameter in the FROM section of your transaction, BDT will accept the parameter but will not use it.
4. For a list of parameters that can be used to allocate a new data set, see Appendix A, “Parameters That Require Other Parameters,” on page 83.

Example

Copy the data set MYDATA.A1 to the new data set YOURDATA.A1. Specify that the primary allocation request on the SPACE parameter be satisfied with several contiguous areas of space. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(MYDATA.A1) LOCATION(KGN01) DAP(SEQ)
       TO DATASET(YOURDATA.A1) LOCATION(KGN03) NEW SHR
       ALX SPACE(1,1) CYLINDERS UNIT(SYSDA)
       BLKSIZE(800) LRECL(80) RECFM(F)
       DSORG(PS) VOLUME(BDTDS8)
```

BDTENQ — Request Shared or Exclusive Use of a Data Set by a BDT Transaction

Use this parameter to request shared use or exclusive use of the “from” or “to” data sets by a BDT transaction. Shared use permits other BDT transactions to read a data set while a transaction is being processed. Exclusive use prevents other BDT transactions from reading or writing to a data set while a transaction is being processed.

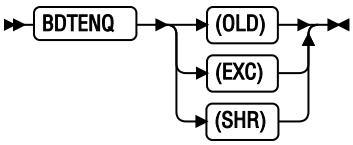
Rules

Optional or Required: Optional

Section: FROM, TO

DAP: SEQ, PDS

Format



SHR
requests shared use of the data set. Other BDT transactions may read from (but not write to) the data set while your transaction is reading the data set.

If you do not specify the BDTENQ parameter, and the MVS status is SHR, SHR is assumed for the “from” data set.

OLD or EXC
requests exclusive use of the data set. No other BDT transaction can read from or write to the data set until your transaction is completed.

If you do not specify the BDTENQ parameter, OLD or EXC is assumed for the “to” data set. If you do not specify the BDTENQ parameter, and the MVS status is not share, OLD or EXC is assumed for the “from” data set.

Usage Notes

1. For a partitioned data set, the BDTENQ status applies to the entire data set, not to individual members.
2. You may want to specify SHR for the “from” data set. This allows multiple transactions to read (but not write to) the data set concurrently.

3. You may want to specify OLD or EXC for the “to” data set. This prevents other transactions from writing to the data set until your transaction completes.

Example

Submit two transactions that should be able to run at the same time. The first is to copy the member MEMA from the partitioned data set PDS1.SET, located at node THISNODE, to the member MEMB of the partitioned data set PDS2.SET at node ELSEWHER. The second is to copy the member MEMC of the data set PDS3.SET, located at node ANYWHERE, to member MEMD of the data set PDS1.SET. Use the TSO prefix (BDT).

```
BDT Q FROM DATASET(PDS1.SET) LOCATION(THISNODE) DAP(SEQ)
      MEMBER(MEMA) BDTENQ(SHR)
      TO DATASET(PDS2.SET) MEMBER(MEMB) LOCATION(ELSEWHER)

BDT Q FROM DATASET(PDS3.SET) MEMBER(MEMC) LOCATION(ANYWHERE)
      DAP(SEQ)
      TO DATASET(PDS1.SET) MEMBER(MEMD) LOCATION(THISNODE)
      OLD BDTENQ(SHR)
```

BLKSIZE — Define the Block Size of a Data Set

Use this parameter to define the block size of the “from” or “to” data sets.

Rules

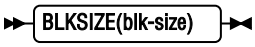
Optional or Required: Required for:

- Data sets that are new
- Data sets that are unlabeled, or when you want to bypass label processing
- Data sets with no explicit block size

Section: FROM, TO

DAP: SEQ, PDS

Format



blk-size
is the block size of the data sets. It is a number from 1 through 32760. The default for existing data sets is the block size in the data set label.

Usage Notes

1. You can use this parameter to reblock and reformat a data set by giving the “to” data set in a transaction a different block size than the “from” data set.
2. Do not use this parameter in the FROM section unless the “from” data set is unlabeled or label processing is bypassed.
3. For more information about the use of BLKSIZE, see the discussion about the DCB BLKSIZE parameter in the JCL manual that is appropriate for your installation.

Example

Copy the data set BIGDATA to the new data set JUNEDATA. Define the block size of JUNEDATA as 800. Use the TSO prefix (BDT).

```
BDT Q FROM DATASET(BIGDATA) LOCATION(SPK01) DAP(SEQ) SHR
      TO DATASET(JUNEDATA) LOCATION(SPK02)
      NEW SPACE(1,1) CYLINDERS UNIT(3380)
```

```
       BLKSIZE(800) LRECL(80) RECFM(F)
       VOLUME(111111)
```

BLOCK — Allocate Space in Block Units

Use this parameter when defining a new "to" data set that is to be stored on a direct access storage device (DASD). BLOCK requests that MVS allocate space for the data set in block units.

Rules

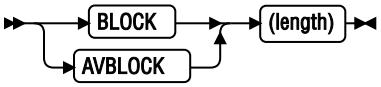
Optional or Required: Optional

Also Requires: The NEW and SPACE parameters

Section: TO

DAP: SEQ, PDS

Format



length
defines the average length of a block of data. *length* can be any number from 1 through 32760.

Usage Notes

1. This parameter is invalid with the CYLINDERS or TRACKS parameters.
2. For a list of parameters that can be used to allocate a new data set, see Appendix A, "Parameters That Require Other Parameters," on page 83.

Example

Allocate direct access space for a new "to" data set in block units. Specify that the average length of a block of data will be 3120. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(FILEDATA) LOCATION(SPK01) DAP(SEQ) SHR
       TO DATASET(RPT) LOCATION(SPK02)
       NEW SPACE(1,1) BLKSIZE(3120) LRECL(80) RECFM(FB)
       BLOCK(3120) UNIT(SYSDA) VOLUME(111111)
```

BUFL — Define Buffer Lengths

Use this parameter when defining a new "to" data set. Use this parameter to define the size of each buffer in the I/O buffer pool.

Rules

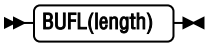
Optional or Required: Optional

Also Requires: The NEW and SPACE parameters, and either the BLOCK, CYLINDERS, or TRACKS parameter

Section: TO

DAP: SEQ, PDS

Format



length
is the length of the buffers in bytes. It is a decimal number from 1 through 32760.

Usage Notes

1. If you code this parameter in the FROM section, BDT will accept the parameter but will not use it.
2. For a list of parameters that can be used to allocate a new data set, see Appendix A, “Parameters That Require Other Parameters,” on page 83.

Example

Copy a data set to a new data set, specifying an I/O buffer length of 4096. Use the JES3 prefix (*S,BDT).

```
*S,BDT Q  FROM DATASET(BDATA.AA) DAP(PDS) LOCATION(NODEB)
          TO DATASET(CDATA.AA) LOCATION(NODEC)
          NEW LRECL(255) BLKSIZE(259) RECFM(V) DIR(12)
          SPACE(1,2) CYL DSORG(PO) VOLUME(111111)
          DISP(KEEP,DELETE) UNIT(3350)
          BUFL(4096)
```

CONTIG — Allocate Space in a Contiguous Area

Use this parameter when allocating a new “to” data set that is to be stored on a direct access storage device (DASD). This parameter requests that the primary space for the data set be allocated in a contiguous area.

Rules

Optional or Required: Optional

Also Requires: The NEW and SPACE parameters, and either the BLOCK, CYLINDERS, or TRACK parameter

Section: TO

DAP: SEQ, PDS

Format

Usage Notes

1. This parameter is invalid with the ALX or MXIG parameters.
2. If you code this parameter in the FROM section, BDT will accept the parameter but will not use it.
3. For a list of parameters that can be used to allocate a new data set, see Appendix A, “Parameters That Require Other Parameters,” on page 83.

Example

Copy the data set DATAB to the new direct access data set DATAC. Allocate the primary space for DATAC in a contiguous area. Use the TSO prefix (BDT).

```
BDT Q FROM DATASET(DATAB) LOCATION(MYNODE) DAP(SEQ)
      TO DATASET(DATAC) LOCATION(YOURNODE)
      NEW SPACE(1,1) UNIT(3380) VOLUME(111111)
      CONTIG BLOCK(80) BLKSIZE(800) LRECL(80) RECFM(F)
```

CSOPT — Compress Duplicate Strings of Blanks or Characters

Use this parameter to remove, or *compress*, duplicate strings of blanks or characters in the data being copied. The duplicate strings are removed before the data set is copied.

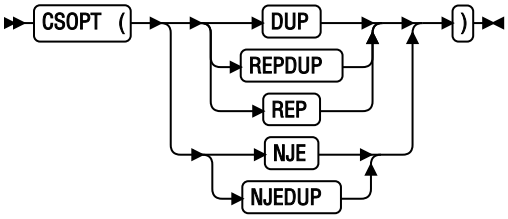
Rules

Optional or Required: Optional

Section: Job definition

DAP: SEQ, PDS

Format



DUP or REPDUP or REP
compresses 3 to 127 duplicate characters (including spaces).

NJE or NJEDUP
compresses 2 to 63 spaces or 3 to 63 consecutive duplicate characters.

Usage Notes

1. For this parameter to function, your installation must have defined the data compression option during initialization. Your system programmer can tell you which option (REPDUP or NJEDUP) is in effect. For more information see the description of the BDTNODE statement in *z/OS BDT Installation*.
2. Use DUP for data sets with a large amount of binary data or small amount of consecutive blanks.
3. Use NJE for data sets with a large amount of consecutive blanks, such as text, SYSOUT, or card-image data transfers.
4. Do not use CSOPT more than once in a transaction.

Example

Copy the data set MYDATA to the data set MOREDATA. Compress repeated repeated blanks in the copied data. Use the TSO prefix (BDT).

```
BDT Q  CSOPT(DUP) FROM DATASET(MYDATA) LOCATION(KGN01)
       DAP(SEQ)
       TO DATASET(MOREDATA) LOCATION(KGN02)
```

CYLINDERS — Allocate Space in Cylinder Units

Use this parameter when allocating a new “to” data set that is to be stored on a direct access storage device (DASD). This parameter allocatesspace for the “to” data set in cylinder units.

Rules

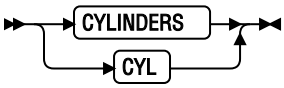
Optional or Required: Optional

Also Requires: The NEW and SPACE parameters

Section: TO

DAP: SEQ, PDS

Format



Usage Notes

1. This parameter is invalid with the BLOCK or TRACKS parameters.
2. For a list of parameters that can be used to allocate a new data set, see Appendix A, “Parameters That Require Other Parameters,” on page 83.

Example

Copy the data set JUNEDATA to a new data set, NEWDATA, and allocate space for NEWDATA in cylinder units. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(JUNEDATA) LOCATION(NODEA) DAP(SEQ)
       TO DATASET(NEWDATA) LOCATION(NODEB) NEW
       CYL SPACE(1,1) UNIT(SYSDA)
       BLKSIZE(800) LRECL(80)
       RECFM(F) VOLUME(111111)
```

DAP — Specify the DAP That Is to Copy the Data Set

Use this parameter to specify the dynamic application program (DAP) that is to copy a data set.

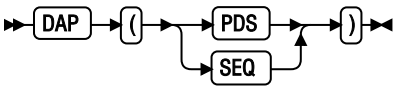
Rules

Optional or Required: Required for partitioned data sets (PDSs)

Section: FROM, TO

DAP: SEQ, PDS

Format



PDS
specifies that the PDS DAP is to copy the data set. Use the PDS DAP to copy an entire partitioned data set or selected members of a partitioned data set.

To copy selected members of a partitioned data set use DAP(PDS) with the SELECT or EXCLUDE options of the PARMS parameter.

SEQ
specifies that the SEQ DAP is to copy the data set. Use the SEQ DAP to copy a sequential data set or one member of a partitioned data set.

To copy one member of a PDS, use DAP(SEQ), and specify the member to be copied as part of the fully-qualified data set name in the DATASET parameter, or with the MEMBER parameter. If you do not specify the DAP parameter, SEQ is assumed.

Usage Notes

1. You only need to include the DAP parameter in the FROM or the TO section of your transaction. If you include the DAP parameter in both the FROM and TO sections, be sure that you specify the same value in both sections.
2. Do not use BDT to copy a PDS member whose directory entry contains user TTRs (relative track addresses) or note lists (pointers to blocks within a PDS member). BDT will not copy user TTRs or note lists. This means that you cannot copy programs that have been processed into an overlay structure by the linkage editor.
3. If you use DAP(SEQ), the record length of the "from" data set must be equal to or less than the maximum record length defined in the "to" data set's data control block (DCB). If not, BDT will truncate records in the "to" data set.
4. The following combinations of parameters are invalid. Do not code them in the same section of a transaction:
   - DAP(PDS) and DUMMY
   - DAP(PDS) and MEMBER

   Note: Do not use DAP(SEQ) and MEMBER(member) to copy a load module that is a member of a PDS. A load module copied with these parameters will not be executable.

Examples

1. Copy a the sequential data set ADATA to the sequential data set BDATA using the SEQ DAP. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(ADATA) LOCATION(ANODE) DAP(SEQ)
       TO DATASET(BDATA) LOCATION(BNODE)
```

2. Copy the PDS member MEMA of the PDS PDS1.AA to the PDS member MEMB of the PDS PDS2.AA. Use the SEQ DAP and the MEMBER parameter.

```
BDT Q  FROM DATASET(PDS1.AA) MEMBER(MEMA) LOCATION(THISNODE)
       DAP(SEQ) SHR
       TO DATASET(PDS2.AA) MEMBER(MEMB) LOCATION(OTHRNODE)
       OLD
```

3. Copy the PDS DATA.ONE to the PDS DATA.TWO using the PDS DAP.

```
BDT Q FROM DATASET(DATA.ONE) LOCATION(ONENODE) DAP(PDS) SHR
      TO DATASET(DATA.TWO) LOCATION(TWONODE) OLD
```

DATASET — Specify a Data Set Name

Use this parameter to specify the names of the "from" and "to" data sets.

