

GC28-1300-2  
File No. S370-36

## **Program Product**

## **MVS JCL**

**MVS/System Product - JES2 Version 1  
5740-XY5**

**MVS/System Product - JES3 Version 1  
5740-XYN**

The IBM logo, consisting of the letters 'IBM' in a bold, sans-serif font, with each letter formed by eight horizontal stripes of varying lengths.

**This edition applies to the following program products:**

**MVS/System Product - JES2 Version 1 Release 3.4 (Program No. 5740-XYS)**

**MVS/System Product - JES3 Version 1 Release 3.4 (Program No. 5740-XYN)**

**MVS/370 Data Facility Product (DFP) Release 1.1 (Program No. 5665-295)**

**Resource Access Control Facility (RACF) Version 1 Release 6 and later (Program No. 5740-XXH)**

**Do not replace your existing documentation until your system consists of the above releases (1) of the base control program with JES2 or JES3 and (2) of DFP.**

**Note:** Because the book has been extensively revised, JES2 installations using Version 1 Release 3.4 should use this edition, even though the previous edition also reflects Version 1 Release 3.4.

### **Third Edition (December, 1984)**

This is a major revision of, and obsoletes, GC28-1300-1. See the Summary of Amendments following the Contents for a summary of the changes made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to the program releases listed in the box above and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department D58, Building 920-2, PO Box 390, Poughkeepsie, New York, U.S.A. 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

# Preface

This publication describes how to control job processing and the resources needed to run a job. These resources include storage, data sets, devices, and volumes. Job and resource control is specified in job control language (JCL) statements, job entry subsystem 2 (JES2) control statements, and job entry subsystem 3 (JES3) control statements.

## Who Should Use This Publication

This book is needed by programmers who code JCL, JES2, and JES3 control statements. Those using this book should understand the concepts of job management and data management.

## Major Sections of This Publication

**Part 1. Introduction:** In this part, chapter 1 introduces job control. It is intended primarily for the programmer who is inexperienced in job control.

Chapter 2 explains the coding conventions for JCL, JES2, and JES3 control statements. These coding conventions are used throughout the book.

**Part 2. Guide to Job and Step Control:** This part discusses how to control jobs and steps using JCL, JES2, and JES3 control statements. It contains:

- The background needed to understand why you should code each parameter.
- Examples to show when to code the parameters.
- Descriptions of how to code combinations of parameters to obtain particular results.

The descriptions of job and step control are grouped into the following chapters:

**Chapter 3. Guide to Job Control** tells how to control the system's handling of your job. It describes the parameters you can code on the JCL JOB statement and on the JES2 or JES3 control statements to direct the system.

**Chapter 4. Guide to Step Control** tells how to control the system's handling of your job step. It describes the parameters you can code on the JCL EXEC statement and on the JES2 or JES3 control statements to direct the system.

**Chapter 5. Guide to Job and Step Control** tells how to control the system's handling of your job and its steps. It describes the parameters you can code on the JCL JOB statement, on the JCL EXEC statement, and on the JES2 or JES3 control statements to direct the system.

The parameters and control statements in this chapter not only influence the job and its steps, but they influence each other.

**Chapter 6. Guide to Data Allocation Control** tells how to control the system's allocation of data resources. It describes the parameters you can code on the JCL JOB and EXEC statements and on the JES3 control statements to direct allocation.

**Part 3. Guide to Data Set Control:** This part discusses how to control your job's data set resources using JCL DD statements and JES2 or JES3 control statements.

**Chapter 7. Guide to Specifying Data Set Information** tells how to control the system's handling of your data sets. It describes the parameters you can code on JCL DD statements and on JES2 or JES3 control statements to tell the system the following:

- Data set information
- The location of a data set
- The size of a data set
- Data attributes
- Data set processing options

**Chapter 8. Guide to Special Data Sets** tells how to use special data sets. It describes the parameters on the JCL, JES2, and JES3 control statements for special data sets.

**Chapter 9. Guide to Cataloged and In-Stream Procedures** tells how to use cataloged and in-stream procedures of JCL statements. It describes how to:

- Create procedures and place them in catalogs.
- Modify and add parameters and statements to cataloged procedures.
- Use symbolic parameters in cataloged procedures.

**Part 4. Reference to Job Control Statements and Parameters:** This part details the coding of each JCL, JES2, and JES3 control statement and of each parameter, in alphabetical order by statement. The chapters are:

Chapter 10. Coding the JOB Statement  
Chapter 11. Coding the EXEC Statement  
Chapter 12. Coding the DD Statement  
Chapter 13. Coding Special DD Statements  
Chapter 14. Coding the OUTPUT JCL Statement  
Chapter 15. Coding Special JCL Statements  
Chapter 16. Coding JES2 Control Statements  
Chapter 17. Coding JES3 Control Statements

For each statement and parameter, this part gives the following information, as needed:

- Parameter type.
- Purpose.
- References to related information in this book or other IBM publications.
- Syntax and coding rules.
- Parameter or subparameter definitions.
- Defaults.
- Overrides.
- Mutually exclusive parameters and subparameters.
- Relationship to other parameters and control statements.
- Programming considerations.



- System action in response to the parameter or subparameter.
- Examples.
- Other information required to code the statement or parameter.

**Part 5. Reference Tables:** In this part, chapter 18 contains reference tables that summarize some job control information. Figures, which follows the Contents, lists the tables.

## Guide to Using this Publication

If your system contains MVS/System Product - JES2 Version 1 program number 5740-XYX, JES2 information in this manual refers to the JES2 function in the MVS/System Product Version 1, unless otherwise noted.

If your system contains MVS/System Product - JES3 Version 1 program number 5740-XYN, JES3 information in this manual refers to the JES3 function in the MVS/System Product Version 1, unless otherwise noted.

Your system must contain Resource Access Control Facility (RACF) Program Product, program number 5740-XXH, in order for you to specify the PROTECT parameter on your DD statements.

**Restrictions on Use of SYSCHK DD Statement and DD Statement RESTART Parameter:** If your system contains (1) MVS/System Product - JES2 Version 1 Release 3 (5740-XYX) or (2) MVS/System Product - JES3 Version 1 Release 3 (5740-XYN) or (3) any subsequent release of these products, but does not contain MVS/370 Data Facility Product program number 5665-295, do not use the SYSCHK DD statement or the RESTART parameter on the JOB statement.

If your system contains the MVS/370 Data Facility Product with JES2 or JES3 Release 3 or a subsequent release, you can use the SYSCHK DD statement or the RESTART parameter on the JOB statement, with certain restrictions. For detailed information on checkpoint/restart, see *Checkpoint/Restart*.

**JCL Statements no Longer Supported or Supported Differently:** Parameters introduced in OS but not supported in MVS/System Product are:

- Main storage hierarchy support and rollout/rollin. The system will check the HIERARCHY and ROLL parameters only for correct syntax.
- The SEP and AFF parameters and the UNIT=SEP subparameter on the DD statement. The system will check them only for correct syntax. The job will fail if they are coded incorrectly.

JCL DD parameters supported differently are:

- SPLIT and SUBALLOC. Their values are converted internally to SPACE requests. When the SUBALLOC keyword is coded, the DD statement from which space is allocated becomes a dummy DD.
- If JES3 is used, the UNIT parameter on a DD statement that names a cataloged data set cannot specify a device type that conflicts with the cataloged device type. For example, a 3330 and a 3375.

## Prerequisite Publication

*Introduction to Virtual Storage in System/370*, GR20-4260.

## Publications Cited in the Text

### General

*Vocabulary for Data Processing, Telecommunications, and Office Systems*, GC20-1699.

### Base Control Program

*OS/VS2 MVS System Programming Library: Job Management*, GC28-1303.  
*OS/VS2 System Programming Library: Supervisor*, GC28-1046.  
*OS/VS2 MVS Supervisor Services and Macro Instructions*, GC28-1114.  
*Operator's Library: OS/VS2 MVS System Commands*, GC28-1031.  
*OS/VS2 MVS System Programming Library: Initialization and Tuning Guide*, GC28-1029.  
*MVS/370 System Generation Reference*, GC26-4063.  
*MVS Diagnostic Techniques*, SY28-1133.  
*OS/VS2 System Programming Library: Debugging Handbook*, Volumes 1 through 3, GC28-1047 through GC28-1049.  
*MVS/370 Message Library: System Messages*, Volume 1, GC28-1374, and Volume 2, GC28-1375.  
*OS/VS Message Library: VS2 System Codes*, GC38-1008.

### Data Facility Product

*MVS/370 Data Management Services*, GC26-4058.  
*MVS/370 System Programming Library: Data Management*, GC26-4056.  
*MVS/370 Data Management Macro Instructions*, GC26-4057.  
*MVS/370 Catalog Users Guide*, GC26-4053.  
*MVS/370 Checkpoint/Restart*, GC26-4054.  
*MVS/370 Magnetic Tape Labels and File Structure*, GC26-4064.  
*MVS/370 Access Method Services Reference for the Integrated Catalog Facility*, GC26-4051.  
*MVS/370 Access Method Services Reference for VSAM Catalogs*, GC26-4059.  
*MVS/370 VSAM Reference*, GC26-4074.  
*MVS/370 VSAM Users Guide*, GC26-4066.  
*OS/VS2 MVS System Programming Library: VTAM*, GC28-0688.  
*OS/VS2 TCAM Programmer's Guide, (levels 8 and 9)*, GC30-2041.  
*OS/VS TCAM System Programmer's Guide, (level 10)*, GC30-2051.  
*OS/VS BTAM*, GC27-6980.

### JES2

*System Programming Library: JES2 Initialization and Tuning*, SC23-0046.  
*Operator's Library: JES2 Commands*, SC23-0048.

## **JES3**

*JES3 System Programming Library: Initialization and Tuning*, SC23-0041.  
*JES3 Commands*, SC23-0045.  
*JES3 Messages*, GC23-0044.  
*JES3 System Programming Library: Diagnosis*, LC28-1369.  
*JES3 System Programming Library: User Modifications and Macros*, LC28-1371.

## **Programs**

*OS/VS2 MVS Interactive Problem Control System (IPCS) System Information*, GC34-2004.  
*OS/VS Mass Storage System (MSS) Services General Information*, GC35-0016.  
*OS/VS Mass Storage System (MSS) Services Reference Information*, GC35-0017.  
*Resource Access Control Facility (RACF) General Information Manual*, GC28-0722.  
*OS/VS2 MVS System Programming Library: Service Aids*, GC28-0674.  
*OS/VS2 MVS System Programming Library: System Management Facilities (SMF)*, GC28-1030.  
*OS/VS2 TSO Command Language Reference*, GC28-0646.  
*MVS/370 Utilities*, GC26-4065.

## **Hardware**

*Print Management Facility User's Guide and Reference*, SH35-0059.  
*IBM 3800 Printing Subsystem Models 3 and 8 Programmer's Guide*, SH35-0061.  
*OS/VS Graphic Programming Services (GPS) for IBM 2260 Display Station*, GC27-6972.  
*2821 Control Unit Component Description*, GA24-3312.  
*3340 Disk/Storage - Fixed Head Feature User's Guide*, GA26-1632.  
*OS and OS/VS Programming Support for the IBM 3505 Card Reader and IBM 3525 Card Punch*, GC21-5097.  
*OS/VS2 IBM 3540 Programmer's Reference*, GC24-5111.  
*3800 Printing Subsystem Programmer's Guide*, GC26-3846.  
*Forms Design Reference Guide for the IBM 3800 Printing Subsystem*, GA26-1633.