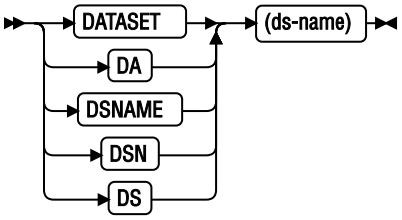
Rules

Optional or Required: Required in the FROM section. Required in the TO section unless you use the INTRDR or DUMMY parameters.

Section: FROM, TO

DAP: SEQ, PDS

Format



ds-name
is the fully-qualified data set name.

Usage Notes

1. Do not code this parameter together with the DUMMY or INTRDR parameters in the TO section. The DUMMY parameter can be used to specify a dummy data set as the “to” data set. The INTRDR parameter can be used to specify the MVS internal reader at the receiving node as the “to” data set.
2. If you omit the DATASET, DUMMY, and INTRDR parameters from the TO section, the system copies the “from” data set to a temporary system data set. After the transfer is completed, the system deletes the temporary data set regardless of any disposition you may have specified with the DISP parameter.
3. The information the system will use to resolve a generation data group (GDG) data set referenced by relative number is controlled by the BDT option GDGLOCS. For more information, see *z/OS BDT Installation*.
4. If a transaction is entered by a TSO user and the data set name is not enclosed in single quotation marks, the TSO prefix specified for the user is added as a prefix to the data set name.

Example

Copy the data set named PRTDATA to the data set named JANDATA. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(PRTDATA) LOCATION(MYNODE) DAP(SEQ)
       TO DATASET(JANDATA) LOCATION(JANNODE)
```

DCBDS — Read DCB Information from a Data Set Label

Use this parameter when allocating a new “to” data set to be stored on a direct access storage device (DASD). This parameter requests that MVS read the data control block (DCB) information for the “to” data set from the label of another data set.

Rules

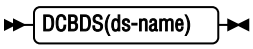
Optional or Required: Optional

Also Requires: The NEW and SPACE parameters, and either the BLOCK, CYLINDERS, or TRACKS parameters

Section: TO

DAP: SEQ, PDS

Format



ds-name
is the fully-qualified data set name of the data set from which you want to obtain DCB information.

The data set:

- Must be cataloged
- Must be on a direct access device
- Must be on the same system as the “to” data set.

Usage Notes

1. The DCB information obtained from the data set label is:
   - Block size
   - Data set organization
   - Logical record length
   - Record format.
2. If you code the BLKSIZE, DSORG, LRECL, or RECFM parameters in your transaction, the parameter value that you code will be used instead of the value found in the data set label of the data set specified with the DCBDS parameter.
3. If a transaction is entered by a TSO user and the data set name is not enclosed in single quotation marks, the TSO prefix specified for the user is added as a prefix to the data set name.
4. Do not code the DCBDS parameter in the FROM section. If you do and the data set’s DCB information is incompatible with the information contained in the DCBDS parameter, the transaction will fail.
5. For a list of parameters that can be used to allocate a new data set, see Appendix A, “Parameters That Require Other Parameters,” on page 83.

Example

Copy the data set FILEA to the new data set FILEB, and read DCB information for FILEB from FILEA. Use the JES3 prefix (*S,BDT).

```
*S,BDT Q   FROM DATASET(FILEA) LOCATION(NODEA) DAP(SEQ)
           TO DATASET(FILEB) LOCATION(NODEB) NEW SPACE
           DCBDS(FILEA) CYLINDERS UNIT(SYSDA)
           VOLUME(111111)
```

DEN — Define the Recording Density of a Tape Data Set

Use this parameter to define the recording density of a data set stored on magnetic tape.

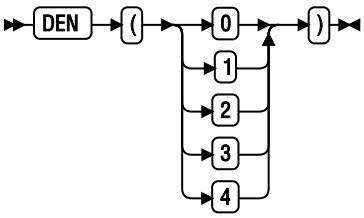
Rules

Optional or Required: Optional

Section: FROM, TO

DAP: SEQ

Format



0
is the density for 200 bpi 7-track tapes.

1
is the density for 556 bpi 7-track tapes.

2
is the density for 800 bpi 7-track or 800 bpi 9-track tapes.

3
is the density for 1600 bpi 9-track tapes.

4
is the density for 6250 bpi 9-track tapes.

If you do not specify this parameter, the highest applicable density is assumed.

Example

Copy the data set DEPTA to the data set NAMES, which is stored on a 1600 bpi 9-track magnetic tape. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(DEPTA) LOCATION(KGN01) DAP(SEQ)
       TO DATASET(NAMES) LOCATION(KGN02)
       NEW VOLSER(11111) UNIT(TAPE9)
       DEN(3) BLKSIZE(8000) LRECL(80) RECFM(F)
```

DIAGNS — Request the Trace Option

Use this parameter to request the OPEN/CLOSE/EOV trace option, which gives a module-by-module trace of OPEN/CLOSE/EOV's work area and the user's data control block (DCB).

Rules

Optional or Required: Optional

Section: TO

DAP: SEQ, PDS

Format

Usage Note

If you use the trace option, the generalized trace facility (GTF) must be active in the system on which your transaction executes. Check with the MVS operator on your system to find out if the GTF is active.

Example

Copy the data set ONEDATA to the data set TWODATA and request the trace option. Use the TSO prefix (BDT).

```
BDT Q FROM(ONEDATA) LOCATION(ONENODE) DAP(SEQ)
      TO(TWODATA) LOCATION(TWONODE) DIAGNS(TRACE)
```

DIR — Request Directory Blocks for a New PDS

Use this parameter when allocating a new "to" partitioned data set (PDS) to be stored on a direct access storage device (DASD). This parameter requests directory blocks for the new PDS.

Rules

Optional or Required: Required for a new PDS

Also Requires: The NEW and SPACE parameters, and either the BLOCK, CYLINDERS, or TRACKS parameters

Section: TO

DAP: PDS

Format

blocks
is the number of directory blocks, from 1 to 999999.

Example

Copy the data set OLDDATA to the new PDS NEWDATA and request 50 directory blocks for NEWDATA. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(OLDDATA) LOCATION(NODEA) DAP(PDS) SHR
       TO DATASET(NEWDATA) LOCATION(NODEB)
       NEW SPACE(1,1) CYL BLKSIZE(800) LRECL(80) RECFM(F)
       DIR(10) UNIT(SYSDA) VOLUME(111111)
```

DISP — Specify the Disposition of a Data Set

Use this parameter to tell MVS what to do with the “to” or “from” data set when the transaction completes or when the BDT address space terminates while the transaction is active.

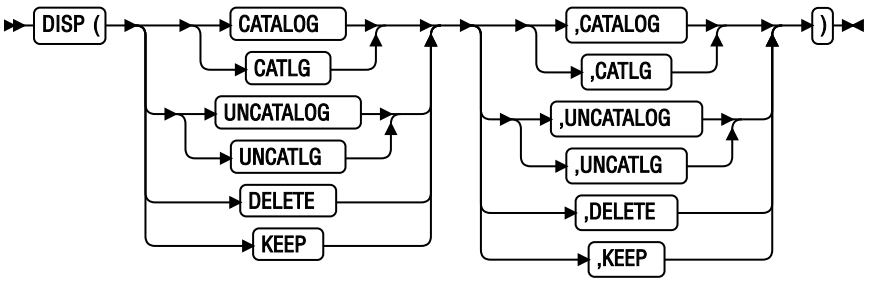
Rules

Optional or Required: Optional

Section: FROM, TO

DAP: SEQ, PDS

Format



If you specify one disposition, it tells MVS what to do with the data set after the data transfer completes normally.

If you specify two dispositions, the first disposition tells MVS what to do with the data set after the data transfer completes normally, and the second disposition tells MVS what to do with the data set if the transaction terminates abnormally.

CATALOG or CATLG
specifies that the data set is to be kept and that an entry pointing to the data set is to be added to the system or user catalog. CATALOG is the default for new “to” data sets.

UNCATALOG or UNCATLG
specifies that the data set is to be kept but the system or user catalog entry for the data set is to be removed.

DELETE
specifies that the data set is no longer needed; its space is to be released for use by other data sets.

KEEP
specifies that the data set is to be kept. KEEP is the default for existing data sets.

Usage Notes

1. If you use DISP(DELETE) together with the EXPDT or RETPD parameter, MVS ignores the EXPDT or RETPD parameter and deletes the data set.
2. For more information on cataloging data sets, see the *z/OS MVS JCL User's Guide* that is appropriate for your installation.

Examples

1. Copy a data set to an existing data set and specify that MVS is to keep the "to" data set after the transcation completes. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(DATAB) LOCATION(KGN01) DAP(PDS)
       TO DATASET(DATAC) LOCATION(KGN02) OLD DISP(KEEP)
```

2. Copy a data set to a new data set. If the transaction completes normally, MVS is to keep the "from" data set, but not catalog it, and also catalog the new "to" data set. If the transaction terminates abnormally, MVS is to delete the "from" and "to" data sets.

```
BDT Q  FROM DATASET(FILEA) LOCATION(NODEA) DAP(PDS)
       DISP(KEEP,DELETE)
       TO DATASET(FILEB) LOCATION(NODEB)
       NEW LRECL(80) BLKSIZE(800) RECFM(F) DIR(12)
       SPACE(1,2) CYL DSORG(PO) VOLUME(222222)
       DISP(CATLG,DELETE) UNIT(3380)
```

DSORG — Specify Data Set Organization

Use this parameter when allocating a new "to" data set that is to be stored on a direct access storage device (DASD). This parameter specifies the organization of the "to" data set.

Rules

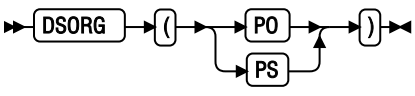
Optional or Required: Optional

Also Requires: The NEW and SPACE parameters, and either the BLOCK, CYLINDERS, or TRACKS parameter

DAP: SEQ, PDS

Section: TO

Format



PO
specifies that the data set organization is partitioned.

PS
specifies that the data set organization is physical sequential.

Usage Notes

1. Do not code the DSORG parameter in the FROM section. If you do so and the data set's data control block (DCB) information is incompatible with the information contained in the DSORG parameter, the transfer will fail.
2. For a list of parameters that can be used to allocate a new data set, see Appendix A, "Parameters That Require Other Parameters," on page 83.

Example

Copy the data set MAYDATA to the new data set JUNEDATA and specify that the organization of the JUNEDATA is partitioned. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(MAYDATA) LOCATION(KGN01) DAP(PDS)
       TO DATASET(JUNEDATA) LOCATION(KGN02)
       NEW DSORG(PO)
       LRECL(255) BLKSIZE(259) RECFM(V) DIR(12)
       SPACE(1,2) CYL VOLUME(111111)
       DISP(KEEP,DELETE) UNIT(3350)
       BUFL(4096)
```

DUMMY — Request a Dummy Data Set

Use this parameter to request a dummy data set instead of a real data set. Using a dummy data set allows you to test a transaction without actually copying data.

Rules

Optional or Required: Optional

Also Requires: The LRECL and RECFM parameters, and either the DCBDS or BLKSIZE parameter

Section: FROM, TO

DAP: SEQ

Format

Usage Notes

1. The DUMMY parameter is invalid if used with the INTRDR or DATASET parameters in the TO section.
2. For a dummy data set, MVS bypasses device and space allocation, I/O operations, and data set disposition.
3. For more information on the use of dummy data sets, see the JCL manual that is appropriate for your installation.

Example

Test a transaction using the data set TESTDSN as the "from" data set and a dummy data set as the "to" data set. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(TESTDSN) LOCATION(KGN01) DAP(SEQ)
       TO DUMMY RECFM(FB) LRECL(80) BLKSIZE(800)
       LOCATION(KGN03)
```

EXPDT — Specify an Expiration Date for a Data Set

Use this parameter to assign an expiration date to a data set. MVS will not allow the data set to be deleted or written over before this date.

Rules

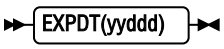
Optional or Required: Optional

Also Requires: The LABEL parameter if the data set is new

Section: TO

DAP: SEQ, PDS

Format



yyddd
is the expiration date you want to assign to the data set. *yy* is the year and may be any number from 00 to 99. *ddd* is the day and may be any number from 001 to 366.

Usage Notes

1. If you want to protect a data set from being overwritten or deleted for a certain number of days, you can use the RETPD parameter.
2. If you code the EXPDT parameter in the FROM section, BDT will accept the parameter but will not use it.
3. For more information on the use of the EXPDT parameter, see the LABEL parameter in the JCL manual that is appropriate for your installation.

Example

Copy the data set MYDSN to the data set THATDSN and assign an expiration date of June 13, 1987 to THATDSN. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(MYDSN) LOCATION(MYNODE) DAP(SEQ)
       TO DATASET(THATDSN) LOCATION(THATNODE) EXPDT(87164)
```

FROM — Begin the FROM Section

Use this parameter to mark the beginning of the FROM section of the transaction. The parameters following FROM describe the data set to be copied and the node at which the data set is located.

Rules

Optional or Required: Optional, unless you place FROM parameters after the TO parameter

Section: FROM

DAP: SEQ, PDS

Format

FROM
marks the beginning of the FROM section of the transaction.

Usage Notes

1. You can split the FROM section of your transaction into several parts, each beginning with a FROM parameter.
2. See Chapter 1, “Writing and Submitting File-to-File Transactions,” on page 1 for general information on the FROM section of a transaction.

Examples

1. Copy the data set DDATA to the data set GDATA. List all the parameters describing the “from” data set before the TO section of the transaction and omit the FROM parameter. Use the TSO prefix (BDT).