# Contents

## Part 1. Introduction

### Chapter 1. Introduction to Job Control 1-1

- The JCL Statements 1-1
- The JES2 Control Statements 1-3
- The JES3 Control Statements 1-5
- Cataloged and In-Stream Procedures 1-7
- Submitting and Executing Your Job 1-7

### Chapter 2. Coding Conventions for JCL, JES2, and JES3 Statements 2-1

- Notation Used to Show Syntax 2-1
- Fields in Control Statements 2-3
- The Parameter Field 2-5
  - Continuing Control Statements 2-6
- Coding Conventions for JES2 Control Statements 2-8
- Coding Conventions for JES3 Control Statements 2-8
- Concatenating Data Sets 2-9
- Character Sets 2-10
  - Coding Special Characters 2-10
- Backward References 2-13
- Symbolic Parameters 2-15
  - Defining Symbolic Parameters When Writing a Procedure 2-15
  - Assigning Values to and Nullifying Symbolic Parameters 2-16
  - Example of an In-Stream Procedure Containing Symbolic Parameters 2-21

## Part 2. Guide to Job and Step Control

### Chapter 3. Guide to Job Control 3-1

- Naming the Job 3-1
- Installation Management Information 3-2
  - Job Accounting Information Parameter 3-2
  - JES2 Accounting Information 3-4
  - Network Accounting Information 3-4
  - Programmer Information: The programmer-name parameter 3-5
- Networking 3-6
  - Routing a Job in a Network (JES2) 3-7
  - Transmitting Data in a Network (JES2) 3-7
  - Controlling Output Destination in a JES2 Network 3-7
  - Example of Obtaining Output (JES2) 3-9
  - Routing a Job in a Network (JES3 Networking) 3-10
  - Example of Routing a Job Through a JES3 Network 3-11
  - Controlling Job Execution Node Using JES3 Networking 3-11

Controlling Sysout Routing in a JES3 Network	3-11
Controlling Output Destination Using JES3	3-12
Remote Job Processing in JES3	3-12
Example of Obtaining Output (JES3)	3-13
Job Log	3-14
MSGLEVEL Parameter	3-14
MSGCLASS Parameter	3-16
JES2 Hard-Copy Log	3-16
JES3 Main Device Scheduler Messages	3-17
JES3 System Messages	3-18
TSO	3-18
NOTIFY Parameter	3-18
The JES2 NOTIFY Control Statement	3-19
The USER Parameter on the JES3 MAIN Control Statement	3-19
Remote Job Processing	3-20
JES2 Remote Job Processing	3-20
JES3 Remote Job Processing	3-22
Special Job Processing	3-23
Deadline Scheduling for JES3	3-27
Dependent Job Control for JES3: The Job Net	3-27
The JES3 NET Control Statement	3-28
How to Code NET Statements	3-29
JES3 Spool Partitioning	3-33
<b>Chapter 4. Guide to Step Control</b>	4-1
Naming a Job Step	4-3
Processing Program Information	4-4
Selecting a Processing Program	4-4
Identifying the Program to be Executed	4-4
The IEFBR14 Program	4-7
Selecting a Cataloged Procedure Library	4-7
Passing Information to the Program in Execution	4-8
PARM Parameter	4-9
Installation Management Information: The ACCT Parameter	4-10
ACCT Parameter	4-11
Dynamically Allocating and Deallocating Data Sets	4-12
Example of Dynamically Deallocating Data Sets	4-13
<b>Chapter 5. Guide to Job and Step Control</b>	5-1
Scheduling a Job	5-1
Selecting a Processor in JES2	5-3
Selecting a Processor in JES3	5-4
Conditionally Executing Job Steps	5-5
Specifying Return Code Tests on the JOB Statement	5-5
Specifying Return Code Tests on the EXEC Statement	5-7
Limiting Job and Job Step Execution Time	5-16
Using the TIME Parameter for Cataloged Procedures	5-17
Examples of Coding the Time Parameter on JOB and EXEC Statements	5-18
Controlling Job Queuing through Job Classes and Priorities	5-18
Establishing job processing balance in JES3	5-19
Assigning a Job to a Job Class in JES2	5-19
Assigning a Job to a Job Class in JES3	5-19
Assigning a Priority to a Job for JES2	5-20
Assigning a Priority to a Job in JES3	5-20

Assigning a Dispatching Priority to Job Steps	5-21
DPRTY Parameter	5-21
Performance of Jobs and Job Steps in JES2	5-22
Performance of Jobs and Job Steps in JES3	5-22
Requesting Storage for Execution	5-23
When to Request Real Storage	5-23
Specifying Storage Requirements with the REGION Parameter	5-24
Using the JES3 LREGION Parameter to Define Logical Storage	5-25
Restarting a Job at a Step or Checkpoint	5-26
The RD Parameter on the JOB Statement	5-27
The RESTART Parameter on the JOB Statement	5-28
The RD Parameter on the EXEC Statement	5-28
The JES2 RESTART Parameter	5-29
The JES3 FAILURE Parameter	5-29

### **Chapter 6. Guide to Data Allocation Control** 6-1

Using JES3 Spool Partitioning	6-1
Controlling Access to RACF-Protected Data Sets	6-2
Dynamically Allocating and Deallocating Data Sets	6-3
Example of Dynamically Deallocating Data Sets	6-4
Allocating Data Resources in a JES3 System	6-4

## **Part 3. Guide to Data Set Control**

### **Chapter 7. Guide to Specifying Data Set Information** 7-1

Specifying the DDNAME Parameter	7-1
When You Code the DDNAME Parameter	7-1
Specifying the DSNNAME Parameter	7-2
Creating or Retrieving a Nontemporary Data Set	7-3
Creating or Retrieving a Temporary Data Set	7-4
Associated Data Sets (3540 Diskette)	7-6
Copying the Data Set Name from an Earlier DD Statement	7-6
Specifying the DSNNAME Parameter in Apostrophes	7-7
Specifying the LABEL Parameter	7-7
Example of Identifying Data Sets to the System	7-13
Disposition Processing of Non-VSAM Data Sets	7-13
Specifying Data Set Status	7-14
Specifying a Disposition for the Data Set	7-14
Default Disposition Processing	7-19
Bypassing Disposition Processing	7-19
Insuring Data Set Integrity	7-19
Examples of Disposition Processing of Non-VSAM Data Sets	7-23
Requesting Units and Volumes	7-24
Specifying Volume Information	7-24
Specifying Unit Information	7-28
Example of Requesting Units and Volumes	7-31
Example of UNIT and VOLUME Affinities	7-35
Specifying Data Sets for Mass Storage Systems (MSS)	7-37
Mass Storage Volume Groups	7-37
Nonspecific Volume Requests for Mass Storage Volumes	7-38
Specific Volume Requests for Mass Storage Volumes	7-38
Requesting Space for Non-VSAM Data Sets	7-39
The Basic Space Request: Unit of Measurement and Primary Quantity	7-40
Assigning Specific Tracks	7-42

Example of Requesting Space	7-43
Specifying Data Set Processing Options	7-43
Processing Output Data Sets for the JOB	7-44
Processing System Output Data Sets Using the OUTPUT JCL Statement	7-44
Using the OUTPUT JCL Statement to Tailor the Job Stream	7-46
Specifying a Destination for the Data Set	7-49
Grouping Data Sets Using the OUTPUT JCL Statement	7-49
Managing the System-Managed Data Sets: The JESDS Parameter	7-49
Assigning System Output Data Sets to Output Classes	7-51
Specifying the Device	7-52
Specifying the Internal Reader	7-52
Example of Using the Internal Reader	7-53
JES Output Class Processing	7-54
Delaying the Writing of an Output Data Set	7-55
Suppressing the Writing of an Output Data Set	7-55
Limiting Output Records	7-55
Specifying JES2 Page Overflow Processing	7-56
Specifying JES3 Forms Overflow Processing and Printer Spacing	7-56
Interpretation of Punched Output	7-57
JES2 Support of the 3211 Indexing Feature	7-57
Requesting Multiple Copies of an Output Data Set Using JES2	7-57
Requesting Multiple Copies of an Output Data Set Using JES3	7-58
Requesting Copy Modification	7-58
Requesting Printer Form and Character Control	7-58
Requesting Forms Overlay	7-62
Bursting of Output	7-62
<b>Chapter 8. Guide to Special Data Sets</b>	<b>8-1</b>
Creating and Using Private and Temporary Libraries	8-1
Creating a Private Library	8-2
Retrieving an Existing Private Library	8-3
Using Private Catalogs	8-5
Temporary Libraries	8-5
Requesting an Abnormal Termination Dump	8-6
Defining a Dummy Data Set	8-8
Coding the DUMMY Parameter	8-8
Coding DSNAME = NULLFILE	8-9
Requests to Read or Write a Dummy Data Set	8-9
Using Virtual Input/Output (VIO) for Temporary Data Sets	8-10
Defining a VIO Temporary Data Set	8-10
Backward References to VIO Data Sets	8-11
Using Virtual Input/Output (VIO) to Pass Temporary Data Sets Among Job Steps	8-12
Entering Data Through the Input Stream	8-13
VSAM Data Sets	8-14
Creating and Retrieving Indexed Sequential Data Sets	8-18
Creating an Indexed Sequential Data Set	8-18
Retrieving an Indexed Sequential Data Set	8-22
Examples of Creating and Retrieving an Indexed Sequential Data Set	8-24
Creating and Retrieving Generation Data Sets	8-25
Building a Generation Data Group Base Entry	8-25
Creating a Generation Data Set	8-26
Retrieving a Generation Data Set	8-28
Deleting and Uncataloging Generation Data Sets	8-30
Submitting a Job for Restart	8-30



Example of Creating and Retrieving Generation Data Sets	8-31
Creating and Using a Subsystem Data Set	8-32

<b>Chapter 9. Guide to Cataloged and In-Stream Procedures</b>	9-1
Writing Cataloged and In-Stream Procedures	9-1
Identifying an In-Stream Procedure	9-2
Placing a Cataloged Procedure in a Procedure Library	9-2
Allowing for Changes in Cataloged and In-Stream Procedures	9-3
Using Cataloged and In-Stream Procedures	9-3
How to Call Cataloged and In-Stream Procedures	9-3
Modifying Cataloged and In-Stream Procedures	9-4
Adding DD Statements to a Procedure	9-8
Adding OUTPUT JCL Statements to a Procedure	9-10
Identifying Procedure Statements on an Output Listing	9-13