```
BDT Q  DATASET(DDATA) LOCATION(KGN01) DAP(SEQ)
       TO DATASET(GDATA) LOCATION(KGN02)
```

2. Copy the data set BDATA to the data set HDATA. Specify the location of the “from” data set after the TO section of the transaction.

```
BDT Q  FROM DATASET(BDATA) DAP(SEQ)
       TO DATASET(HDATA) LOCATION(SPK01)
       FROM LOCATION(SPK02)
```

GMJDLIB — Request a GMJD Library

Use this parameter to request that BDT combine your transaction definition with a transaction definition that is stored in a private GMJD library.

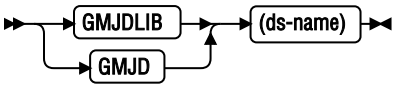
Rules

Optional or Required: Optional

Section: Job definition

DAP: SEQ, PDS

Format



ds-name
is the name of the data set that contains the private GMJD library.

Usage Notes

1. The data set that contains the private GMJD library must be located at the node where you submit your transaction, must be accessible to BDT, and must be cataloged.
2. Do not use this parameter with a transaction code of Q.
3. For general information about submitting transactions from GMJD libraries, see Chapter 2, “Storing and Reusing Transaction Definitions — GMJD Libraries,” on page 9.

Examples

1. Submit the transaction definition stored as member ENDREPT in your private GMJD library, which is a partitioned data set named JOESLIB. Use the TSO prefix (BDT).

```
BDT ENDREPT GMJDLIB(JOESLIB)
```

2. Submit the transaction definition stored in the private GMJD library that is a sequential data set named REPORT. Use the TSO prefix (BDT).

```
BDT AUG GMJDLIB(REPORT)
```

HOLD — Put a Transaction into Operator Hold

Use this parameter to put a transaction into operator hold on the BDT work queue.This prevents the transaction from running until it is released from hold.

Rules

Optional or Required: Optional

DAP: SEQ, PDS

Section: Job definition, FROM, TO

Format

Usage Note

To release the transaction from hold, use the F,J,R command. See *z/OS BDT Commands* for more information.

Example

Request that a transaction be put into operator hold. Use the JES3 prefix (*S,BDT).

```
*S,BDT Q JOBNAME(MYJOB) HOLD
         FROM DATASET(ADATA) LOCATION(NODEA) DAP(SEQ)
         TO DATASET(BDATA) LOCATION(NODEB)
```

INTRDR — Copy a Data Set to the MVS Internal Reader

Use this parameter to copy a data set to the MVS internal reader at the receiving node.

Rules

Optional or Required: Optional

Also Requires: The BLKSIZE parameter

Section: TO

DAP: SEQ

Format

Usage Notes

1. This parameter is invalid if used with either the DATASET or DUMMY parameters in the TO section.
2. INTRDR also allows a user's BDT transaction stream to submit an MVS job at a remote node. Conversely, the BDT batch program allows a user's MVS job stream to submit BDT transactions at a

remote node. See Chapter 1, “Writing and Submitting File-to-File Transactions,” on page 1 for more information.

Example

Copy the data set COPYDATA to the MVS internal reader at node KGN01. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(COPYDATA) LOCATION(NODEA) DAP(SEQ)
       TO INTRDR LOCATION(KGN01) BLKSIZE(1330)
```

JOBNAME — Assign a Name to the Job That Results from a Transaction

Use this parameter to assign a name to the job that results from your transaction.

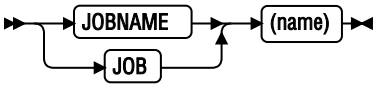
Rules

Optional or Required: Required with the NETREL parameter

Section: Job definition, FROM, TO

DAP: SEQ, PDS

Format



name
is the name you want to be assigned to the job. The name may be 1 to 8 alphanumeric characters.

If you do not specify this parameter, your job will be given one of the following names:

- The GMJD member name if you have specified a GMJD library
- “AQJOB” if you have specified a Q-type transaction (if you have used a transaction code of Q).

Usage Notes

1. Use JOBNAME if you merge several transactions with the same GMJD library member. The name (and the job number) will help you distinguish one transaction from another.
2. Use JOBNAME when creating dependent transaction control networks, to assign names to predecessors and successors in the network.

Examples

1. Submit a transaction to copy a data set and name the resulting job MYJOB. Use the TSO prefix (BDT).

   ```
   BDT Q JOBNAME(MYJOB)
         FROM DATASET(ONEDATA) LOCATION(ONENODE) DAP(SEQ)
         TO DATASET(TWODATA) LOCATION(TWONODE)
   ```

2. Submit a transaction using a GMJD library and name the resulting job THISJOB. The GMJD library is a partitioned data set named MYLIB. The member of the library that contains the stored transaction definition is MYMEMBER. Use the TSO prefix (BDT).

   ```
   BDT MYMEMBER GMJD(MYLIB) JOBNAME(THISJOB)
   ```

LABEL — Supply Label Information for a Data Set

Use this parameter to supply label information for a data set.

Rules

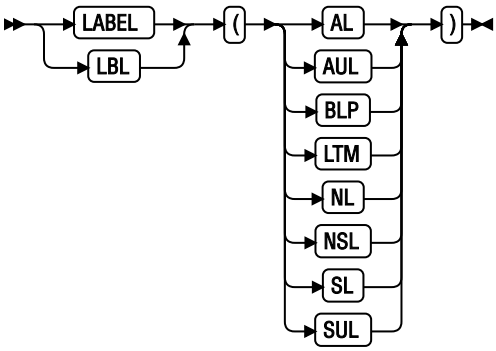
Optional or Required: Required for:

- An existing data set, if the data set has a label other than a standard IBM label
- A new data set, if:
  - You want to assign a label other than a standard IBM label to the data set.
  - You use the PASSWORD parameter with the data set.
  - You use the EXPDT parameter or the RETPD parameter with the data set.

Section: FROM, TO

DAP: SEQ, PDS

Format



AL
specifies the American National Standard label.

AUL
specifies American National Standard and user labels.

BLP
specifies that label processing is to be bypassed.

LTM
specifies a leading tape mark.

NL
specifies no labels.

NSL
specifies nonstandard labels.

SL
specifies the IBM standard label.

SUL
specifies both IBM standard and user labels.

If you do not specify this parameter:

- A label is obtained from the catalog for a cataloged data set.
- An IBM standard label is given to a new data set.

Usage Note

For more information on data set labels, see the *z/OS MVS JCL User's Guide* that is appropriate for your installation.

Example

Copy the data set JANDATA, which is stored on tape, to a new tape data set, RPTDATA. Specify that no label information is to be supplied for either data set. Use the TSO prefix (BDT).

```
BDT Q FROM DATASET(JANDATA) LOCATION(KGN01) DAP(SEQ)
      LABEL(NL)
      UNIT(TAPE) VOL(1025) RECFM(FB) LRECL(80) BLKSIZE(3120)
      TO DATASET(RPTDATA) LOCATION(KGN02)
      LABEL(NL)
      NEW EXPDT(82163)
      UNIT(TAPE) RECFM(FB) LRECL(80) BLKSIZE(3120)
      VOLUME(222222)
```

LOCATION — Specify the Location of a Data Set

Use this parameter to specify the locations (nodes) of the “from” or “to” data sets.

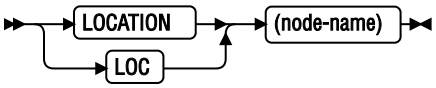
Rules

Optional or Required: Required in a single-BDT complex unless the data set named in the DATASET parameter is at the node where the transaction is submitted. Required in a poly-BDT complex unless the data set named in the DATASET parameter is at the node that is defined as the default for your installation.

Section: FROM, TO

DAP: SEQ, PDS

Format



node-name
is the name of the node at which the data set is located.

If you do not specify this parameter, the following location is assumed:

- In a single-BDT complex, the node at which you submit the transaction. Your installation may override this default.
- In a poly-BDT complex, the node that is defined as the default node for your installation.

Contact your BDT system programmer for the default node and for the names of the nodes in your network.

Example

Copy a data set located at your node to a data set located at the node named OTHERNODE. You do not need to use the LOCATION parameter in the FROM section of the transaction because the “from” data set is located at the node where you submit the transaction. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(MY.DATA.SET) DAP(SEQ)
       TO DATASET(OTHER.DATA.SET) LOCATION(OTHERNODE)
```

LRECL — Specify the Logical Record Length of a Data Set

Use this parameter when allocating a new “to” data set. This parameter tells MVS the logical record length of the data set.

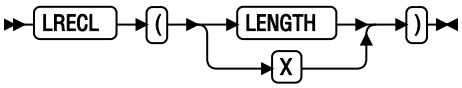
Rules

Optional or Required: Required for new “to” data sets, and for existing tape data sets that are not labeled

Section: FROM, TO

DAP: SEQ, PDS

Format



length
is the logical record length of the data set in bytes. It is a number from 0 through 32760.

X
specifies that the data set has a variable-length or spanned record type, with a logical record length exceeding 32756 bytes.

The default for existing data sets is the logical record length in the data set label. For a data set without a label, there is no IBM-defined default.

Example

Copy the data set CODATA to a new data set named NEWDATA. Specify a logical record length of 80 bytes for NEWDATA. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(CODATA) LOCATION(CONODE) DAP(SEQ)
       TO DATASET(NEWDATA) LOCATION(NNODE)
       NEW SPACE(2,1) CYL VOLUME(KGN01) DSORG(PS)
       LRECL(80) BLKSIZE(3120) RECFM(FB) DIR(2)
       UNIT(SYSDA)
```

MAXVOL — Specify the Maximum Number of Volumes Required for a Data Set

Use this parameter when allocating a new “to” data set. This parameter specifies the maximum number of volumes required for the data set.

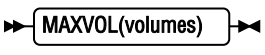
Rules

Optional or Required: Required if the data set needs more than five volumes and you do not use the VOLUME parameter

Section: TO

DAP: SEQ

Format



volumes
is the maximum number of volumes required by the data set. *volumes* is a number from 1 through 255.

If you do not specify this parameter, 5 volumes is assumed, unless you use the VOLUME parameter.

Usage Note

If you use the both the MAXVOL and VOLUME parameters, the VOLUME parameter will override the MAXVOL parameter.

Example

Copy the data set DATA to the new tape data set NEWDATA and specify that NEWDATA will require a maximum of 7 volumes. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(DATA) LOCATION(NODEA) DAP(SEQ)
       TO DATASET(NEWDATA) LOCATION(NODEB) NEW
       MAXVOL(7) LRECL(133) BLKSIZE(133) RECFM(F)
       UNIT(TAPE) (KGN02)
```

MEMBER — Specify a Member of a Partitioned Data Set

Use this parameter to specify a member of a partitioned data set (PDS).

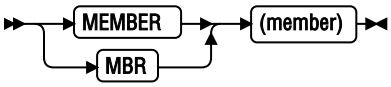
Rules

Optional or Required: Optional

Section: FROM, TO

DAP: SEQ

Format



member
is the name of the PDS member you want to copy or to receive the copied data set.

Usage Notes

1. If you use this parameter to copy data sets with different logical record lengths and do not use the PAD parameter, the “to” data set will be padded with binary zeroes.
2. The PARMS parameter offers more options for copying members of partitioned data sets.

Examples

1. Copy the member FIRST from the partitioned data set MY.DATA to the member ONE of the partitioned data set THAT.DATA. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(MY.DATA) LOCATION(MYNODE) DAP(SEQ)
       MEMBER(FIRST) OLD
       TO DATASET(THAT.DATA) LOCATION(THATNODE)
       MEMBER(ONE)
```

2. Copy the member MEMONE from the PDS MY.DATA to the sequential data set TARGET.

```
BDT Q  FROM DATASET(MY.DATA) LOCATION(MYNODE) DAP(SEQ)
       MEMBER(MEMONE)
       TO DATASET(TARGET) LOCATION(OTHERNODE)
```

3. Copy the sequential data set SOURCE to the member SRCMEM of the PDS MANY.SOURCES.

```
BDT Q  FROM DATASET(SOURCE) LOCATION(MYNODE) DAP(SEQ)
       TO DATASET(MANY.SOURCES) LOCATION(OTHERNODE)
       MEMBER(SRCMEM)
```

MOD — Copy a Data Set to the End of an Existing Data Set

Use the MOD parameter to:

- Add a copied data set to the end of an existing sequential data set.
- Add a copied data set to an existing partitioned data set (PDS) as a new member.

MOD gives your BDT subsystem exclusive use of the TO data set.

Rules

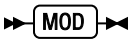
Optional or Required: Optional

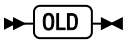
Section: TO

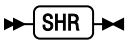
DAP: SEQ, PDS

Format

MOD is one of four data set status parameters that can be specified in the TO section of a transaction. All four are shown and explained subsequently. Use only one in the TO section of a transaction.







Used in the TO section of a transaction,

MOD
specifies that the “to” data set is an existing data set, and that the copied data set is to be added to it. This parameter gives your BDT subsystem exclusive use of the “to” data set.

NEW
specifies that the “to” data set is a new data set to be allocated in the transaction. It gives your BDT subsystem exclusive use of the data set.

OLD
specifies that the “to” data set is an existing data set, and that the copied data set is to overwrite it. This parameter gives your BDT subsystem exclusive use of the data set.

This is the default for the TO section.

SHR
specifies to MVS that the “to” data set is an existing data set and allows other users to share the use of it with your BDT subsystem.

Usage Notes

1. You may want to use the BDTENQ parameter to indicate to BDT whether a data set can be shared by other BDT transactions. The default value for the BDTENQ parameter is exclusive use of the “to” data set. See the BDTENQ parameter for more information.
2. Use only one of the four data set status parameters in the TO section of a transaction.
3. If you code the MOD parameter in the FROM section, BDT will accept the parameter but will not use it.
4. If you use MOD to add the copied data as a new PDS member, you must also use the MEMBER parameter to name the member. The name you give to the PDS member must not already exist in the PDS directory.

Example

Add the data set DATAB to the end of the data set DATAC. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(DATAB) LOCATION(SPK01) DAP(SEQ)
       TO DATASET(DATAC) LOCATION(SPK01) MOD
```

MSGCLASS — Specify the Destination of Messages

Use this parameter to indicate where BDT is to send messages that pertain to this transaction.

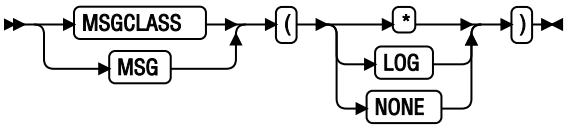
Rules

Optional or Required: Optional

Section: Job definition, FROM, TO

DAP: SEQ, PDS

Format



*****
specifies that messages are to be sent to the terminal where you submitted the transaction and to the BDT system log.

LOG
specifies that messages are to be sent to a log defined by the installation. The messages may be processed by an installation-written exit routine.

NONE
specifies that messages are to be sent to the BDT system log.

Usage Notes

1. When using the * option to send messages to a terminal, you can reduce the number of messages sent to the terminal by specifying MSG=S on the PARMS parameter.
2. Do not confuse this parameter (which stands alone) with the MSG option of the PARMS parameter.

Example

Copy a data set to another data set and specify that messages related to the transaction be sent to the terminal at which you submit the transaction.

```
BDT Q MSGCLASS(*)
      FROM LOC(SYSA1) DATASET(ADATA.SET) DAP(SEQ)
      TO   LOC(SYSA2) DATASET(BDATA.SET)
      NEW LRECL(80) BLKSIZE(3120) RECFM(FB)
      SPACE(12,1) CYLINDERS VOLUME(BDTDS8)
      DISP(CATLG,DELETE) DSORG(PS)
      UNIT(3380)
```

MSVGP — Specify the Mass Storage Volumes on Which a Data Set Is Located

Use this parameter to indicate that a data set is located on a specific group of mass storage volumes on a mass storage system (MSS) device.

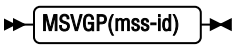
Rules

Optional or Required: Required for MSS data sets

Section: TO

DAP: SEQ, PDS

Format



mss-id
is the mass storage volume group identifier (1 to 8 characters).

Usage Note

This parameter is invalid with the ALX, MXIG, and VOLUME parameters.

Example

Copy a data set to the data set SDATA, which is stored on the group of mass storage volumes named MYGROUP. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(RDATA) LOCATION(KGN01) DAP(SEQ)
       TO DATASET(SDATA) MSVGP(MYGROUP) LOCATION(KGN02)
```

MXIG — Request the Maximum Contiguous Space for a Data Set

Use this parameter when allocating a new data set that is to be stored on a direct access storage device (DASD). This parameter requests the largest area of contiguous space on a volume.

Rules

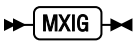
Optional or Required: Optional

Also Requires: The NEW and SPACE parameters, and either the BLOCK, CYLINDERS, or TRACKS parameters

Section: TO

DAP: SEQ, PDS

Format



Usage Notes

1. The size of the space allocated with this parameter must be equal to or larger than the amount of primary space requested through the SPACE parameter.
2. This parameter is invalid with the ALX or CONTIG parameters.
3. If you code the MXIG parameter in the FROM section, BDT will accept the parameter but will not use it.
4. For a list of parameters that can be used to allocate a new data set, see Appendix A, “Parameters That Require Other Parameters,” on page 83.

Example

Copy the data set OLDDATA to the new data set NEWDATA, and request the largest area of contiguous space on the volume for NEWDATA. Use the TSO prefix (BDT).

```
BDT Q  FROM LOC(SYSA1) DATASET(OLDATA) DAP(SEQ) SHR
       TO  LOC(SYSA2) DATASET(NEWDATA)
       NEW LRECL(80) BLKSIZE(3120) RECFM(FB)
       MXIG SPACE(12,1) CYLINDERS VOLUME(NEWDS8)
       DSORG(PS) UNIT(SYSDA)
```

NETCOND — Tell BDT What to Do with a Transaction in a DTC Network If a Predecessor Fails

Use this parameter with transactions that belong to a dependent transaction control (DTC) network. This parameter tells BDT what to do with this transaction after any of its predecessors in the DTC network completes normally or fails.

Rules

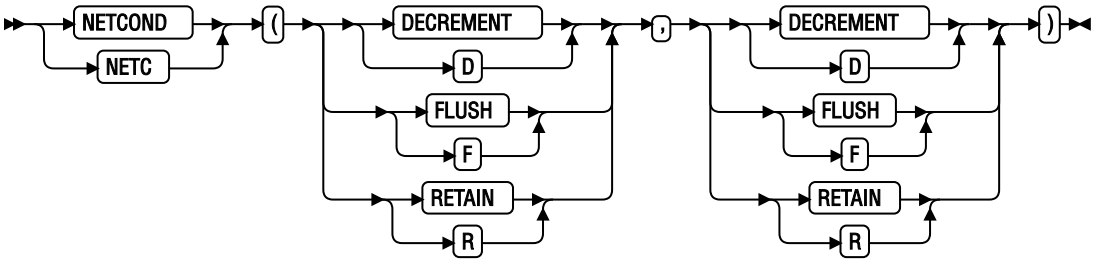
Optional or Required: Optional

Also Requires: The NETID parameter

Section: Job definition, FROM, TO

DAP: SEQ, PDS

Format



The first option (the one immediately after NETCOND) tells BDT what to do after a predecessor transaction completes normally.

The second option tells BDT what to do after a predecessor transaction completes abnormally (fails).

D or DECREMENT
decreases this transaction's hold count by 1 when any predecessor completes. This is the default.

F or FLUSH
flushes this transaction and its successors when any predecessor completes.

R or RETAIN
retains (does not change) the hold count of this transaction when any predecessor completes.

Usage Note

For basic information about dependent transaction control networks, see Chapter 3, "Controlling the Order in Which Transactions Are Processed — DTC Networks," on page 13.

Example

Specify that MYJOB3, in the DTC network MYNET, should have its hold count lowered when any of its predecessor transactions completes normally, but should be flushed if any of its predecessor transactions fails. Use the TSO prefix (BDT).

```
BDT Q JOBNAME(MYJOB3) NETID(MYNET) NETHOLD(1)
      NETREL(MYJOB4) NETCOND(D,F)
      FROM DATASET(MYDATA) LOCATION(KGN01) DAP(SEQ)
      TO DATASET(YOURDATA) LOCATION(KGN02)
```

NETHOLD — Specify the Hold Count of a Transaction in a DTC Network

Use this parameter to specify the hold count of a transaction in a dependent transaction control (DTC) network.

Rules

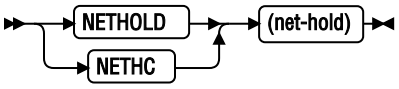
Optional or Required: Optional

Also Requires: The NETID parameter

Section: Job definition, FROM, TO

DAP: SEQ, PDS

Format



net-hold
is the hold count of the transaction. *net-hold* is a number from 1 to 32767.

Usage Notes

1. In most cases, the hold count you give a transaction should agree with the number of predecessor transactions specified with the NETREL parameter. However, you can give a transaction a hold count that is higher than the number of predecessor transactions. This will hold the transaction until you release it with the F,NET,ID,J,D command or the F,NET,ID,J,R command. See *z/OS BDT Commands* for more information.
2. You can give a transaction a hold count that is lower than the number of predecessor transactions. You might want to do this when you want a transaction to run after some, rather than all, of its predecessor transactions have completed.
3. You can use operator hold to hold the processing of transactions in a DTC network. Use either the F,J,C or the F,J,H command to put the first transaction in a DTC network into operator hold, and then the F,J,R command to release the transaction from hold. See *z/OS BDT Commands* for more information.
4. For basic information about dependent transaction control networks, see Chapter 3, “Controlling the Order in Which Transactions Are Processed — DTC Networks,” on page 13.

Examples

1. Create a DTC network called MYNET, in which MYJOB1 releases MYJOB2 which releases MYJOB3. Note that MYJOB2 and MYJOB3 each have one predecessor and a NETHOLD count of 1.

Give MYJOB1 a hold count of 1 with the NETHOLD parameter to keep it from running before the rest of the network is submitted. Use the TSO prefix (BDT).

```
BDT Q JOBNAME(MYJOB1) NETID(MYNET) NETREL(MYJOB2) NETHOLD(1)
      FROM DATASET(AADATA) LOCATION(BANODE) DAP(SEQ)
      TO DATASET(DADATA) LOCATION(DANODE) MOD
```

```
BDT Q JOBNAME(MYJOB2) NETID(MYNET) NETHOLD(1) NETREL(MYJOB3)
      FROM DATASET(BADATA) LOCATION(BANODE) DAP(SEQ)
      TO DATASET(DADATA) LOCATION(DANODE) MOD

BDT Q JOBNAME(MYJOB3) NETID(MYNET) NETHOLD(1)
      FROM DATASET(CADATA) LOCATION(BANODE) DAP(SEQ)
      TO DATASET(DADATA) LOCATION(DANODE) MOD
```

2. Create a DTC network called ANET. In ANET, AJOB1 releases AJOB2 **and** AJOB3. AJOB2 also releases AJOB3. As AJOB3 has two predecessors, give AJOB3 a hold count of 2.

   Give AJOB1 a hold count of 1 (with the NETHOLD parameter) to keep it from running before the rest of the network is submitted.

```
BDT Q JOBNAME(AJOB1) NETID(ANET) NETREL(AJOB2,AJOB3) NETHOLD(1)
      FROM DATASET(ADATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD

BDT Q JOBNAME(AJOB2) NETID(ANET) NETHOLD(1) NETREL(AJOB3)
      FROM DATASET(CDATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD

BDT Q JOBNAME(AJOB3) NETID(ANET) NETHOLD(2)
      FROM DATASET(DDATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD
```

3. Create a DTC network called BNET, in which BJOB1 and BJOB2 are predecessors to BJOB3. Specify that BJOB3 is to remain in DTC hold until it is released by a command. To do this, give BJOB3 a NETHOLD count of 3 although it has only two predecessors.

   Submit BJOB1 and BJOB2 in operator hold (using the HOLD parameter) to keep them from running before BJOB3 is submitted.

```
BDT Q JOBNAME(BJOB1) NETID(BNET) NETREL(BJOB3)  HOLD
      FROM DATASET(ADATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD

BDT Q JOBNAME(BJOB2) NETID(BNET) NETREL(BJOB3) HOLD
      FROM DATASET(CDATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD

BDT Q JOBNAME(BJOB3) NETID(BNET) NETHOLD(3)
      FROM DATASET(DDATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD
```

4. Create a DTC network called CNET, in which CJOB3 is released by CJOB1 and CJOB2. Allow CJOB3 to run when either CJOB1 *or* CJOB2 completes. To do this, give CJOB3 a hold count of 1 even though it has two predecessors.

```
BDT Q JOBNAME(CJOB1) NETID(CNET) NETREL(CJOB3)
      FROM DATASET(ADATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD

BDT Q JOBNAME(CJOB2) NETID(CNET) NETREL(CJOB3)
      FROM DATASET(CDATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD

BDT Q JOBNAME(CJOB3) NETID(CNET) NETHOLD(1)
      FROM DATASET(DDATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD
```

NETID — Assign a Transaction to a DTC Network

Use this parameter to assign a transaction to a specific dependent transaction control (DTC) network.

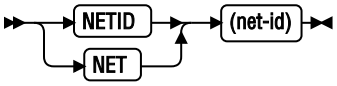
Rules

Optional or Required: Optional

Section: Job definition, FROM, TO

DAP: SEQ, PDS

Format



net-id
is the name of the DTC network to which BDT is to assign the transaction. The *net-id* must be 1 to 8 alphanumeric characters. The first character must be alphabetic.

Usage Notes

1. All transactions that are to be in the same DTC network must use the same *net-id*.
2. For basic information about dependent transaction control (DTC) networks, see Chapter 3, "Controlling the Order in Which Transactions Are Processed — DTC Networks," on page 13.

Example

Create a DTC network called MYNET, in which MYJOB1 releases MYJOB2 which releases MYJOB3. Use the TSO prefix (BDT).

```
BDT Q JOBNAME(MYJOB1) NETID(MYNET) NETREL(MYJOB2)
      FROM DATASET(ADATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD

BDT Q JOBNAME(MYJOB2) NETID(MYNET) NETHOLD(1) NETREL(MYJOB3)
      FROM DATASET(CDATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD

BDT Q JOBNAME(MYJOB3) NETID(MYNET) NETHOLD(1)
      FROM DATASET(DDATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD
```

NETREL — Specify Successor Transactions in a DTC Network

Use this parameter to tell BDT what transactions are successors to a transaction in a dependent transaction control (DTC) network.

Rules

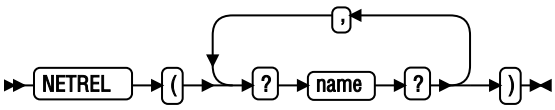
Optional or Required: Optional

Also Requires: The NETID, JOBNAME, and NETHOLD parameters

Section: Job definition, FROM, TO

DAP: SEQ, PDS

Format



name
specifies the name or names of the transactions that are successors to this transaction. Names are assigned to transactions with the JOBNAME parameter.