## Part 4. Reference to Job Control Statements and Parameters

<b>Chapter 10. Coding the JOB Statement</b>	10-1
Name Field	10-1
Parameter Field	10-1
Comments Field	10-2
Location in the JCL	10-2
Examples of JOB Statements	10-2
Accounting Information Parameter	10-3
Subparameter Definition	10-4
JES2 Accounting Information Format	10-4
Examples of the Accounting Information Parameter	10-5
ADDRSPC Parameter	10-7
Subparameter Definition	10-7
Defaults	10-7
Overrides	10-7
Relationship to the JOB REGION Parameter	10-7
Examples of the ADDRSPC Parameter	10-8
CLASS Parameter	10-9
Subparameter Definition	10-9
Defaults	10-9
Overrides	10-9
Example of the CLASS Parameter	10-9
COND Parameter	10-10
Subparameter Definition	10-10
Overrides	10-11
Summary of COND Parameters	10-11
Examples of the COND Parameter	10-11
GROUP Parameter	10-12
Subparameter Definition	10-12
Defaults	10-12
Relationship to Other Parameters	10-13
Example of the GROUP Parameter	10-13
MSGCLASS Parameter	10-14
Subparameter Definition	10-14
Defaults	10-14
Significance of Output Classes	10-14
Examples of the MSGCLASS Parameter	10-15
MSGLEVEL Parameter	10-16

Subparameter Definition	10-16
Defaults	10-17
Examples of the MSGLEVEL Parameter	10-17
NOTIFY Parameter	10-18
Subparameter Definition	10-18
Relationship to JES2 /*JOBPARM SYSAFF Parameter	10-18
Receiving Notification of Job Completion	10-18
Example of the NOTIFY Parameter	10-19
PASSWORD Parameter	10-20
Subparameter Definition	10-21
Relationship to Other Parameters	10-21
Examples of the PASSWORD Parameter	10-21
PERFORM Parameter	10-22
Subparameter Definition	10-22
Defaults	10-22
Overrides	10-22
Example of the PERFORM Parameter	10-23
Programmer's Name Parameter	10-24
Parameter Definition	10-24
Examples of the Programmer's Name Parameter	10-25
PRTY Parameter	10-26
Subparameter Definition	10-26
Defaults in a JES3 System	10-26
Relationship to Other Control Statements in a JES2 System	10-26
Example of the PRTY Parameter	10-27
RD Parameter	10-28
Subparameter Definition	10-29
Defaults	10-30
Overrides	10-30
Relationship to Other Control Statements	10-30
Examples of the RD Parameter	10-30
REGION Parameter	10-31
Subparameter Definition	10-31
Defaults	10-31
Overrides	10-32
Relationship to the JOB ADDRSPC Parameter	10-32
Examples of the REGION Parameter	10-32
RESTART Parameter	10-33
Subparameter Definition	10-34
Relationship to Other Control Statements	10-34
Cautions When Coding the RESTART Parameter	10-34
Generation Data Sets in Restarted Jobs	10-35
Examples of the RESTART Parameter	10-35
TIME Parameter	10-36
Subparameter Definition	10-36
Overrides	10-37
Defaults	10-37
Time Handling	10-37
Examples of the TIME Parameter	10-37
Examples of the TIME Parameter on JOB and EXEC Statements	10-38
TYPRUN Parameter	10-39
Subparameter Definition	10-39
Example of the TYPRUN Parameter	10-40
USER Parameter	10-41

Subparameter Definition	10-41
Defaults	10-41
Relationship to Other Parameters	10-42
Example of the USER Parameter	10-42
<b>Chapter 11. Coding the EXEC Statement</b>	<b>11-1</b>
Name Field	11-1
Parameter Field	11-2
Comments Field	11-3
Location in the JCL	11-3
Examples of EXEC Statements	11-3
<b>ACCT Parameter</b>	<b>11-4</b>
Subparameter Definition	11-5
On EXEC Statement that Calls a Procedure	11-5
Examples of the ACCT Parameter	11-5
<b>ADDRSPC Parameter</b>	<b>11-6</b>
Subparameter Definition	11-6
Defaults	11-6
Overrides	11-6
Relationship to the JOB or EXEC REGION Parameter	11-6
On EXEC Statement that Calls a Procedure	11-7
Examples of the ADDRSPC Parameter	11-7
<b>COND Parameter</b>	<b>11-8</b>
Subparameter Definition	11-9
Overrides	11-10
On EXEC Statement that Calls a Procedure	11-10
Cautions when Specifying COND Parameters	11-10
Summary of COND Parameters	11-11
Examples of the COND Parameter	11-12
<b>DPRTY Parameter</b>	<b>11-13</b>
Subparameter Definition	11-13
Defaults	11-14
On EXEC Statement that Calls a Procedure	11-14
Examples of the DPRTY Parameter	11-14
<b>DYNAMNBR Parameter</b>	<b>11-15</b>
Subparameter Definition	11-15
Defaults	11-15
On EXEC Statement that Calls a Procedure	11-15
Example of the DYNAMNBR Parameter	11-16
<b>PARM Parameter</b>	<b>11-17</b>
Subparameter Definition	11-17
On EXEC Statement that Calls a Procedure	11-18
Examples of the PARM Parameter	11-18
<b>PERFORM Parameter</b>	<b>11-19</b>
Subparameter Definition	11-19
Defaults	11-19
Overrides	11-19
On EXEC Statement that Calls a Procedure	11-20
Example of the PERFORM Parameter	11-20
<b>PGM Parameter</b>	<b>11-21</b>
Subparameter Definition	11-21
Checking JCL Syntax without Executing the Step	11-22
Examples of the PGM Parameter	11-22
<b>PROC and Procedure Name Parameters</b>	<b>11-24</b>

Subparameter Definition	11-24
Effect of PROC Parameter on Other Parameters and Following Statements	11-24
Examples of the PROC Parameter	11-25
RD Parameter	11-26
Subparameter Definition	11-27
Defaults	11-28
Overrides	11-28
Relationship to Other Control Statements	11-28
On EXEC Statement that Calls a Procedure	11-28
Examples of the RD Parameter	11-29
REGION Parameter	11-30
Subparameter Definition	11-30
Defaults	11-31
Overrides	11-31
Relationship to the EXEC ADDRSPC Parameter	11-31
Examples of the REGION Parameter	11-31
TIME Parameter	11-32
Subparameter Definition	11-32
Defaults	11-33
Overrides	11-33
Time Handling	11-33
Examples of the TIME Parameter	11-33
<b>Chapter 12. Coding the DD Statement</b>	<b>12-1</b>
Name Field	12-1
Parameter Field	12-3
Comments Field	12-3
Location in the JCL	12-3
DD Statements for Cataloged and In-stream Procedures	12-3
Examples of DD Statements and ddnames	12-5
* Parameter	12-6
Defaults	12-6
Relationship to Other Parameters	12-6
Relationship to Other Control Statements	12-7
Location in the JCL	12-7
Unread Records	12-7
Examples of the * Parameter	12-7
ACCODE Parameter	12-9
Subparameter definition	12-9
Defaults	12-9
Overrides	12-10
Example of the ACCODE Parameter	12-10
AMP Parameter	12-11
Subparameter Definition	12-12
Relationship to Other Parameters	12-14
Buffer Requirements	12-15
Examples of the AMP Parameter	12-15
BURST Parameter	12-16
Subparameter Definition	12-16
Default	12-16
Overrides	12-16
Relationship to Other Parameters	12-17
Relationship to Other Control Statements	12-17
Example of the BURST Parameter	12-17

CHARS Parameter	12-18
Subparameter Definition	12-18
Defaults	12-19
Overrides	12-19
Relationship to Other Parameters	12-19
Relationship to Other Control Statements	12-20
Printing Device Reassignment	12-20
Requesting a High-Density Dump in a JES3 System	12-20
Examples of the CHARS Parameter	12-20
CHKPT Parameter	12-21
Subparameter Definition	12-21
Overrides	12-21
Relationship to Other Parameters	12-21
Relationship to the SYSCKEOV DD Statement	12-21
Checkpointing Concatenated Data Sets	12-22
Examples of the CHKPT Parameter	12-22
CNTL Parameter	12-23
Subparameter Definition	12-23
Examples of the CNTL Parameter	12-23
COPIES Parameter	12-25
Subparameter Definition	12-25
Defaults	12-26
Overrides	12-26
Relationship to Other Parameters	12-26
Relationship to Other Control Statements	12-27
Examples of the COPIES Parameter	12-27
DATA Parameter	12-28
Defaults	12-28
Relationship to Other Parameters	12-28
Relationship to Other Control Statements	12-29
Location in the JCL	12-29
Unread Records	12-29
Examples of the DATA Parameter	12-29
DCB Parameter	12-31
Subparameter Definition	12-32
Defaults	12-32
Relationship to Other Parameters	12-33
Completing the Data Control Block	12-33
Examples of the DCB Parameter	12-35
DDNAME Parameter	12-49
Subparameter Definition	12-49
Relationship to Other Parameters	12-49
Overrides	12-49
Location in the JCL	12-50
Referenced DD Statement	12-50
Backward References	12-51
Examples of the DDNAME Parameter	12-51
DEST Parameter	12-53
Subparameter Definition for JES2 Systems	12-53
Subparameter Definition for JES3 Systems	12-54
Defaults	12-55
Overrides	12-55
Relationship to Other Parameters	12-55
Relationship to Other Control Statements	12-56

Examples of the DEST Parameter	12-56
DISP Parameter	12-57
Subparameter Definition	12-57
Defaults	12-60
Relationship to Other Parameters	12-60
Disposition of VSAM Data Sets	12-61
Disposition of Temporary Data Sets	12-61
Disposition of Partitioned Data Sets	12-61
DISP=MOD for a Multivolume Data Set	12-61
Examples of the DISP Parameter	12-62
DLM Parameter	12-64
Subparameter Definition	12-64
Default	12-65
Relationship to Other Parameters	12-65
Invalid Delimiters	12-65
Example of the DLM Parameter	12-65
DSID Parameter	12-66
Subparameter Definition	12-66
Relationship to Other Parameters	12-67
Example of the DSID Parameter	12-67
DSNAME Parameter	12-68
Subparameter Definition	12-69
Relationship to Other Parameters	12-71
Examples of the DSNAME Parameter	12-71
The DUMMY Parameter	12-73
Parameters on DD DUMMY Statements	12-73
Relationship to Other Parameters	12-74
Relationship to Other Control Statements	12-74
Relationship to Access Methods	12-74
Examples of the DUMMY Parameter	12-74
DYNAM Parameter	12-76
Relationship to Other Parameters	12-76
Relationship to Other Control Statements	12-76
Example of the DYNAM Parameter	12-76
FCB Parameter	12-77
Subparameter Definition	12-77
Defaults	12-78
Overrides	12-78
Relationship to Other Parameters	12-78
Relationship to Other Control Statements	12-79
Defining an FCB Image for a Work Station	12-79
Requesting a High-Density Dump in a JES3 System	12-79
Examples of the FCB Parameter	12-79
FLASH Parameter	12-81
Subparameter Definition	12-81
Defaults	12-81
Overrides	12-82
Relationship to Other Parameters	12-82
Relationship to Other Control Statements	12-82
Verification of Forms Overlay Frame	12-82
Printing without Flashing	12-83
Example of the FLASH Parameter	12-83
FREE Parameter	12-84
Subparameter Definition	12-84