Usage Notes

1. You must use the NETHOLD parameter on transactions that are released by (are successors to) this transaction; otherwise they will not wait to be released.
2. This transaction and its successor transactions must all be part of the same DTC network, that is, they must all have the same *net-id*, specified with the NETID parameter.
3. The NETREL parameter can only release a successor transaction that is scheduled for processing at the same node as the predecessor transaction. See Chapter 3, “Controlling the Order in Which Transactions Are Processed — DTC Networks,” on page 13 for more information.

Example

Create a DTC network called XNET in which XJOB1 releases XJOB2 and XJOB3, and XJOB2 also releases XJOB3. Give XJOB3 a hold count of 2 (with the NETHOLD parameter) so that it will not run until both XJOB1 and XJOB2 complete. As XJOB3 does not have any successors, it does not need a NETREL parameter.

Submit XJOB1 in operator hold (with the HOLD parameter) to keep it from running before the rest of the network is submitted. Use the TSO prefix (BDT).

```
BDT Q JOBNAME(XJOB1) NETID(XNET) NETREL(XJOB2,XJOB3) HOLD
      FROM DATASET(ADATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD

BDT Q JOBNAME(XJOB2) NETID(XNET) NETHOLD(1) NETREL(XJOB3)
      FROM DATASET(CDATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD

BDT Q JOBNAME(XJOB3) NETID(XNET) NETHOLD(2)
      FROM DATASET(DDATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(BNODE) MOD
```

NEW — Specify That the “To” Data Set Is New

Use this parameter to indicate to MVS that the “to” data set is a new data set to be allocated in the transaction.

Your BDT subsystem will have exclusive use of the data set while the transaction is being processed.

Rules

Optional or Required: Required for new “to” data sets

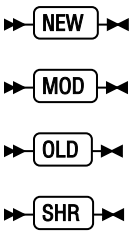
Also Requires: The BLKSIZE, LRECL, RECFM, UNIT, and VOLUME parameters. If the data set is a DASD data set, NEW also requires the SPACE parameter, and either the BLOCK, CYLINDERS, or TRACKS parameter. If the data set is a PDS, NEW also requires the DIR parameter.

Section: TO

DAP: SEQ, PDS

Format

NEW is one of four data set status parameters that can be specified in the TO section of a transaction. All four are shown and explained subsequently. Use only one in the TO section of a transaction.



Used in the TO section of a transaction,

NEW
specifies that the “to” data set is a new data set to be allocated in the transaction. This parameter gives your BDT subsystem exclusive us of the “to” data set.

MOD
specifies that the “to” data set is an existing data set, and that the copied data set is to be added to it. This parameter gives your BDT subsystem exclusive us of the “to” data set.

OLD
specifies that the “to” data set is an existing data set, and that the copied data set is to overwrite it. This parameter gives your BDT subsystem exclusive use of the data set.

This is the default for the TO section.

SHR
specifies that the “to” data set is an existing data set and that other users may share the use of it with your BDT subsystem.

Usage Notes

1. When allocating a new sequential data set, you may be able to accept defaults and omit the SPACE parameter, and the BLOCK, CYLINDERS, or TRACKS parameter. Defaults are provided as part of the MVS product; they may have been modified at your installation.
2. You may want to use the BDTENQ parameter to indicate to BDT whether a data set can be shared by other BDT transactions. The default value for the BDTENQ parameter is exclusive use of the “to” data set. See the BDTENQ parameter for more information.
3. Use only one of the four data set status parameters in the TO section of a transaction.
4. If you do not specify a data set disposition, using the DISP parameter, the system catalogs and keeps the new “to” data set after the transaction completes.
5. For a list of parameters that can be used to allocate new data sets, see Appendix A, “Parameters That Require Other Parameters,” on page 83.

Examples

1. Copy the existing data set, BOBSDATA, to a new data set, JOESDATA. Use the TSO prefix (BDT).

```
BDT Q FROM DATASET(BOBSDATA) OLD LOCATION(BOBSNODE) DAP(SEQ)
      TO DATASET(JOESDATA) LOCATION(JOESNODE)
      NEW SPACE(1,1) TRACKS UNIT(SYSDA)
      BLKSIZE(800) LRECL(80) RECFM(FB)
      VOLUME(111111)
```

2. Copy a DASD data set named DATA.ONE to a new tape data set named DATA.TWO.

```
BDT Q FROM DATASET(DATA.ONE) UNIT(3380) VOLUME(11111)
      LOCATION(NODEONE)  DAP(SEQ)
      TO DATASET(DATA.TWO) NEW
      RECFM(F) LRECL(80) BLKSIZE(800)
```

```
UNIT(TAPE) LABEL(SL)
VOLUME(22222)
```

OLD — Specify Exclusive Use of an Existing Data Set

Use this parameter to indicate to MVS that a data set already exists, and that you want your BDT subsystem to have exclusive use of the data set while the transaction is being processed.

Rules

Optional or Required: Optional

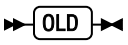
Section: FROM, TO

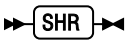
DAP: SEQ, PDS

Format

OLD is one of two data set status parameters that can be specified in the FROM section of a transaction. In addition, OLD is one of four data set status parameters that can be specified in the TO section of a transaction. The options for the FROM and TO sections are explained separately subsequently.

FROM Options:



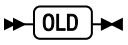


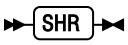
Use only one of the options in the FROM section of a transaction.

OLD
specifies that the “from” data set is an existing data set, and that your BDT subsystem has exclusive use of the data set while the transaction is being processed.

SHR
specifies that the “from” data set is an existing data set and that other users may share the use of it with your BDT subsystem. This is the default for the “from” data set.

TO Options:





Use only one of the options in the TO section of a transaction.

OLD
specifies that the “to” data set is an existing data set, and that the copied data set is to overwrite it. This parameter gives your BDT subsystem exclusive use of the data set while the transaction is being processed.

This is the default for the TO section.

SHR
specifies that the “to” data set is an existing data set and that other users may share the use of it with your BDT subsystem.

MOD
specifies that the “to” data set is an existing data set and that the copied data set is to be added to it. This parameter gives your BDT subsystem exclusive us of the “to” data set.

NEW
specifies that the “to” data set is a new data set to be allocated in the transaction. This parameter gives your BDT subsystem exclusive us of the “to” data set.

Usage Note

You may want to use the BDTENQ parameter to indicate to BDT whether a data set can be shared by other BDT transactions. The default value for the BDTENQ parameter is exclusive use of the “to” data set. See the BDTENQ parameter for more information.

Example

Copy an existing data set, IDATA, to another existing data set, JDATA, and specify exclusive use of both data sets while the transaction is being processed. Use the JES3 prefix (*S,BDT).

```
*S,BDT Q  FROM DATASET(IDATA) LOCATION(INODE) DAP(SEQ) OLD
          TO DATASET(JDATA) LOCATION(JNODE) OLD
```

PARALLEL — Specify Parallel Mounting of Volumes

Use this parameter to indicate that an existing data set occupies more than one volume and that each volume is to be mounted on a separate device.

Rules

Optional or Required: Optional

Also Requires: The VOLUME parameter, with a volume serial for each volume

Section: FROM, TO

DAP: SEQ, PDS

Format

Usage Notes

1. All volumes of the data set must be mounted before BDT will execute your transaction.
2. If you use the UNIT parameter together with this parameter, do not specify a unit address with the UNIT parameter.

Example

Copy the data set DATA.AAA to the multi-volume data set DATA.BBB, and specify that each volume of DATA.BBB is to be mounted on a separate device. Use the TSO prefix (BDT).

```
BDT Q FROM LOCATION(SYSA1)
      DATASET(DATA.AAA) DAP(SEQ) SHR
      TO LOC(SYSA2)
      DATASET(DATA.BBB) OLD VOLUME(BDTDS8,BDTDS9)
      PARALLEL
```

PARMS — Select Messages, Pad Logical Records, and Select, Rename, or Replace PDS Members

Use the PARMS parameter to:

- Select the type of messages you want to receive about a transaction.
- Define a padding character when copying sequential data sets. BDT uses this padding character to pad the “to” data set when the logical record length of the “to” data set is larger than the logical record length of the “from” data set.
- Select the members of the “from” PDS to copy, and specify whether to replace or rename the members in the “to” data set.

Each of these options will be discussed separately. When using the PARMS parameter, however, you can combine the MSG and PAD options. See the examples for an illustration of this.

PARMS(MSG) — Select the Type of Messages You Receive About a Transaction

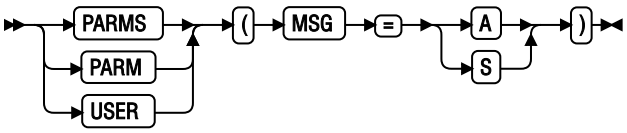
Rules

Optional or Required: Optional

Section: If used with DAP(SEQ), this parameter can be used in either the FROM or TO sections; if used with DAP(PDS), this parameter can be used in the TO section

DAP: SEQ, PDS

Format



MSG=A
specifies that you wish to receive all messages.

MSG=S
specifies that you wish to receive status messages only. This is the default.

Usage Notes

1. You might want to use PARMS(MSG) to reduce the number of messages you get when your transaction completes.
2. Do not confuse the PARMS(MSG) parameter with the MSGCLASS parameter that stands alone. The MSGCLASS parameter directs messages to a particular destination; the PARMS(MSG) parameter controls the type of messages that are sent to you.

PARMS(PAD) — Pad Logical Records in the TO Data Set

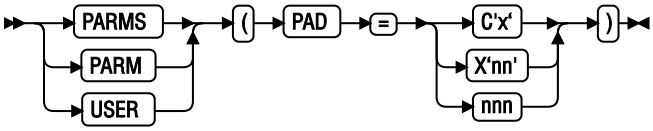
Rules

Optional or Required: Optional

Section: TO

DAP: SEQ

Format



PAD=C‘x’
specifies that the EBCDIC character *x* is to be used for padding. *x* can be any EBCDIC character or a blank. The quotation marks are required.

PAD=X‘nn’
specifies that the hexadecimal number *nn* is to be used for padding. The number *nn* can be 1 or 2 digits, in the range from 0 to FF. The quotation marks are required. Leading zeroes are not required. Hex 40 will pad each line with EBCDIC blanks.

PAD=nnn
specifies that the decimal number *nnn* is to be used for padding. *nnn* can be 1 to 3 digits, in the range from 0 to 255.

If you do not specify a padding character, each record in the “to” data set that is longer than the record sent from the “from” data set is padded with binary zeros (hex 00).

PARMS(R,S,E, or M) — Select, Rename, or Replace PDS Members

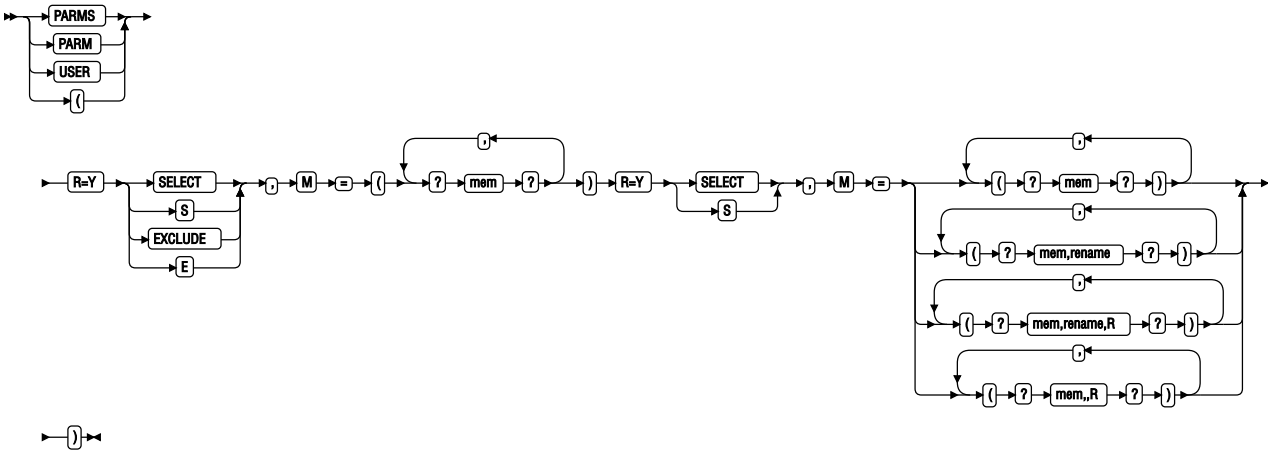
Rules

Optional or Required: Optional

Section: FROM

DAP: PDS

Format



R=Y
specifies that members of the “to” data set are to be replaced by members with the same member names from the “from” data set.

Used alone, R=Y copies the entire PDS, and replaces all members of the “to” data set with members of the “from” data set that have the same member names.

Used with S (SELECT), R=Y copies and replaces those members of the PDS that are selected. Used with E (EXCLUDE), R=Y copies and replaces all members of the PDS except those that are excluded. If you use R=Y with either S (SELECT) or E (EXCLUDE), you must specify R=Y first.

S or SELECT
selects members of the PDS to be copied, copied and renamed, or copied, renamed and replaced, as indicated by R and M.

E or EXCLUDE
excludes (prevents) members of the PDS from being copied and replaced.

M
begins the list of members to be selected or excluded.

mem
is the member to be selected or excluded. If you list only one member, you may omit the parentheses.

The maximum number of members you may specify is 230.

(mem,rename)
assigns a new name, *rename*, in the "to" data set to the member named *mem* when *mem* is copied from the "from" data set. If there is already a member named *rename* in the "to" data set, it is not affected unless you use R=Y. If there is already a member named *mem* in the "to" data set, it is not affected.

Use this only with SELECT.