Defaults	12-84
Overrides	12-84
Relationship to Other Parameters	12-84
Relationship to Other Control Statements	12-85
Relationship to the CLOSE Macro Instruction	12-85
Examples of the FREE Parameter	12-85
HOLD Parameter	12-87
Subparameter Definition	12-87
Defaults	12-87
Overrides	12-87
Relationship to Other Parameters	12-88
Relationship to Other Control Statements	12-88
Example of the HOLD Parameter	12-88
LABEL Parameter	12-89
Subparameter Definition	12-89
Defaults	12-93
Relationship to Other Parameters	12-93
Deleting a Data Set Before its Expiration Date	12-93
Translation	12-94
Examples of the LABEL Parameter	12-94
MODIFY Parameter	12-95
Subparameter Definition.	12-95
Defaults	12-96
Overrides	12-96
Relationship to Other Parameters	12-96
Relationship to other Control Statements	12-96
Example of the MODIFY Parameter	12-97
MSVGP Parameter	12-98
Subparameter Definition	12-98
Relationship to Other Parameters	12-99
Allocation when MSVGP is Not Coded	12-99
Examples of the MSVGP Parameter	12-100
OUTLIM Parameter	12-101
Subparameter Definition	12-101
Default	12-101
Relationship to Other Parameters	12-101
Relationship to Other Control Statements	12-102
Example of the OUTLIM Parameter	12-102
OUTPUT Parameter	12-103
Subparameter Definition	12-104
Defaults	12-104
Overrides	12-104
Relationship to Other Subparameters	12-105
Location in the JCL	12-105
No Match for OUTPUT Name	12-105
Processing Options in Multiple References	12-105
Examples of the OUTPUT Parameter	12-105
PROTECT Parameter	12-109
Subparameter Definition	12-109
Relationship to Other Parameters	12-109
Requirements for Protecting a Tape Volume	12-109
Requirements for Protecting a Direct Access Data Set	12-110
Examples of the PROTECT Parameter	12-110
QNAME Parameter	12-111

Subparameter Definition	12-111
Relationship to Other Parameters	12-111
Example of the QNAME Parameter	12-111
SPACE Parameter	12-112
Subparameter Definition	12-113
Relationship to Other Parameters	12-115
SPACE for New Data Sets on Mass Storage Volumes	12-116
Examples of the SPACE Parameter	12-116
SUBSYS Parameter	12-117
Subparameter Definition	12-117
Relationship to Other Parameters	12-118
Examples of the SUBSYS Parameter	12-118
SYSOUT Parameter	12-120
Subparameter Definition	12-121
Defaults	12-121
Overrides	12-121
Relationship to Other Parameters	12-122
Relationship to Other Control Statements	12-122
Starting an External Writer when Requested	12-122
Backward References	12-122
Held Classes in a JES2 System	12-123
Significance of Output Classes	12-123
Examples of the SYSOUT Parameter	12-123
TERM Parameter	12-125
Subparameter Definition	12-125
Relationship to Other Parameters	12-125
Location in the JCL	12-125
Examples of the TERM Parameter	12-126
UCS Parameter	12-127
Subparameter Definition	12-127
Defaults	12-128
Overrides	12-128
Relationship to Other Parameters	12-129
Using Special Character Sets	12-129
Examples of the UCS Parameter	12-129
UNIT Parameter	12-130
Subparameter Definition	12-130
Overrides	12-133
Relationship to Other Parameters	12-133
Location in the JCL	12-133
Examples of the UNIT Parameter	12-134
VOLUME Parameter	12-135
Subparameter Definition	12-136
Overrides	12-139
Relationship to Other Parameters	12-139
VOLUME Information for a Checkpoint/Restart Data Set	12-140
VOLUME Parameter in a JES3 System	12-140
VOLUME Parameter for Optical Readers	12-140
Examples of the VOLUME Parameter	12-140
<b>Chapter 13. Coding Special DD Statements</b>	<b>13-1</b>
JOB CAT DD Statement	13-2
Parameters on JOB CAT DD Statements	13-2
Relationship to STEPCAT DD Statement	13-2



Relationship to Other Control Statements	13-2
Location in the JCL	13-2
Example of the JOBCAT DD Statement	13-3
JOBLIB DD Statement	13-4
Parameters on JOBLIB DD Statements	13-4
Relationship to Other Control Statements	13-5
Location in the JCL	13-5
Relationship of a JOBLIB to a STEPLIB	13-5
Examples of the JOBLIB DD Statement	13-6
STEPCAT DD Statement	13-7
Parameters on STEPCAT DD Statements	13-7
Relationship to Other Control Statements	13-7
Location in the JCL	13-7
Example of the STEPCAT DD Statement	13-7
STEPLIB DD Statement	13-8
Parameters on STEPLIB DD Statements	13-8
Relationship to Other Control Statements	13-9
Location in the JCL	13-9
Relationship of a STEPLIB to a JOBLIB	13-9
Examples of the STEPLIB DD Statement	13-10
SYSABEND, SYSMDUMP, and SYSUDUMP DD Statements	13-11
Location in the JCL	13-11
Storing a Dump	13-11
Printing a Dump	13-12
Overriding Dump DD Statements	13-13
Duplicate Dump Requests	13-13
Examples of the SYSABEND, SYSMDUMP, and SYSUDUMP DD Statements	13-13
SYSCHK DD Statement	13-15
Parameters on SYSCHK DD Statements	13-15
Relationship to Other Control Statements	13-17
Location in the JCL	13-17
Examples of the SYSCHK DD Statement	13-17
SYSCKEOV DD Statement	13-18
Parameters on SYSCKEOV DD Statements	13-18
Coding SYSCKEOV for VSAM Data Sets	13-19
Example of the SYSCKEOV DD Statement	13-19
<b>Chapter 14. Coding the OUTPUT JCL Statement</b>	<b>14-1</b>
Name Field	14-2
Parameter Field	14-2
Comments Field	14-3
Location in the JCL	14-3
Overrides	14-4
BURST Parameter	14-6
Subparameter Definition	14-6
Defaults	14-6
Overrides	14-6
Example of the BURST Parameter	14-6
CHARS Parameter	14-7
Subparameter Definition	14-7
Defaults	14-8
Overrides	14-8
Requesting a High-Density Dump in a JES3 System	14-9
Example of the CHARS Parameter	14-9

CKPTLINE Parameter	14-10
Subparameter Definition	14-10
Defaults	14-10
Example of the CKPTLINE Parameter	14-10
CKPTPAGE Parameter	14-11
Subparameter Definition	14-11
Defaults	14-11
Relationship to Other Parameters	14-11
Example of the CKPTPAGE Parameter	14-11
CKPTSEC Parameter	14-12
Subparameter Definition	14-12
Defaults	14-12
Relationship to Other Parameters	14-12
Example of the CKPTSEC Parameter	14-12
CLASS Parameter	14-13
Subparameter Definition	14-13
Overrides	14-13
Held Classes	14-13
Significance of Output Classes	14-14
Examples of the CLASS Parameter	14-14
COMPACT Parameter	14-15
Subparameter Definition	14-15
Defaults	14-15
Overrides	14-15
Example of the COMPACT Parameter	14-15
CONTROL Parameter	14-16
Subparameter Definition	14-16
Defaults	14-16
Example of the CONTROL Parameter	14-16
COPIES Parameter	14-17
Subparameter Definition	14-17
Defaults	14-18
Overrides	14-18
Relationship to Other Parameters	14-18
Relationship to Other Control Statements	14-18
Examples of the COPIES Parameter	14-18
DEFAULT Parameter	14-20
Subparameter Definition	14-20
Defaults	14-20
Location in the JCL	14-20
References to Default OUTPUT JCL Statements	14-21
Example of the DEFAULT Parameter	14-21
DEST Parameter	14-23
Subparameter Definition for JES2 Systems	14-23
Subparameter Definition for JES3 Systems	14-24
Defaults	14-25
Overrides	14-25
Examples of the DEST Parameter	14-25
FCB Parameter	14-26
Subparameter Definition	14-26
Defaults	14-27
Overrides	14-27
Relationship to Other Parameters	14-27
Requesting a High-Density Dump in a JES3 System	14-27

Example of the FCB Parameter	14-27
FLASH Parameter	14-28
Subparameter Definition	14-28
Defaults	14-29
Overrides	14-29
Relationship to Other Parameters	14-29
Verification of Forms Overlay Frame	14-29
Printing without Flashing	14-29
Example of the FLASH Parameter	14-30
FORMDEF Parameter	14-31
Subparameter Definition	14-31
Overrides	14-31
Example of the FORMDEF Parameter	14-32
FORMS Parameter	14-33
Subparameter Definition	14-33
Defaults	14-33
Overrides	14-33
Example of the FORMS Parameter	14-33
GROUPID Parameter	14-34
Subparameter Definition	14-34
Examples of the GROUPID Parameter	14-34
INDEX Parameter	14-36
Subparameter Definition	14-36
Defaults	14-36
Relationship to Other Parameters	14-36
Example of the INDEX Parameter	14-36
JESDS Parameter	14-37
Subparameter Definition	14-37
Overrides	14-38
Location in the JCL	14-38
Destination for the System Data Sets	14-38
Example of the JESDS Parameter	14-38
LINDEX Parameter	14-39
Subparameter Definition	14-39
Defaults	14-39
Relationship to Other Parameters	14-39
Example of the LINDEX Parameter	14-39
LINECT Parameter	14-40
Subparameter Definition	14-40
Defaults	14-40
Example of the LINECT Parameter	14-40
MODIFY Parameter	14-41
Subparameter Definition	14-41
Defaults	14-42
Overrides	14-42
Relationship to Other Parameters	14-42
Example of the MODIFY Parameter	14-42
PAGEDEF Parameter	14-43
Subparameter Definition	14-43
Overrides	14-44
Example of the PAGEDEF Parameter	14-44
PIMSG Parameter	14-45
Subparameter Definition	14-45
Defaults	14-45

Example of the PIMSG Parameter	14-45
PRMODE Parameter	14-46
Subparameter Definition	14-46
Defaults	14-46
Printing a Line-Mode Data Set Using PSF	14-47
Example of the PRMODE Parameter	14-47
PRTY Parameter	14-48
Subparameter Definition	14-48
Defaults	14-48
Overrides	14-48
Example of the PRTY Parameter	14-48
THRESHLD Parameter	14-49
Subparameter Definition	14-49
Defaults	14-49
Example of the THRESHLD Parameter	14-50
TRC Parameter	14-51
Subparameter Definition	14-51
Defaults	14-51
Relationship to Other Parameters	14-51
Example of the TRC Parameter	14-52
UCS Parameter	14-53
Subparameter Definition	14-53
Defaults	14-54
Overrides	14-54
Using Special Characters Sets	14-55
Example of the UCS Parameter	14-55
WRITER Parameter	14-56
Subparameter Definition	14-56
Defaults	14-56
Overrides	14-56
Starting an External Writer	14-56
Example of the WRITER Parameter	14-57
<b>Chapter 15. Coding Special JCL Statements</b>	<b>15-1</b>
JCL Command Statement	15-2
Command	15-2
Parameter Field	15-3
Comments Field	15-3
Location in the JCL	15-3
Example of the Command Statement	15-3
Comment Statement	15-4
Location in the JCL	15-4
Listing of Comments Statements	15-4
Example of the Comment Statement	15-4
CNTL Statement	15-5
Label Field	15-5
Parameter Field	15-5
Comments Field	15-5
Location in the JCL	15-6
Program Control Statements	15-6
Program Control Statements in Procedures	15-6
Example of the CNTL Statement	15-6
Delimiter Statement	15-7
Relationship to the DD Statement DLM Parameter	15-7

Example of the Delimiter Statement	15-7
ENDCNTL Statement	15-8
Label Field	15-8
Comments Field	15-8
Location in the JCL	15-8
Example of the ENDCNTL Statement	15-8
Null Statement	15-9
Location in the JCL	15-9
Example of the Null Statement	15-9
PEND Statement	15-10
Name Field	15-10
Comments Field	15-10
Location in the JCL	15-10
Examples of the PEND Statement	15-10
PROC Statement	15-11
Name Field	15-11
Parameter Field	15-11
Comments Field	15-12
Overrides	15-12
Using Symbolic Parameters	15-12
Examples of the PROC Statement	15-12
<b>Chapter 16. Coding JES2 Control Statements</b>	<b>16-1</b>
Location in the JCL	16-1
Internal Reader	16-1
Command Statement	16-2
Parameter Definition	16-2
Location in the JCL	16-3
Examples of the Command Statement	16-3
/*JOBPARM Statement	16-4
Parameter Definition	16-4
Overrides	16-7
Location in the JCL	16-7
Execution Node	16-7
Example of the /*JOBPARM Statement	16-8
/*MESSAGE Statement	16-9
Relationship to the /*ROUTE XEQ Statement	16-9
Location in the JCL	16-9
Example of the /*MESSAGE Statement	16-9
/*NETACCT Statement	16-10
Parameter Definition	16-10
Defaults	16-10
Overrides	16-10
Location in the JCL	16-10
Example of the /*NETACCT Statement	16-10
/*NOTIFY Statement	16-11
Parameter Definition	16-11
Overrides	16-11
Relationship to Other Control Statements	16-12
Examples of the NOTIFY Statement	16-12
/*OUTPUT Statement	16-13
Parameter Definition	16-14
Overrides	16-20
Relationship to Other Control Statements	16-21

Location in the JCL	16-21
Example of the /*OUTPUT Statement	16-21
/*PRIORITY Statement	16-22
Parameter Definition	16-22
Overrides	16-22
Relationship to Other Control Statements	16-22
Location in the JCL	16-23
Example of the PRIORITY Statement	16-23
/*ROUTE Statement	16-24
Parameter Definition	16-24
Location in the JCL	16-26
Processing of /*ROUTE Statements	16-26
Multiple /*ROUTE Statements	16-26
Examples of the ROUTE Statement	16-26
/*SETUP Statement	16-28
Parameter Definition	16-28
Location in the JCL	16-28
Example of the /*SETUP Statement	16-28
/*SIGNOFF Statement	16-29
Example of the /*SIGNOFF Statement	16-29
/*SIGNON Statement	16-30
Location in the JCL	16-30
Parameter Definition	16-30
Examples of the /*SIGNON Statement	16-31
/*XEQ Statement	16-32
Parameter Definition	16-32
Location in the JCL	16-32
Multiple /*XEQ Statements	16-32
Example of the XEQ Statement	16-32
/*XMIT Statement	16-33
Parameter Definition	16-33
Defaults	16-34
Location in the JCL	16-35
Example of the XMIT Statement	16-35
<b>Chapter 17. Coding JES3 Control Statements</b>	<b>17-1</b>
Location in the JCL	17-1
Internal Reader	17-1
Examples of JES3 Control Statements	17-2
Command Statement	17-3
Parameter Definition	17-3
Location in the JCL	17-4
Examples of the Command Statement	17-4
/*DATASET Statement	17-5
Parameter Definition	17-5
Examples of the /*DATASET Statement	17-6
/*ENDDATASET Statement	17-7
Location in the JCL	17-7
Example of the /*ENDDATASET Statement	17-7
/*ENDPROCESS Statement	17-8
Location in the JCL	17-8
Example of the /*ENDPROCESS Statement	17-8
/*FORMAT PR Statement	17-9
Parameter Definition	17-10

Relationship to Sysout DD and OUTPUT JCL Statements	17-17
Relationship to <code>/*PROCESS</code> Statement	17-17
Examples of the <code>/*FORMAT PR</code> Statement	17-17
<code>/*FORMAT PU</code> Statement	17-18
Parameter Definition	17-19
Relationship to Sysout DD and OUTPUT JCL Statements	17-22
Relationship to <code>/*PROCESS</code> Statement	17-22
Example of the <code>/*FORMAT PU</code> Statement	17-22
<code>/*MAIN</code> Statement	17-23
Parameter Definition	17-24
Location in the JCL	17-34
Example of the <code>/*MAIN</code> Statement	17-34
<code>/*NET</code> Statement	17-35
Parameter Definition	17-35
Examples of the <code>/*NET</code> Statement	17-39
<code>/*NETACCT</code> Statement	17-40
Parameter Definition	17-40
Defaults	17-41
Example of the <code>/*NETACCT</code> Statement	17-41
<code>/*OPERATOR</code> Statement	17-42
Example of the <code>/*OPERATOR</code> Statement	17-42
<code>/*PAUSE</code> Statement	17-43
Example of the <code>/*PAUSE</code> Statement	17-43
<code>/*PROCESS</code> Statement	17-44
Parameter Definition	17-44
Location in the JCL	17-45
Examples of the <code>/*PROCESS</code> Statement	17-46
<code>/*ROUTE XEQ</code> Statement	17-47
Parameter Definition	17-47
Location in the JCL	17-47
Example of the <code>/*ROUTE XEQ</code> Statement	17-48
<code>/*SIGNOFF</code> Statement	17-49
Example of the <code>/*SIGNOFF</code> Statement	17-49
<code>/*SIGNON</code> Statement	17-50
Parameter Definition	17-50
Example of the <code>/*SIGNON</code> Statement	17-51

## Part 5. Reference Tables

Chapter 18. Reference Tables	18-1
------------------------------	------

Index	X-1
-------	-----





# Figures

1-1.	Job Control Statements	1-1
1-2.	JES2 Control Statements	1-3
1-3.	JES3 Control Statements	1-5
1-4.	A Job in the Input Stream	1-9
1-5.	A Job with Several Job Steps	1-10
1-6.	Job Boundaries in the Input Stream	1-10
2-1.	JCL Control Statement Fields	2-5
2-2.	Character Sets	2-10
4-1.	Using the EXEC Statement	4-1
4-2.	Modifying a Cataloged Procedure	4-2
5-1.	Using the COND Parameter	5-11
5-2.	Using the COND Parameter within a Failing Step	5-13
6-1.	Types of JES3 Setup	6-7
7-1.	How Device Status Affects Eligibility for Allocation	7-24
7-2.	Unit and Volume Affinity	7-34
8-1.	DD parameters used with VSAM	8-15
8-2.	DD parameters you should avoid with VSAM	8-16
9-1.	Identification of Cataloged Procedure Statements on the Output Listing	9-13
9-2.	Identification of In-stream Procedure Statements on the Output Listing	9-14
10-1.	Continuation or Termination of the Job Based on COND Parameter	10-11
11-1.	Execution or Bypassing of Current Step Based on COND Parameter	11-11
11-2.	Effect of EVEN and ONLY Subparameters on Step Execution	11-11
12-1.	Special Character Sets for the 1403, 3203 Model 5, and 3211 Printers	12-128
14-1.	Using job- and step-level OUTPUT JCL statements	14-1
14-2.	Special Character Sets for the 1403, 3203 Model 5, and 3211 Printers	14-54
17-1.	Table of Allowable DSPs for PROCESS Statements	17-45
18-1.	DD Parameters for Creating a Data Set	18-2
18-2.	DD Parameters for Retrieving a Data Set	18-4
18-3.	DD Parameters for Extending a Data Set	18-5
18-4.	DD Parameters for Retrieving or Extending an Indexed Sequential Data Set	18-6
18-5.	Area Arrangement of Indexed Sequential Data Sets	18-7
18-6.	Table of Mutually Exclusive DD Parameters	18-8
18-7.	Disposition Processing Table	18-9
18-8.	Direct Access Capacities	18-10
18-9.	Track Capacities	18-11
18-10.	The JOB Statement	18-13
18-11.	The EXEC Statement	18-14
18-12.	The DD Statement	18-15
18-13.	The OUTPUT JCL Statement	18-18



# Contents Directory

<b>JOB Statement</b>	_____	<b>JOB</b>
<b>EXEC Statement</b>	_____	<b>EXEC</b>
<b>DD Statement</b>	_____	<b>DD</b>
<b>Special DD Statement</b>	_____	<b>Spec DD</b>
<b>OUTPUT Statement</b>	_____	<b>OUTPUT</b>
<b>Special JCL Statements</b>	_____	<b>Spec JCL</b>
<b>JES2 Statements</b>	_____	<b>JES2</b>
<b>JES3 Statements</b>	_____	<b>JES3</b>
<b>Reference Tables</b>	_____	<b>Tables</b>
<b>Index</b>	_____	<b>Index</b>

11/11/11

11/11/11

11/11/11

11/11/11

11/11/11

## Summary of Amendments

### Summary of Amendments for GC28-1300-2 as updated December 21, 1984

This revision supports MVS/System Product Version 1 Release 3.4 with the following changes:

- The OUTPUT JCL statement can now be used in JES3 systems.
- The DEST parameter on the DD statement and on the JES3 `//*FORMAT` statement; JES3 now defines the default origin as the submitting node.

The revision also includes maintenance and editorial changes.

### Summary of Amendments for GC28-1300-1 as updated October 12, 1984 by TNL GN28-1016

This technical newsletter contains information to support changes in the Resource Access Control Facility (RACF) requirements for the JOB statement USER, PASSWORD, and GROUP parameters and to support early authorization verification.

### Summary of Amendments for GC28-1300-1 as updated December 1983

This revision contains:

- Information to support MVS/System Product Version 1 Release 3.4.
- Maintenance updates, which reflect the changes made in response to comments from our readers.



## Part 1. Introduction

You write your program in a programming language such as FORTRAN, assembler, or COBOL. The operating system translates this programming language into machine language, so that the computer can execute the instructions and perform work.

The program you code has specific resource requirements: storage, data sets, devices, and volumes. To communicate these resource requirements to the operating system, you use a programming language called job control language (JCL) in the input stream.

In addition, the job entry subsystem in use at your installation provides certain resources for your job and your job's data sets. Using Job Entry Subsystem 2 (JES2) and/or Job Entry Subsystem 3 (JES3) control statements, you can specify processing requirements for:

- Your job
- All data sets for your job
- Specific data sets in your job

A collection of related programs you submit to the operating system is a **job**. A job is made up of one or more **job steps**; each step is the unit of work associated with one of the programs that make up the job.

To introduce you to job control, the first chapter in this part describes:

- The JCL statements.
- The JES2 control statements.
- The JES3 control statements.
- Cataloged and in-stream procedures, which are standard sets of JCL statements for jobs you run frequently.
- Submitting and executing your job.