(mem,rename,R)
assigns a new name, *rename*, in the "to" data set to the member named *mem* when *mem* is copied from the "from" data set. If there is already a member named *rename* in the "to" data set, it is replaced. If there is already a member named *mem* in the "to" data set, it is not affected.

Use this only with SELECT.

(mem,,R)
replaces the member named *mem* in the "to" data set with the member named *mem* from the "from" data set.

Use this to replace individual members of a PDS.

Use this only with SELECT.

Usage Notes

1. Do not use both SELECT and EXCLUDE.
2. If you want to use SELECT or EXCLUDE to copy members with aliases (alternate names), you must specify each alias you want to select or exclude. When the primary member and its aliases are copied, the primary member in the "from" data set becomes the primary member in the "to" data set.
3. If only aliases are copied, the member that is the lowest in the sorting order sequence becomes the primary in the "to" data set.
4. If only one alias is specified, it alone is copied and it becomes the primary in the "to" data set.

Examples

1. Copy the data set FEBDATA to the data set MARDATA and ask to receive only status messages about the transaction. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(FEBDATA) LOCATION(ANODE) OLD DAP(SEQ)
       TO DATASET(MARDATA) LOCATION(BNODE) OLD
       PARMS(MSG=S)
```

2. Copy the data set XDATA to the data set YDATA, and pad YDATA with blanks.

```
BDT Q FROM DATASET(XDATA) LOCATION(XNODE) OLD DAP(SEQ)
      TO DATASET(YDATA) LOCATION(YNODE) OLD
      PARMS(PAD=C' ')
```

3. Copy the data set FEBDATA to the data set MARDATA, pad MARDATA with blanks, and ask to receive only status messages about the transaction.

```
BDT Q  FROM DATASET(FEBDATA) LOCATION(ANODE) OLD DAP(SEQ)
       TO DATASET(MARDATA) LOCATION(BNODE) OLD
       PARMS(MSG=S,PAD=C' ')
```

4. Copy members MEM1 and MEM2 of the PDS PDSDATA to the PDS PDDATA.

```
BDT Q  FROM DATASET(PDSDATA) LOCATION(KGN02) OLD DAP(PDS)
       PARMS(S M=(MEM1,MEM2))
       TO DATASET(PDDATA) LOCATION(KGN03) OLD
```

PASSWORD — Supply a Password for a Password-Protected Data Set

Use this parameter to provide the password for a password-protected data set.

Rules

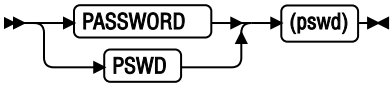
Optional or Required: Required for password-protected data sets

Also Requires: The LABEL parameter for new data sets

Section: FROM, TO

DAP: SEQ, PDS

Format



pswd
is the password. The password may be 1 to 8 alphanumeric characters.

Usage Notes

1. You cannot use this parameter to assign a password.
2. The passwords for the “to” and “from” data sets, if both are password-protected, may be different.
3. See the SECGROUP, SECPSWD, and SECUSER parameters for information on supplying RACF security information.

Example

Copy the data set PASSDATA to the data set SECDATA. Both data sets are password protected. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(PASSDATA) LOCATION(KGN01) DAP(SEQ)
       PASSWORD(3ED4RF)
       TO DATASET(SECDATA) LOCATION(KGN02)
       PASSWORD(CFTGY7)
```

POSITION — Specify the Position of a Data Set on a Tape Volume

Use this parameter to tell MVS the relative position of a data set on a tape volume that contains multiple data sets.

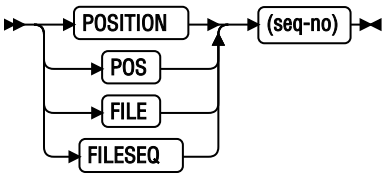
Rules

Optional or Required: Required for tape transfers if the data set is not cataloged and there are multiple files on the volume

Section: FROM, TO

DAP: SEQ

Format



seq-no
is the file sequence number. *seq-no* may be any decimal number from 1 to 9999. If you do not specify this parameter, MVS assumes that the data set is the first data set on the volume.

Example

Copy the data set DATA.AAA to the data set DATA.BBB. Data set DATA.BBB is stored on a tape volume that contains several other data sets. Specify that the file sequence number of DATA.BBB is 2. Use the TSO prefix (BDT).

```
BDT Q FROM LOCATION(SYSA1)
      DATASET(DATA.AAA) DAP(SEQ) SHR
      TO  LOCATION(SYSA2)
      DATASET(DATA.BBB) OLD
      POSITION(2) UNIT(TAPE)
      VOLUME(111111)
```

PRIORITY — Assign a Priority to a Job

Use this parameter to assign a BDT scheduling priority to the BDT job that results from a transaction. For jobs going to the same destination, BDT schedules higher priority jobs before lower priority jobs.

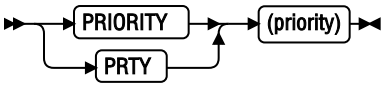
Rules

Optional or Required: Optional

Section: Job definition, FROM, TO

DAP: SEQ, PDS

Format



priority
is the priority to be assigned to the job. Priorities range from 0 to 15, with 15 the highest priority. If you do not specify this parameter a priority of 4 is assumed.

Usage Note

Your installation may restrict you to a limited range of priorities. Check with the system programmer.

Example

Copy a data set to another data set and assign a priority of 8 to the BDT job that performs the copying. Use the TSO prefix (BDT).

```
BDT Q JOB(MYJOB) PRIORITY(8)
      FROM DATASET(ADATA) LOCATION(KGN01) DAP(SEQ)
      TO DATASET(BDATA) LOCATION(KGN02)
```

PROGRAMMER — Supply a Programmer's Name

Use this parameter to supply a name or identification with your transaction, to be used with your installation's accounting routines.

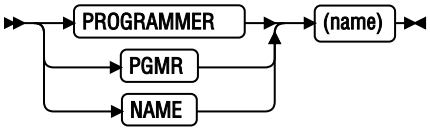
Rules

Optional or Required: Optional

Section: Job definition, FROM, TO

DAP: SEQ, PDS

Format



name
is the name or other identification to be supplied. This information may be up to 20 characters long, and may include any EBCDIC character.

Example

Copy a data set to another data set and supply your name with the transaction. Use the TSO prefix (BDT).

```
BDT Q PGMR(BROMLEY)
      FROM DATASET(DDATA) LOCATION(ANODE) DAP(SEQ)
      TO DATASET(EDATA) LOCATION(BNODE)
```

PROTECT — Provide RACF Protection for a New Data Set

Use this parameter to provide RACF protection for a newly created data set.

Rules

Optional or Required: Optional

Section: TO

DAP: SEQ, PDS

Format

Usage Notes

1. If you use this parameter, your installation must have RACF Release 1.6 or higher or the transaction will fail.
2. If you use PROTECT with an existing data set an allocation error will occur.
3. For more information, see the PROTECT keyword in theJCL manual that is appropriate for your installation.

Example

Copy an existing data set, BOBSDATA, to a new data set, JOESDATA, and RACF protect JOESDATA. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(BOBSDATA) OLD LOCATION(BOBSNODE)
       DAP(SEQ)
       TO DATASET(JOESDATA) LOCATION(JOESNODE)
       PROTECT NEW VOLUME(KGN02)
       SPACE(1,1) CYL UNIT(SYSDA)
       BLKSIZE(800) LRECL(80) RECFM(F)
```

RECFM — Specify the Record Format of a Data Set

Use this parameter to tell BDT the record format of a data set.

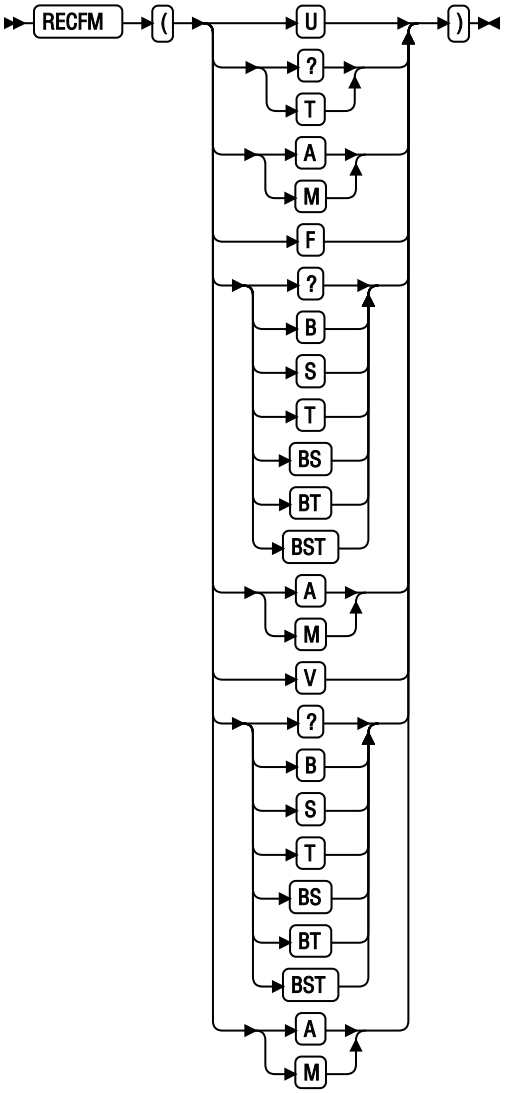
Rules

Optional or Required: Required on uncataloged data sets and new data sets

Section: TO

DAP: SEQ, PDS

Format



A
specifies that records contain ANSI device control characteristics.

B
specifies that the records are blocked.

F
specifies that the records are fixed-length.

M
specifies that the records contain machine code control characteristics.

S
specifies that the records are standard blocks of fixed length or spanned blocks of variable length.

T
specifies that, if required, records may be written onto overflow tracks.

U
specifies that the records are of an undefined length. This is the default.

V
specifies that the records are variable length non-ASCII.

Usage Note

Do not code RECFM in the FROM section. If you do so and the data set's data control block (DCB) information is incompatible with the information contained in the RECFM parameter, the transaction will fail.

Example

Copy the data set BDATA to a new data set, CDATA. Specify a record format of variable length non-ASCII for CDATA. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(BDATA) DAP(PDS) LOCATION(NODEB)
       TO DATASET(CDATA) LOCATION(NODEC) NEW
       RECFM(V) LRECL(255) BLKSIZE(259) DIR(12)
       SPACE(1,2) CYL DSORG(PO) VOLUME(111111)
       BUFL(4096) UNIT(3380)
```

RELEASE — Release Unused Space in a New DASD Data Set

Use this parameter to release unused space in a new "to" data set that is on a direct access storage device (DASD). The space is to be released after the transaction has completed.

Rules

Optional or Required: Optional

Also Requires: The NEW and SPACE parameters, and either the BLOCK, CYLINDERS, or TRACKS parameter

Section: TO

DAP: SEQ, PDS

Format

Usage Notes

1. Using this parameter helps make efficient use of DASD space. Use it when you may have asked for more space than is needed for a new data set.
2. Do not use this parameter for tape data sets.
3. If you code this parameter in the FROM section, BDT will accept the parameter but will not use it.
4. For a list of parameters that can be used to allocate a new data set, see Appendix A, "Parameters That Require Other Parameters," on page 83.

Example

Copy a data set to a new sequential DASD data set, and release unused direct access space in the new data set after the transaction is completed. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(ONEDATA) LOCATION(ONENODE) DAP(SEQ)
       TO DATASET(TWODATA) LOCATION(TWONODE)
       NEW SPACE(1,1) CYL BLKSIZE(800)
       LRECL(80) RECFM(FB) UNIT(3350)
       RELEASE VOLUME(111111)
```

RETPD — Specify a Retention Period for a Data Set

Use this parameter to specify a retention period of a certain number of days for a data set. The data set cannot be written over or deleted until the retention period has ended.

Rules

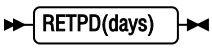
Optional or Required: Optional

Also Requires: The LABEL parameter for new data sets

Section: TO

DAP: SEQ, PDS

Format



days
is the number of days the data set is to be protected from being written over or deleted. *days* may be any decimal number from 0 to 9999.

Usage Notes

1. If you want to protect a data set from being written over or deleted before a certain date, use the EXPDT parameter.
2. If you code the RETPD parameter in the FROM section, BDT will accept the parameter but will not use it.
3. For more information about the use of RETPD, see the JCL manual that is appropriate for your installation.

Example

Copy the data set MYDSN to the data set THATDSN and assign a retention period of 100 days to THATDSN. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(MYDSN) LOCATION(MYNODE) DAP(SEQ)
       TO DATASET(THATDSN) LOCATION(THATNODE)
       RETPD(100)
```

ROUND — Allocate Space in Whole Cylinders

Use this parameter when allocating a new “to” data set that is to be stored on a direct access storage device (DASD). This parameter requests that MVS allocate direct access space for the data set in whole cylinder increments, rounding up to the next full cylinder where necessary.

Rules

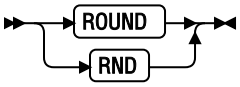
Optional or Required: Optional

Also Requires: The NEW and SPACE parameters, and either the BLOCK, CYLINDERS, or TRACKS parameter

Section: TO

DAP: SEQ, PDS

Format



Usage Notes

1. If you code the ROUND parameter in the FROM section, BDT will accept the parameter but will not use it.
2. For a list of parameters that can be used to allocate a new data set, see Appendix A, “Parameters That Require Other Parameters,” on page 83.