The second chapter in this part explains the coding conventions used throughout the book to describe the JCL statements and JES2 and JES3 control statements.





# Chapter 1. Introduction to Job Control

## The JCL Statements

Job control language (JCL) consists of the statements summarized in Figure 1-1. Each statement is described in detail in later chapters.

Statement	Name	Purpose
// JOB	job	Marks the beginning of a job; assigns a name to the job.
// EXEC	execute	Marks the beginning of a job step; assigns a name to the step; identifies the program or the cataloged or in-stream procedure to be executed in this step.
// DD	data definition	Identifies and describes a data set.
// OUTPUT	output	Specifies the processing options that the job entry subsystem is to use for printing output data sets.
// CNTL	control	Marks the beginning of one or more program control statements.
// ENDCNTL	end control	Marks the end of one or more program control statements.
// PEND	procedure end	Marks the end of an in-stream procedure.
// PROC	procedure	Marks the beginning of an in-stream procedure and may mark the beginning of a cataloged procedure; assigns default values to parameters defined in the procedure.
// command	command	Enters a system operator command through the input stream. The command statement is used primarily by the operator.  <b>Note:</b> JES3 ignores the JCL command statement.
//* comment	comment	Contains comments. The comment statement is used primarily to document a program and its resource requirements.
/*	delimiter	Indicates the end of data placed in the input stream.  <b>Note:</b> Any two characters can be designated by the user to be the delimiter.
//	null	Marks the end of a job in a JES3 system.  <b>Note:</b> JES2 ignores the null statement.

**Figure 1-1. Job Control Statements**

In addition to using JCL statements to identify your job, job steps, and data sets, you use JCL statement parameters to request resources and services from the operating system. The operating system, together with your job entry subsystem, is responsible for managing all the

resources of the computing system. It performs many job processing services automatically, but you can influence the way your job is processed by the JCL parameters you code. For example, the job entry subsystem selects jobs for execution, but you can speed up or delay selection of your job by the parameters you code on the JOB statement in a JES2 system or on the *//\*MAIN* statement in a JES3 system. Also, you can ask for a specific volume on which to write a data set.

The following paragraphs describe some of the functions that are requested through the major JCL statements:

*JOB statement:* Parameters on the JOB statement can:

- Provide accounting information to the installation's accounting routines.
- Define execution characteristics.
- Specify conditions for early termination of the job.
- Request a specific class for system messages and JCL statements.
- Hold a job for later execution.
- Limit the time that the job can use the processor.

*EXEC statement:* Parameters on the EXEC statement can:

- Identify the program or cataloged or in-stream procedure that the system is to execute.
- Provide job step accounting information.
- Give conditions for bypassing or executing a job step.
- Limit the time that the job step can use the processor.
- Pass information to a processing program, such as the linkage editor.

*DD statement:* Parameters on the DD statement can provide the system with information:

- The name of the data set.
- The type of I/O device that holds the data set.
- The serial number of the volume on which it resides.
- Whether a data set is old, new, or temporary.
- What to do with the data set after processing is finished.
- The format of the records in the data set.
- The size of newly created data sets.
- The access method that will be used to create or refer to the data.

*OUTPUT statement:* Parameters on the OUTPUT statement provide the system with processing options for system output data sets. Parameters on this statement can:

- Specify processing options for output data sets.
- Route output to a specific destination.
- In JES2 systems, process output data sets as a group.

## The JES2 Control Statements

In a JES2 installation, you can use JES2 control statements in the input stream to control the input, output, and processing of a program. See Figure 1-2 for the purpose of the JES2 statements that you can use with JCL statements to define your job's resource and processing requirements to the system.

Statement	Purpose
/*\$command	Enters JES2 operator commands through the input stream.
/*JOBPARM	Specifies certain job-related parameters at input time.
/*MESSAGE	Sends messages to the operator via the operator console.
/*NETACCT	Specifies an account number for a network job.
/*NOTIFY	Specifies the destination of notification messages.
/*OUTPUT	Specifies characteristics and options of groups of SYSOUT data sets or of a specific SYSOUT data set.
/*PRIORITY	Assigns a job queue selection priority.
/*ROUTE	Specifies the default output destination.
/*SETUP	Indicates volumes needed to execute the job.
/*SIGNOFF	Ends a remote job stream processing session.
/*SIGNON	Begins a remote job stream processing session.
/*XEQ	Specifies the execution node for a job.
/*XMIT	Indicates a job or data stream to be transmitted to another JES2 node or eligible non-JES2 node.

**Figure 1-2. JES2 Control Statements**

Several of these statements are briefly discussed here.

*JES2 /\*JOBPARM Statement:* Parameters on the JES2 /\*JOBPARM statement can specify:

- The estimated number of cards to be produced as output from a job.
- The number of copies of printed output desired.
- The default print or punch forms for this job.
- The number of output lines on each page.
- The estimated total number of output lines from the job.
- The estimated total number of system output data set bytes from the job.
- Your office number.
- Any system affinity that may be required.
- The estimated job execution time.
- The printing of the job log.
- The name of the cataloged procedure library to be used to convert the JCL for the job.

*JES2 /\*NETACCT Statement:* The network account number on the JES2 /\*NETACCT statement lets you specify accounting information that can be accepted and interpreted by all nodes. Some nodes may use the number as is, while others can translate it to a local account number.

*JES2 /\*ROUTE Statement:* Parameters on the JES2 /\*ROUTE statement can:

- Route the execution of a job to any processor in the network.
- Route the printed or punched output to any local device, remote terminal, or node in the network.

*JES2 /\*XEQ Statement:* The JES2 /\*XEQ statement causes JES2 to send the job to any processor in the JES2 network for execution.

*JES2 /\*XMIT Statement:* The JES2 /\*XMIT statement causes JES2 to transmit a data stream or job to a specified JES2 node or eligible non-JES2 node without JES2 input service functions being performed on the data stream.

# The JES3 Control Statements

In a JES3 installation, you can use JES3 control statements in the input stream to control the input, output, and processing of a program. See Figure 1-3 for the purpose of the JES3 statements that you can use with JCL statements to define your job's resource and processing requirements to the system.

Statement	Purpose
<code>/**command</code>	Enters JES3 operator commands, except <code>*DUMP</code> and <code>*RETURN</code> , through the input stream.
<code>/*DATASET</code>	Begins each additional input data set in the input stream.
<code>/*ENDDATASET</code>	Ends the input data set that began with a <code>DATASET</code> statement.
<code>/*ENDPROCESS</code>	Ends a series of <code>PROCESS</code> statements.
<code>/*FORMAT</code>	Specifies special destination and format-related instructions for a specific <code>SYSOUT</code> or JES3-managed print or punch data set.
<code>/*MAIN</code>	Defines selected processing parameters for the current job.
<code>/*NET</code>	Identifies relationships between predecessor and successor jobs in a dependent job control net.
<code>/*NETACCT</code>	Specifies an account number for a network job.
<code>/*OPERATOR</code>	Sends messages to the operator.
<code>/**PAUSE</code>	Halts the input reader.
<code>/*PROCESS</code>	Identifies a nonstandard job.
<code>/*ROUTE</code>	Specifies the destination node in a network.
<code>/*SIGNOFF</code>	Ends a remote job stream processing session.
<code>/*SIGNON</code>	Begins a remote job stream processing session.

Figure 1-3. JES3 Control Statements

Several of these statements are briefly discussed here.

*JES3 /\*\*FORMAT Statement:* Parameters on the JES3 `/**FORMAT` statement differ according to the type of request you are making. For print and punch data sets, keyword parameters specify such options as:

- Output destination.
- Number of output copies.
- Types of output forms.

*JES3 /\*\*MAIN Statement:* Parameters on the JES3 `/**MAIN` statement specify such options as:

- The main processor name or type of system to be used for the job.
- The type of control program to be used.
- The estimated number of cards or lines of output.
- The job class for the job.
- The time that the job is due to be completed.

*JES3 /\*\*PROCESS Statement:* A job, which is identified by a JOB statement, is a series of related problem programs, each identified by an EXEC statement. A job is also a series of JES3 processing functions. Standard processing needs only the standard scheduler elements: converter/interpreter service, main service, output service, and purge service.

A standard job consists of related programs to be processed by MVS; a standard job requires no special processing. A nonstandard job requires one or more special processing functions in place of or in addition to standard processing. Specify a nonstandard job by following the JOB statement with a JES3 /\*\*PROCESS statement for each job processing function.

For example, it is not always necessary to have all of the standard processes in a job. You can submit a job for debugging only. Because JES3 is not to execute the debugging job, you can skip some of the standard processing.

*JES3 /\*\*ROUTE XEQ Statement:* Parameters on the JES3 ROUTE statement direct a job to another node in the network for execution.

## Cataloged and In-Stream Procedures

You often use the same set of JCL statements repeatedly with little or no change, for example, to compile, link-edit, and execute a program. To save time and prevent errors, you can prepare standard job step definitions and place, or catalog, them in a partitioned data set known as the procedure library. Such a set of JCL statements in the system procedure library, SYS1.PROCLIB, is called a cataloged procedure. A cataloged procedure consists of EXEC and DD statements.

*Note:* Do not place any JES2 or JES3 control statements within a cataloged procedure.

To retrieve a cataloged procedure, use a JOB statement and an EXEC statement. On the EXEC statement, specify the name of the procedure. Your job uses the JCL statements in the cataloged procedure as if the JCL statements appeared in the input stream. If necessary, you can modify the cataloged procedure by a process known as overriding.

Before putting a procedure into the procedure library, you should test it. For testing, create an in-stream procedure; an in-stream procedure is a set of JCL statements starting with a PROC statement and ending with a PEND statement. You call this procedure with an EXEC statement that is in the same job as the procedure. After testing the procedure, catalog the in-stream procedure and call it with an EXEC statement whenever you want to use it.

*Note:* The maximum number of in-stream procedures you can code within any job is 15.

Cataloged and in-stream procedures are not checked for correct syntax until an EXEC statement that calls the procedure is syntax checked. Therefore, to test a procedure, an EXEC statement must call it.

## Submitting and Executing Your Job

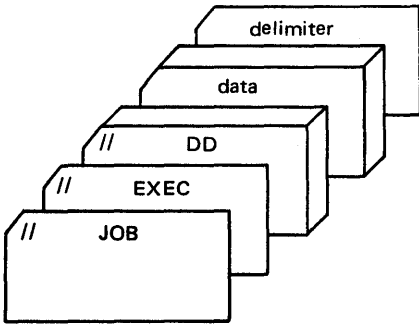
To have the computer execute your job, submit your JCL statements and any input data to the operating system through an input/output (I/O) device or an internal reader. The input device can be a card reader, a magnetic tape device, a terminal, or a direct access device. The input stream consists of the JCL statements and input data for all the jobs being submitted through an input device. The operating system distinguishes a job control statement from data in the input stream by the contents of the records.

A job can be simple or complex; you can have a procedure in the input stream or call a cataloged procedure. A job can consist of up to 255 job steps, including all steps in any procedure that the job calls. Specification of a greater number of steps produces unpredictable results.