Example

Copy a data set to a new data set. Allocate space for the new data set in tracks, but request that the space be rounded up to a whole cylinder. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(SMP.DATA) LOCATION(SMPNODE) DAP(SEQ)
       TO DATASET(NEW.DATA) LOCATION(SMNODE)
       NEW SPACE(12,1) BLKSIZE(2660) LRECL(133) RECFM(VB)
       TRACKS ROUND UNIT(SYSDA)
       VOLUME(11111) DISP(CATLG,DELETE)
```

SECGROUP — Supply a Security Group ID for a Data Set That Is Protected with RACF

Use this parameter to supply your RACF group ID when accessing RACF-protected resources.

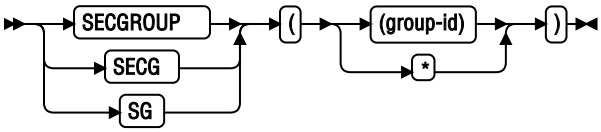
Rules

Optional or Required: Optional

Section: FROM, TO

DAP: SEQ, PDS

Format



*****
with RACF Release 1.6 or later, will cause BDT to use the default user ID or password or both if they are not coded with the SECUSER and SECPSWD parameters.

group-id
is the RACF group ID (1 to 8 alphanumeric characters).

If you do not specify this parameter, a default group ID will be provided:

- If you are at an MCS or JES3 console, the default is provided by user exit BDTUX19. User exit BDTUX19 is described in *z/OS BDT Installation*.
- If you are at a TSO terminal, the default is the group name you used to log on to the system.
- If you are running a batch job, the default is the group ID you specified on the JOB card.

Example

Copy the RACF-protected data set MYDSN to the RACF-protected data set YOURDSN, supplying RACF group IDs to access the data sets. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(MYDSN) LOCATION(KGN01) DAP(SEQ)
       SECGROUP(MYGROUP)
       TO DATASET(YOURDSN) LOCATION(KGN02)
       SECGROUP(YOURGROUP)
```

SECPSWD — Supply a Security Password for a Data Set That Is Protected with RACF

Use this parameter to supply the password for a RACF-protected data set.

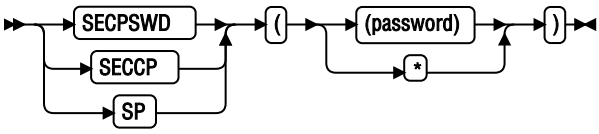
Rules

Optional or Required: Optional

Section: FROM, TO

DAP: SEQ, PDS

Format



*****
Will cause BDT to use the default user ID or group ID or both if they are not specified with the SECUSER and SECGROUP parameters.

password
is the password for the data set (1 to 8 alphanumeric characters).

If you do not specify this parameter, a default password will be provided for a data set that is at the node where you submit the transaction:

- If you are at a TSO terminal, the default is your logon password.
- If you are running a batch job, the default is the password you specified on the JOB card.

You must use SECPSWD(*) to request the default password for a data set that is at another node.

Usage Notes

1. You cannot use this parameter at an MCS or JES3 console.
2. Passwords may be encrypted if:
   - Your installation has defined an encryption algorithm in user exit BDTUX19.

   Both the “from” and “to” nodes must support the same encryption algorithm in order for the password to be encrypted. See your system programmer for further information.
3. If you code your user ID with the SECUSER parameter, do not default your group ID by using the asterisk (*) in the SECGROUP parameter, and do not code the password in the SECPSWD parameter, BDT will default the password to blanks. See the preceding figure.

If you code this for the SECUSER parameter:	And code this for the SECGROUP parameter:	The default for SECPSWD will be:
(user ID)	(user ID)	blanks

If you code this for the SECUSER parameter:	And code this for the SECGROUP parameter:	The default for SECPSWD will be:
(user ID)	(*)	default
(user ID)		blanks

Example

Copy the data set MYDSN to the data set YOURDSN, both of which are RACF-protected. Supply the passwords with the SECPSWD parameter. Use the TSO prefix (BDT).

```
BDT Q FROM DATASET(MYDSN) LOCATION(SPK01) DAP(SEQ)
      SECPSWD(MYPSWD)
      TO DATASET(YOURDSN) LOCATION(SPK02)
      SECPSWD(YOURPSWD)
```

SECUSER — Supply a Security User ID for a Data Set That Is Protected with RACF

Use this parameter to supply your user security ID for RACF-protected data sets.

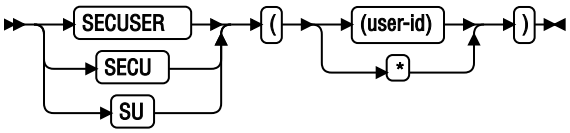
Rules

Optional or Required: Optional

Section: FROM, TO

DAP: SEQ, PDS

Format



*****
with RACF Release 1.6 or later, will cause BDT to use the default password or group ID or both if they are not specified with the SECPSWD and SECGROUP parameters.

user-id
is your user security ID (1 to 8 alphanumeric characters).

If you do not specify this parameter, a default user ID will be provided automatically for a data set that is at the node where you submit the transaction:

- If you are at an MCS or JES3 console, the default will be provided by user exit BDTUX19. User exit BDTUX19 is described in *z/OS BDT Installation*.
- If you are at a TSO terminal, the default is your logon user ID.
- If you are running a batch job, the default is the user ID you specified on the JOB card.

You must use SECUSER(*) to request the default value for a data set that is located at another node.

Usage Note

With RACF Release 1.6 or later, requesting the default by coding SECUSER(*) causes BDT to use the default value contained in the BTUDEFU text unit for SECUSER.

Example

Copy the data set MYDSN to the data set YOURDSN. Both data sets are RACF-protected. Supply your security ID with the SECUSER parameter. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(MYDSN) LOCATION(KGN01) DAP(SEQ)
       SECUSER(MYUSERID)
       TO DATASET(YOURDSN) LOCATION(KGN02)
       SECUSER(YOURUSERID)
```

SHR — Permit Shared Use of a Data Set

Use this parameter to indicate to MVS that the data set exists and that other address spaces within MVS, in addition to your BDT subsystem, may share the use of the data set.

Rules

Optional or Required: Optional

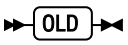
Section: FROM, TO

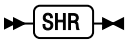
DAP: SEQ, PDS

Format

SHR is one of two data set status parameters that can be specified in the FROM section of a transaction. In addition, SHR is one of four data set status parameters that can be specified in the TO section of a transaction. The options for the FROM and TO sections are discussed separately in the preceding section.

FROM Options:



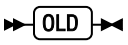


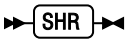
Use only one of the options in the FROM section of a transaction.

SHR
specifies that the “from” data set is an existing data set and that other users may share the use of it with your BDT subsystem while the transaction is being processed. This is the default for the “from” data set.

OLD
specifies that the “from” data set is an existing data set, and that your BDT subsystem has exclusive use of the data set while the transaction is being processed.

TO Options:





Use only one of the options in the TO section of a transaction.

SHR
specifies that the “to” data set is an existing data set and that other users may share the use of it with your BDT subsystem.

OLD
specifies that the “to” data set is an existing data set, and that the copied data set is to overwrite the existing data set. This parameter gives your BDT subsystem exclusive use of the data set.

This is the default for the TO section.

MOD
specifies that the “to” data set is an existing data set, and that the copied data set is to be added to it. This parameter also gives your BDT subsystem exclusive use of the “to” data set.

NEW
specifies that the “to” data set is a new data set to be allocated in the transaction. This parameter also gives your BDT subsystem exclusive use of the “to” data set.

Example

Copy the data set PDS1.AA to the data set PDS2.BB. Both are existing data sets. Request shared use of PDS1.AA and exclusive use of PDS2.BB. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(PDS1.AA) LOCATION(THISNODE)
       DAP(PDS) SHR
       TO DATASET(PDS2.BB) LOCATION(OTHRNODE) OLD
```

SPACE — Request Space for a New Data Set

Use this parameter to request primary and secondary space for a new data set stored on a direct access storage device (DASD).

Rules

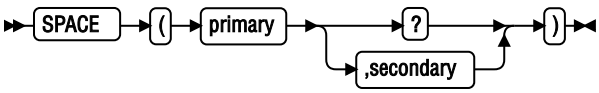
Optional or Required: Required for new DASD data sets

Also Requires: The NEW parameter and either the BLOCK, CYLINDERS, or TRACKS parameter

Section: TO

DAP: SEQ, PDS

Format



primary
is the amount of primary space to be allocated to the data set. *primary* may be a number from 0 to 999999, depending on the limits at your installation.

secondary
is the amount of secondary space to be allocated to the data set. *secondary* may be a number from 1 to 999999, depending on the limits of your installation. Secondary space is space to be used only if the data set exceeds the primary space.

Usage Notes

1. Specify the units of primary and secondary space by coding another parameter with the SPACE parameter:
   - Use the BLOCK parameter to allocate the space in blocks.
   - Use the CYLINDERS parameter to allocate the space in cylinders.
   - Use the TRACKS parameter to allocate the space in tracks.
2. This parameter is invalid with the OLD, MOD, or SHR parameters.

3. Do not use this parameter for data sets stored on tape.
4. For a list of parameters that can be used to allocate a new data set, see Appendix A, “Parameters That Require Other Parameters,” on page 83.

Example

Copy the data set BDATA to a new data set, CDATA. Specify primary space of 1 and secondary space of 2. Use the CYLINDERS parameter to specify cylinder units. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(BDATA) DAP(PDS) LOCATION(NODEB)
       TO DATASET(CDATA) LOCATION(NODEC) NEW
       SPACE(1,2) LRECL(255) BLKSIZE(259) RECFM(V)
       DIR(12) CYLINDERS DSORG(PO) VOLUME(111111)
       DISP(KEEP,DELETE) UNIT(3380)
       BUFL(4096)
```

SYSTEM — Specify the BDT Node in a Poly-BDT Complex

Use this parameter in a poly-BDT complex to identify the BDT node on which you wish to have your transaction executed.

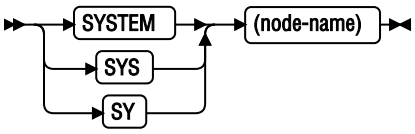
Rules

Optional or Required: Optional, unless you want the transaction to be executed at a node other than the node that has been defined as the default for your installation

Section: Job Definition, FROM, TO

DAP: SEQ, PDS

Format



node-name
is the name of the BDT node at which the transaction is to be executed. If you do not specify this parameter, the node that has been defined as the default for your installation is assumed. Contact your system programmer for the name of the default node and for a list of valid node names.

Usage Note

This parameter cannot be used in a transaction definition stored in a GMJD library.

Examples

1. Submit a transaction from a TSO terminal in a poly-BDT complex. Request that the transaction be executed at node BDT01.

```
BDT SY(BDT01) Q  FROM DATASET(BIGDATA) LOCATION(KGN01)
                 DAP(SEQ)
                 TO DATASET(LITDATA) LOCATION(KGN02)
```

2. Submit a transaction from a JES3 console in a poly-BDT complex. Request that the transaction be executed at node BDT01.

```
*S,BDT SY(BDT01) Q  FROM DATASET(BIGDATA) LOCATION(KGN01)
                    DAP(SEQ)
                    TO DATASET(LITDATA) LOCATION(KGN02)
```

TIME — Specify the Maximum Processor Time for a Transaction

Use this parameter to put a maximum on the amount of processor time you wish to allow your transaction to use. If your transaction exceeds the maximum, BDT will remove it from the system.

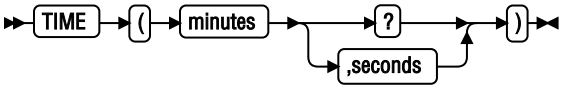
Rules

Optional or Required: Optional

Section: Job definition, FROM, TO

DAP: SEQ, PDS

Format



minutes
is the minutes of processor time the transaction is to be allowed. *minutes* can be any integer up to 1440.

seconds
is the seconds of processor time the transaction is to be allowed. *seconds* can be from 1 to 59.

If you do not specify this parameter, a maximum processor time of 4 minutes is assumed.

Usage Notes

1. Use this parameter if you suspect that there are problems that will cause your transaction to use an excessive amount of processor time.
2. BDT will remove a transaction from the system when it exceeds its time limit, even if the data transfer is only partially completed.

Example

Copy the data set ONEDATA to the data set TWODATA, and set a maximum processor time for the transaction of 15 minutes. Use the JES3 prefix (*S,BDT).

```
*S,BDT Q TIME(15)  FROM DATASET(ONEDATA) LOCATION(ONENODE)
                   DAP(SEQ)
                   TO DATASET(TWODATA) LOCATION(TWONODE)
```

TO — Begin the TO Section

Use this parameter to mark the beginning of the TO section in the transaction. The parameters following TO describe the “to” node and data set.

Rules

Optional or Required: Required

Section: TO

DAP: SEQ, PDS

Format

Usage Notes

1. See Chapter 1, "Writing and Submitting File-to-File Transactions," on page 1 for general information on the TO section.
2. You may use the FROM or TO parameters more than once in a transaction. Each time you use the FROM parameter, you must use a TO parameter if you want to add parameters that belong in the TO section.
3. Once BDT finds the TO parameter, all parameters that are not job definition parameters are assumed to be TO parameters unless the FROM parameter is used to begin more FROM parameters.

Examples

1. Copy the data set TRYDATA to the data set SETDATA, which is located at node BNODE. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(TRYDATA) LOCATION(ANODE) DAP(SEQ)
       TO DATASET(SETDATA) LOCATION(BNODE)
```

2. Copy the data set TRYDATA to the data set SETDATA. Break the FROM and TO sections of the transaction into several parts.

```
BDT Q  FROM DATASET(TRYDATA) DAP(SEQ)
       TO DATASET(SETDATA)
       FROM LOCATION(ANODE)
       TO LOCATION(BNODE)
```

TRACKS — Allocate Space for a New Data Set in Track Units

Use this parameter when allocating a new "to" data set that is to be stored on a direct access storage device (DASD). This parameter requests that space for the new data set be allocated in track units.

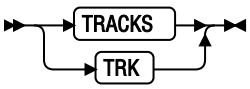
Rules

Optional or Required: Optional

Also Requires: The NEW and SPACE parameters

Section: TO

DAP: SEQ, PDS



Usage Notes

1. This parameter is invalid with the BLOCK and CYLINDERS parameters.
2. For a list of parameters that can be used to allocate a new data set, see Appendix A, "Parameters That Require Other Parameters," on page 83.

Example

Copy the data set BSET to a new data set DSET, and allocate space for the new data set in track units. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(BSET) LOCATION(KGN01) DAP(SEQ)
       TO DATASET(DSET) LOCATION(KGN02)
       NEW SPACE(1,1)
       TRACKS UNIT(SYSDA) VOLUME(111111)
       BLKSIZE(800) LRECL(80) RECFM(F)
```

TRTCH — Specify the Translation Technique for Tape

Use this parameter to indicate the translation technique for 7-track tape.

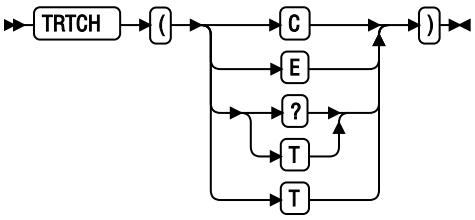
Rules

Optional or Required: Optional

Section: FROM, TO

DAP: SEQ

Format



C
specifies data conversion with odd parity and no translation.

E
specifies even parity with no translation and no conversion.

T
specifies odd parity with no conversion. When reading, BCD is to be translated to EBCDIC; when writing, EBCDIC is to be translated to BCD.

ET
specifies even parity with no conversion. When reading, BCD is to be translated to EBCDIC; when writing, EBCDIC is to be translated to BCD.

Example

Copy the data set DEPTA, which is on an 800 bpi 7-track magnetic tape, to the new data set NAMES. For a translation technique, specify even parity with no conversion. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(DEPTA) LOCATION(KGN01) DAP(SEQ)
       UNIT(TAPE7)
       TO DATASET(NAMES) LOCATION(KGN02)
       NEW UNIT(TAPE) VOLSER(11111)
       TRTCH(ET) DEN(2)
       BLKSIZE(259) LRECL(255)
       RECFM(V)
```

UCOUNT — Specify the Unit Count for a Multivolume Data Set

Use this parameter to indicate the number of devices to be allocated to a multivolume data set. This parameter does not allocate specific volumes.

Rules

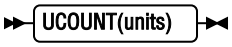
Optional or Required: Optional

Also Requires: The UNIT parameter

Section: TO

DAP: SEQ

Format



units
is the number of devices to be allocated. *units* may be any decimal number from 1 to 59.

Usage Notes

1. You might want to use this parameter when allocating scratch tapes.
2. This parameter is invalid with the PARALLEL parameter.

Example

Copy the data set ABA.DATA to the data set ABA.DATB. ABA.DATB is a new DASD data set. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(ABA.DATA) LOCATION(SYSA1)
       DAP(SEQ) SHR
       TO DATASET(ABA.DATB) LOCATION(SYSA2)
       UNIT(SYSDA) UCOUNT(1) VOLUME(ABADS8)
       NEW LRECL(80) BLKSIZE(800) RECFM(F)
       SPACE(12,1) CYL DSORG(PS) DISP(KEEP,DELETE)
```

UNIT — Specify the Device on Which a Data Set Is Located

Use this parameter to identify the device, by unit group name, unit device type, or unit device address, on which a data set is located or is to be allocated.

Rules

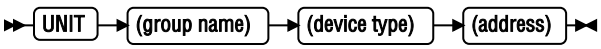
Optional or Required: Required for new data sets and for uncataloged data sets.

Also Requires: The VOLUME parameter

Section: FROM, TO

DAP: SEQ, PDS

Format



group name
is the unit group name. Unit group names are defined by your installation. They identify groups of devices that may or may not be all of the same type.

device type
is the unit device type. Unit device types identify devices of a single type, such as 3350 or 3380 direct access storage devices.

address
is the unit address. Unit addresses are 3-character or 4-character identifiers of particular devices. They are defined by your installation.

See your system programmer for valid unit group names, unit device types, and unit addresses.

Usage Note

For identifying devices for multivolume data sets, see the PARALLEL and UCOUNT parameters.

Example

Copy the data set DATAA to a new tape data set DATAB. The device type of DATAA is 3350; the device type of DATAB is TAPE. Use the TSO prefix (BDT).

```
BDT Q FROM DATASET(DATAA) DAP(SEQ)
      UNIT(3350) VOLUME(11111)
      TO DATASET(DATAB)  UNIT(TAPE) NEW LABEL(NL)
      RECFM(F) LRECL(80) BLKSIZE(800) VOLUME(22222)
```

VOLREF — Locate a Data Set on the Same Volume as a Cataloged Data Set

Use this parameter to locate a data set on the same volume or volumes as a cataloged data set.

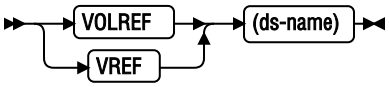
Rules

Optional or Required: Optional

Section: FROM, TO

DAP: SEQ, PDS

Format



ds-name
is the name of a cataloged data set. BDT obtains the volume serial number or numbers for the data set and then uses those numbers to locate another data set (specified with the DATASET parameter) on the same volume or volumes.

Usage Notes

1. This parameter is invalid with the VOLUME parameter.
2. If a transaction is entered by a TSO user and the data set name is not enclosed in single quotation marks, the TSO prefix specified for the user is added as a prefix to the data set name.

Example

Copy the data set DATA.NOV to the new data set DATA.DEC. Put the new data set on the same volume as DATA.OCT. Use the TSO prefix (BDT).

```
BDT  Q JOB(VOLUME5)
       FROM LOC(SYSA1)
       DA(DATA.NOV) DAP(SEQ) SHR
       TO  LOC(SYSA2)
       DA(DATA.DEC) NEW LRECL(80) BLKSIZE(80) RECFM(F)
       SPACE(55) TRK DSORG(PS) DISP(CATLG,DELETE)
       VOLREF(DATA.OCT)
```

VOLSEQ — Specify the Volume Sequence Number

Use this parameter to identify which volume of a multivolume data set is to be used to begin processing.

Rules

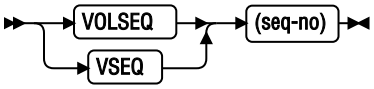
Optional or Required: Optional

Also Requires: The VOLUME parameter. If the data set is not cataloged, this parameter also requires the UNIT parameter.

Section: FROM, TO

DAP: SEQ

Format



seq-no
is a number from 0 to 255 that identifies the volume you want used to begin processing.

Example

Copy the data set BDTREG3.KGN1A to the new data set BDTREG3.KGN2A. The "from" data set is a multivolume data set; use the VOLSEQ parameter to specify that you want to begin processing with volume 0. Use the TSO prefix (BDT).

```
BDT Q FROM DA(BDTREG3.KGN1A) LOC(SYSA1)
      DAP(SEQ) SHR
      VOLUME(BDTDS8) UNIT(3330-1) VOLSEQ(0)
      TO DA(BDTREG3.KGN2A) LOC(SYSA2)
      NEW LRECL(133) BLKSIZE(2660) RECFM(VB)
      SPACE(140) TRK DSORG(PS) DISP(CATLG,DELETE)
      VOLUME(BDTDS8) UNIT(3350)
```

VOLUME — Specify a Volume Serial Number

Use the VOLUME parameter to identify the serial number of the volume on which a data set is located.

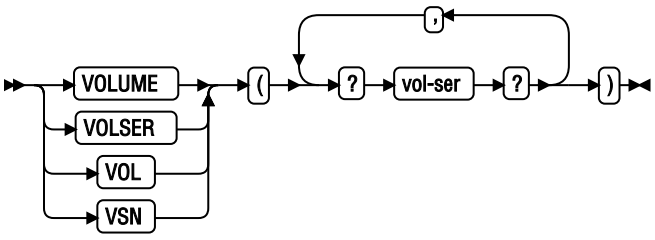
Rules

Optional or Required: Optional

Section: FROM, TO

DAP: SEQ, PDS

Format



vol-ser
is the serial number of the volume on which the data set is located. If you do not specify this parameter, the volume serial number in the data set label is assumed for existing data sets.

Usage Note

If the "to" data set is a new, multivolume data set, the primary SPACE parameter must specify an amount of space that is equal to or less than the space remaining on the first volume. Otherwise, the transaction will fail.

Example

Copy the data set ONEDATA to the new data set TWODATA. Specify that the serial number of the volume on which TWODATA is to be allocated is 111111. Use the TSO prefix (BDT).

```
BDT Q  FROM DATASET(ONEDATA) LOCATION(KGN01) DAP(PDS)
       TO DATASET(TWODATA) LOCATION(KGN02)
       NEW LRECL(255) BLKSIZE(259) RECFM(V) DIR(12)
       SPACE(1,2) CYL DISP(KEEP,DELETE)
       BUFL(4096) DSORG(PO)
       VOLUME(111111) UNIT(3380)
```

Appendix A. Parameters That Require Other Parameters

The tables on the following pages provide a quick reference to parameters that require other parameters.

Parameters That Allocate New "To" Data Sets

Parameter	Purpose	Other Parameters Required
NEW	Allocate a new "to" data set	BLKSIZE, LRECL, RECFM, UNIT, VOLUME Also, SPACE, and either BLOCK, CYLINDERS, or TRACKS for DASD data sets Also, DIR for partitioned data sets

These parameters can be added for DASD data sets:

Parameter	Purpose	Other Parameters Required
ALX	Allocate the space in several contiguous areas	NEW, BLKSIZE, LRECL, RECFM, SPACE and either BLOCK, CYLINDERS, or TRACKS
BUFL	Define the size of the buffer in the I/O buffer pool	DIR if the data set is partitioned
CONTIG	Allocate the space in a contiguous area	
DCBDS	Read DCB information for the new data set from the label of an existing data set	
DSORG	Specify the organization for the new data set	
MXIG	Allocate the largest contiguous area on a volume for the data set	
RELEASE	Release unused space from the data set after the transaction completes	
ROUND	Allocate the space in whole cylinders	

Parameters That Control DTC Networks

Parameter	Purpose	Other Parameters Required
NETCOND	Tell BDT what to do if a predecessor transaction fails	NETID
NETHOLD	Assign hold counts to transactions	NETID
NETREL	Name the successor transactions to a transaction	NETID, NETHOLD, JOBNAME

Other Parameters That Require Additional Parameters

Parameter	Purpose	Other Parameters Required
DUMMY	Specify a dummy data set for the “from” or “to” data set	LRECL, RECFM, and either DCBDS or BLKSIZE
EXPDT	Assign an expiration date to a data set	LABEL if the data set is a new data set
INTRDR	Make the “to” data set the MVS internal reader	BLKSIZE
PARALLEL	Specify that an existing data set occupies more than one volume and that each volume is to be mounted on a separate device	VOLUME
PASSWORD	Supply the password for a password-protected data set	LABEL if the data set is new
RETPD	Assign a retention period to a data set	LABEL if the data set is new
UCOUNT	Specify the number of devices to be allocated to a multivolume data set that is not on specific volumes	UNIT
UNIT	Identify the device on which a data set is located	VOLUME
VOLSEQ	Identify the volume of a multivolume data set with which to begin processing	VOLUME, and UNIT if the data set is uncataloged

Appendix B. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed email message to mhvrcfs@us.ibm.com.

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or `3.1` or `3.1.1`. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number `3.1`) are mutually exclusive alternatives. If you hear the lines `3.1 USERID` and `3.1 SYSTEMID`, your syntax can include either `USERID` or `SYSTEMID`, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3* , 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.

3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line `6.1+ data area`, you must include at least one data area. If you hear the lines `2+, 2 HOST,` and `2 STATE`, you know that you must include `HOST, STATE,` or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for the Knowledge Centers. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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
Site Counsel
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 document 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, JES2, JES3, and MVS™, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and Trademark information (www.ibm.com/legal/copytrade.shtml).

GLOSSARY

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.

catalog
The collection of all data set indexes that is used by the control program to locate a volume containing a specific data set.

checkpoint data set
See TQI checkpoint data set.

DAP
Dynamic application program.

dependent transaction control (DTC)
A method of controlling the scheduling of file-to-file transactions by organizing the transactions into a network in which some transactions wait for the completion of other transactions before being scheduled.

DTC
Dependent transaction control.

dynamic application program (DAP)
A part of BDT that performs a particular function, especially the transfer of data.

generic master job definition 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.

GMJD
Generic master job definition.

Interactive System Productivity Facility (ISPF)
A licensed program that provides menus and data entry panels for using system functions.

ISPF
Interactive System Productivity Facility.

JES3
Job entry subsystem 3.

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

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.

MCS
Multiple console support.

multiple console support (MCS)
A feature of MVS that permits selective message routing to multiple operator's consoles.

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.

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

RACF
Resource Access Control Facility.

Resource Access Control Facility (RACF)
A licensed program 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.

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

SNA
Systems Network Architecture.

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.

Time sharing option (TSO)
A component of MVS that provides interactive computing from remote stations.

TQI
Transaction queuing integrity.

TQI checkpoint data set
In BDT, a data set on which the transaction queuing integrity (TQI) facility records user-submitted commands and file-to-tile transactions before sending them to the BDT address space. Should a command or transaction fail to reach the BDT address space, BDT automatically recovers it from the TQI checkpoint data set and attempts the transfer again.

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 that BDT is to copy, the data set into which BDT is to write the copied data set, and parameter values that BDT is to use while copying the data set.

transaction queuing integrity
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.

TSO
Time sharing option.

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.

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

Index

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z

SC14-7583-30