See Figure 1-4 for some examples of jobs. One example shows the use of JES2 statements; these statements could have been placed within all of the jobs. In a system that uses JES3, the JES3 statements can be used in any of the jobs and are placed after the JOB statement.

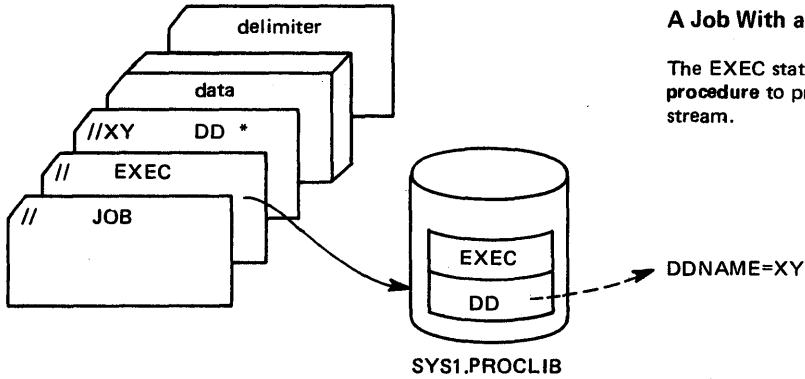
Figure 1-5 shows a job that contains several job steps: a compilation, a link-edit, and a program execution.

Figure 1-6 shows how several jobs run one after another through the input stream. Your job would be one job in the group of jobs that make up an input stream.



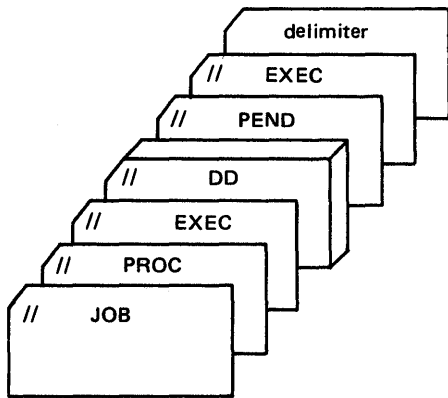
### A Job With One Job Step

The EXEC statement defines the program to be executed; the DD statements define the data to be used. There is also data in the input stream.



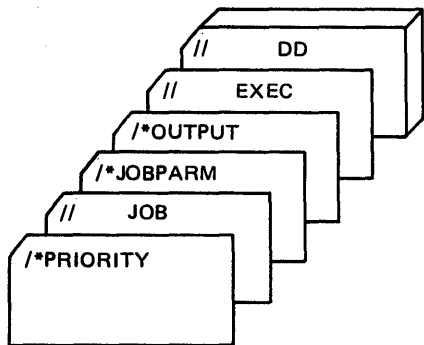
### A Job With a Cataloged Procedure

The EXEC statement is calling a cataloged procedure to process the data in the input stream.



### A Job With an In-stream Procedure

The EXEC statement refers to an in-stream procedure which is shown using the PROC and PEND statements.



### A Job With JES2 Statements

A simple job using JES2 control statements. The PRIORITY, command, and any comment statements would be the only control statements to be placed in front of the JOB statement.

Figure 1-4. A Job in the Input Stream



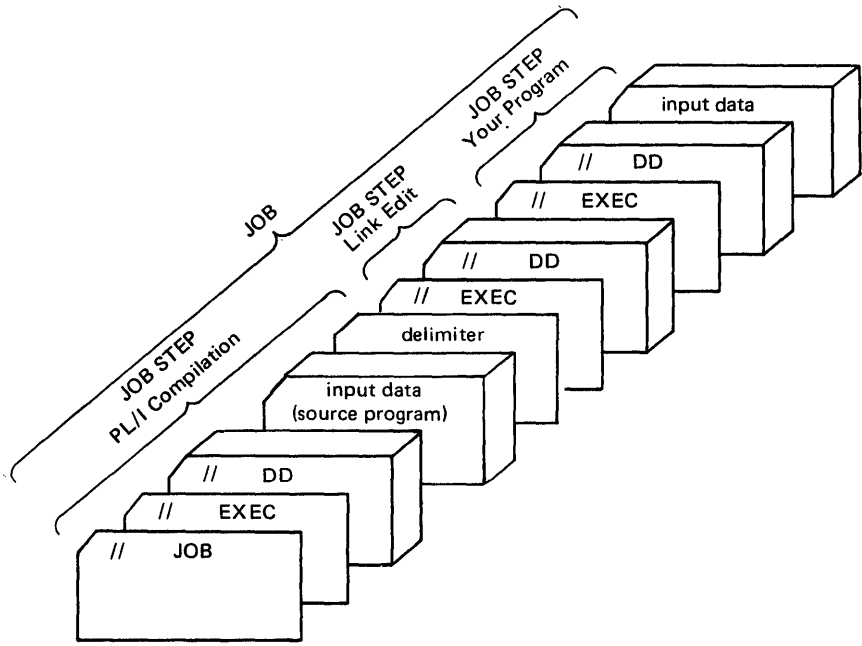


Figure 1-5. A Job with Several Job Steps

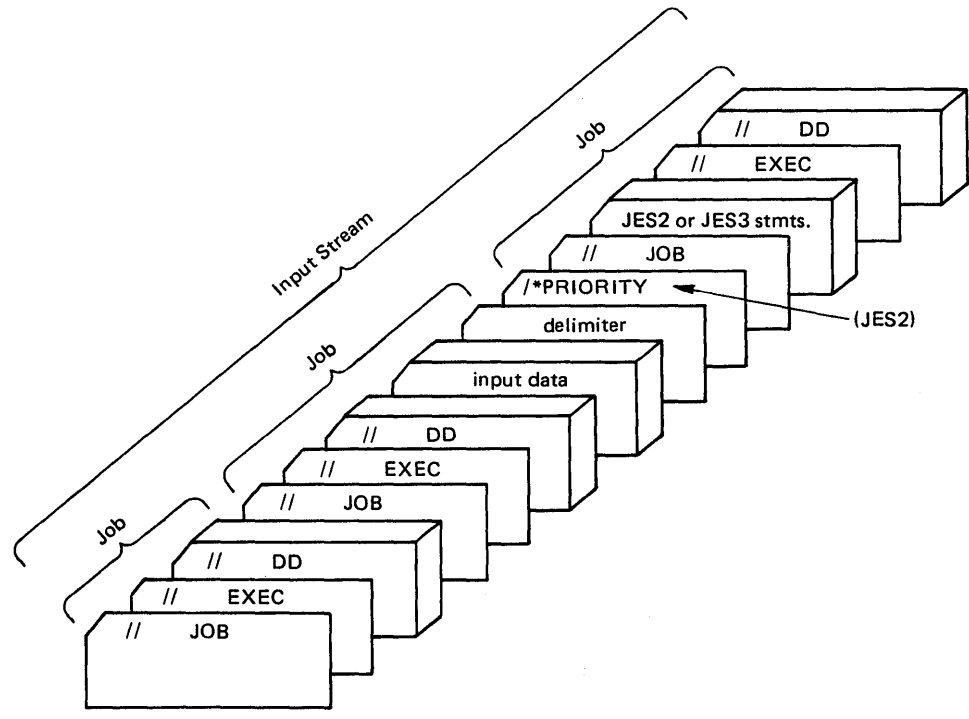


Figure 1-6. Job Boundaries in the Input Stream



## Chapter 2. Coding Conventions for JCL, JES2, and JES3 Statements

Syntax rules define how to code job control statements and their parameters. The syntax indicates:

- What the system requires.
- What is optional for the specific purpose or process you are requesting.
- How the statement and its parameters are to appear.

Some syntax rules are the same for JCL and JES parameters.

The following rules apply to all job control statements: JCL statements, JES2 control statements, and JES3 control statements. Additional coding conventions for JES2 and JES3 control statements are given under separate headings below.

You must follow the syntax rules in coding job control statements to achieve specific results. If you do not follow the rules, you may get error messages or unpredictable results. IBM does not support the use of statements or parameters to achieve results other than those stated in this publication.

### Notation Used to Show Syntax

The syntax of the job control statements and of their parameters appear in the chapters that describe the statements. The notation used in this publication for the syntax follows.

#### Uppercase letters, words, and characters

Code uppercase letters, words, and the characters listed below exactly as they appear in the syntax.

- & ampersand
- \* asterisk
- , comma
- = equal sign
- () parentheses
- . period

#### Lowercase letters, words, and symbols

Lowercase letters, words, and symbols in the syntax represent variables. When you code the parameter, you substitute specific information for them.

For example, CLASS=jobclass is the syntax for the CLASS parameter. When you code the CLASS parameter on a JOB statement, you substitute an alphanumeric character for the word "jobclass."

**| (vertical bar)**

A vertical bar indicates an exclusive OR. Never code it on a control statement. It is used in the syntax between choices within braces or brackets; it indicates that you code only one of the items within the braces or brackets.

For example, FREE={END|CLOSE} is the syntax for the FREE parameter. On a DD statement, you are to code either FREE=END or FREE=CLOSE.

**{ } (braces)**

Braces surround *required*, related items and indicate that you *must* code one of the enclosed items. Never code them on a control statement.

For example, the following is part of the syntax for the SPACE parameter on the DD statement.

```
{TRK          }
{CYL          }
{blocklength}
```

When coding the SPACE parameter, you must code TRK, CYL, or a numerical value substituted for "blocklength."

**[ ] (brackets)**

Brackets surround an *optional* item or items and indicate that you can code one or none of the enclosed items. Never code them on a control statement.

For example, [,DEFER] is part of the format description for the UNIT parameter. When you code the UNIT parameter, you can include ,DEFER in the UNIT parameter or omit it.

An example of several items in brackets appears in the LABEL parameter of the DD statement:

```
[ ,RETPD=nnnn ]
[ ,EXPDT=yyddd ]
```

You can code either ,EXPDT=yyddd or ,RETPD=nnnn in the LABEL parameter, or you can omit both.

Sometimes one of the items in brackets is a comma. Code the comma when you are not coding any of the other items in the brackets but you are coding a following part of the parameter. For example, the SYSOUT parameter of the DD statement appears in the format description as:

```
SYSOUT=( {class-name|,} [ ,writer-name|, ] [ ,form-name|,code-name ] )
```

You can code both "writer-name" and "form-name":

```
SYSOUT=(A,writer-name,form-name)
```

You can omit both:

```
SYSOUT=A
```

Or you can code only one:

```
SYSOUT=(A,writer-name) or SYSOUT=(A,,form-name)
```

Note in the second example that the comma after the vertical bar in the first set of brackets [,writer-name|,] must be coded when “,writer-name” is omitted and “,form-name” is included.

**– (underline)**

An underline indicates the default that the system uses when you do not code a subparameter. For example:

```
ADDRSPC={VIRT|REAL}
```

The underline indicates that VIRT is the default if you do not code the ADDRSPC parameter.

**... (ellipsis)**

An ellipsis follows an item that you can code more than once. Never code it on a control statement.

For example, COND=((code,operator)[,(code,operator)]...) is the syntax for the COND parameter on the JOB statement. The ellipsis indicates that you can repeat “,(code,operator).”

```
COND=( (12,GE) , (8,EQ) , (4,EQ) )
```

**.. (two consecutive periods)**

Two consecutive periods indicate that a parameter consists of a symbolic parameter followed by other information; only part of the field is variable. For example, &DEPT..MACS is such a parameter. If &DEPT = D58, then the actual value is D58.MACS.

## Fields in Control Statements

A job control statement consists of one or more 80-byte records. Each record is in the form of an 80-column punched-card image. Each job control statement is logically divided into the following five fields. All five fields do not appear on every control statement.

### Identifier field

The identifier field indicates to the system that a statement is a job control statement rather than data. The identifier field consists of the following:

- Columns 1 and 2 of all JCL statements, except the delimiter statement, contain //
- Columns 1 and 2 of the delimiter statement contain either /\* or two other characters designated by the user in a DLM parameter to be the delimiter
- Columns 1, 2, and 3 of a comment statement contain /\*

### **Name field**

The name field identifies a control statement so that other statements and the system can refer to it. For JCL control statements, code the name as follows:

- The name must begin in column 3.
- The name is 1 through 8 alphanumeric or national characters. See Figure 2-2 on page 2-10 for the character sets.
- The first character must be an alphabetic or national character.
- The name must be followed by at least one blank.

For JES control statements, code the name field as it appears in the control statement syntax.

### **Operation field**

The operation field specifies the type of control statement, or, for the command statement, the command. Code the operation field as follows:

- The operation follows the name field.
- The operation must be preceded and followed by at least one blank.

### **Parameter field**

The parameter field contains parameters separated by commas. Code the parameter field as follows:

- The parameter field follows the operation field.
- The parameter field must be preceded and followed by at least one blank.

See "The Parameter Field" on page 2-5 for details on coding the parameter field.

### **Comments field**

The comments field contains any information you deem helpful when you code the control statement. Code the comments field as follows:

- The comments field follows the parameter field.
- The comments field must be preceded by at least one blank.

You can code comments after the parameter field even though you continue the parameter field on a subsequent statement; see "Continuing Control Statements" on page 2-6.

For most statements, if you do not code any parameters, do not code any comments.

**Location of Fields on Statements:** Code the identifier field beginning in column 1 and the name field immediately after the identifier, with no intervening blanks. Code the operation, parameter, and comments fields in free form. Free form means that the fields need not begin in a particular column. Separate between fields with at least one blank; the blank serves as the delimiter between fields.

Do not code fields, except on the comment statement, past column 71. If the total length of the fields exceeds 71 columns, continue the fields onto one or more following statements.

Continuing fields is described under “Continuing Control Statements” on page 2-6. The comment statement can be coded through column 80.

**Use Keywords Only for Parameters or Subparameters:** Do not use parameter or subparameter keywords from any JCL, JES2, or JES3 statements as symbolic parameters, names, or labels.

Statement	Identifier	Fields
JOB	//	name JOB parameter <sup>1</sup> comments <sup>2</sup>
EXEC	//	name <sup>1</sup> EXEC parameter comments <sup>2</sup>
DD	//	name <sup>1</sup> DD parameter comments <sup>2</sup>
OUTPUT	//	name OUTPUT parameter comments <sup>2</sup>
PROC(cataloged)	//	name <sup>1</sup> PROC parameter comments <sup>2</sup>
PROC (in-stream)	//	name PROC parameter <sup>1</sup> comments <sup>2</sup>
PEND	//	name <sup>1</sup> PEND comments <sup>1</sup>
Command	//	command parameter <sup>1</sup> comments <sup>1</sup>
Delimiter	/* xx	comments <sup>1</sup> comments <sup>1</sup>
Null	//	
Comment	/*	comments

<sup>1</sup> Optional  
<sup>2</sup> Optional -- If parameters are not coded, comments cannot be coded.  
If parameters are coded, comments are optional.

Figure 2-1. JCL Control Statement Fields

## The Parameter Field

The parameter field is made up of two types of parameters: **positional parameters** and **keyword parameters**. Positional parameters must precede all keyword parameters. Keyword parameters follow the positional parameters.

**Commas:** You must use commas to separate all positional parameters, keyword parameters, and subparameters coded in the parameter field.

**Positional Parameters:** A positional parameter consists of (1) characters that appear in uppercase in the syntax and must be coded as shown, (2) variable information, or (3) a combination. For example, DATA on a DD statement, programmer’s-name on a JOB statement, and PGM = program-name on an EXEC statement.

Code positional parameters first in the parameter field in the order indicated in the syntax. If you omit a positional parameter and code a following positional parameter, code a comma to indicate the omitted parameter. Do not code the replacing comma if:

- The omitted positional parameter is the last positional parameter.
- All following positional parameters are also omitted.
- Only keyword parameters follow.
- All positional parameters are omitted.

**Keyword Parameters:** A keyword consists of characters that appear in uppercase in the syntax and must be coded as shown followed by an equals sign followed by either characters that must be coded as shown or variable information. For example, RD = R and MSGCLASS = class-name on the JOB statement.

Code any of the keyword parameters for a statement in any order in the parameter field after the positional parameters. Because of this positional independence, do not code a comma to indicate the absence of a keyword parameter.

**Multiple Subparameters:** A positional parameter or the variable information in a keyword parameter sometimes consists of more than one item, called a subparameter list. A subparameter list can consist of both positional and keyword subparameters. These subparameters follow the same rules as positional and keyword parameters.

When a parameter contains more than one subparameter, separate the subparameters by commas and enclose the subparameter list in parentheses or, if indicated in the syntax, by apostrophes. If the list is a single keyword subparameter or a single positional subparameter with no omitted preceding subparameters, omit the parentheses or apostrophes.

**Symbolic Parameters:** The EXEC and DD statements in cataloged and in-stream procedures can contain one other type of parameter: a *symbolic parameter*. A symbolic parameter consists of an ampersand (&) followed by a name. For example, DEST = &LOC on a procedure DD statement.

A symbolic parameter stands as a symbol for a parameter, a subparameter, or a value. Use symbolic parameters to make variable any information in the parameter field of a procedure EXEC statement or DD statement. You assign a value to a symbolic parameter by coding the value on the EXEC statement that calls the procedure. This value is in effect only for this execution of the procedure. For example:

```
//STEP1 EXEC PROC=A,LOC=NYC
```

For a detailed discussion of symbolic parameters, see "Symbolic Parameters" on page 2-15.

## Continuing Control Statements

When the total length of the fields on a control statement exceeds 71 columns, continue the fields onto one or more following statements.

JCL statements that you **cannot** continue follow. While you cannot continue these statements, you can code as many separate statements as you need.

- Command statement
- Comment statement
- Delimiter statement
- Null statement

For all other JCL statements, you can continue the parameter field or the comments field.



**Continuing the Parameter Field:** The continuation conventions for the parameter field are:

1. Interrupt the field after a complete parameter or subparameter, including the comma that follows it, at or before column 71.
2. Include comments by following the interrupted parameter field with at least one blank.
3. Code a nonblank character in column 72 when you are continuing a comments field and, optionally, when you are continuing the parameter field.

*Note:* The system treats a following statement as a continuation, even when column 72 is blank, when conventions 4, 5, and 6 are followed.

4. Code // in columns 1 and 2 of the following statement.
5. Continue the interrupted parameter or field beginning in any column from 4 through 16. If you begin coding after column 16, the system treats this statement as a comment field.
6. If column 3 contains a nonblank character other than an asterisk, the system assumes the following statement is a new statement. The system issues an error message indicating that no continuation is found and fails the job.

**Continuing the Comments Field:**

1. Interrupt the comment at a convenient place before column 72.
2. Code a nonblank character in column 72.
3. Code // in columns 1 and 2 of the following statement.
4. Continue the comments field beginning in any column after column 3.

### Identifying Comments on an Output Listing

The system lists in the job log the control statement and how it was interpreted.

//\* in columns 1 through 3: indicates a control statement in the input stream, other than a comment statement, that the system considers to contain only comments.

XX\* in columns 1 through 3: indicates a control statement in a cataloged procedure, other than a comment statement, that the system considers to contain only comments.

\*\*\* in columns 1 through 3: indicates a comment statement.

+ + in columns 1 and 2: indicates any control statements in an in-stream procedure.

For additional information, see "Identifying Procedure Statements on an Output Listing" on page 9-13.

## Coding Conventions for JES2 Control Statements

Code JES2 control statements with JCL statements to control the input and output processing of jobs. The rules for coding JCL statements, including syntax, discussed in the preceding topics, apply to the JES2 control statements. However, there are additional rules for coding JES2 statements. They are:

- Columns 1 and 2 always contain the characters /\*.
- The /\*OUTPUT statement is the only JES2 control statement that you can continue.

For all other JES2 control statements, code multiple control statements if you require more than one statement.

- If you code more than one of the same parameters on the same statement, JES2 uses the last parameter value coded.

When coding more than one of the same JES2 control statements, be aware of the following system actions:

- If you code more than one statement with the same parameter, JES2 uses the parameter value coded on the last statement.
- If you code more than one statement with different parameters, JES2 uses all parameters.

## Coding Conventions for JES3 Control Statements

You can code JES3 statements in combination with JCL statements to control the input and output processing of your job. Rules for coding JCL, including syntax, discussed in previous topics, apply to the JES3 statements. However, there are additional rules for coding JES3 statements. They are:

- Columns 1 through 3 usually contain the characters /\*. There are some JES3 control statements that have /\* in columns 1 and 2.
- Columns 3 and 4 must be nonblanks.
- Continue JES3 statements, except command statements, by:
  1. Coding a comma as the last character of the first statement.
  2. Coding /\* in columns 1 through 3 of the continuation statement.
  3. Resuming the text in column 4 of the continuation statement.
- Do not include comments on JES3 control statements, except ENDPROCESS and PAUSE statements.

## Concatenating Data Sets

You can logically connect (concatenate) up to 255 sequential or 16 partitioned input data sets for the duration of a job step. Each of these data sets may reside on a different volume.

To concatenate data sets, omit the ddnames from all the DD statements except the first in the sequence. When MVS encounters this ddname in a data control block in the processing program, MVS processes each data set in the same sequence as the DD statements defining them.

You may concatenate data sets on different devices as long as you do not concatenate data sets on RPS devices to data sets on non-RPS devices, or vice versa.

You may also concatenate data sets that have different block sizes as long as the data set with the largest block size appears first in the concatenation. For further details on concatenating data sets, refer to *Data Management Services Guide*.

### *Concatenation Cautions*

- If you make a *backward reference* to a concatenation (using \*.), the system obtains information only from the first data set defined in the sequence.
- If you make a *forward reference* to a concatenation (using the DDNAME parameter), the system only obtains information from the first data set defined in the sequence.
- If you issue a *RDJFCB macro* instruction to a DD statement that is concatenated, only the job file control block (JFCB) for the first data set is read.
- If you define a data set using the *DUMMY parameter* you should not concatenate other data sets to it. When the processing program asks to read a dummy data set, the system takes an end-of-data set exit immediately and ignores any data set that might be concatenated to the dummy.

# Character Sets

You can code job control statements using a combination of the characters from three different character sets. Figure 2-2 illustrates the contents of each of the character sets.

Character Set	Contents	
Alphanumeric	Alphabetic Numeric	A through Z 0 through 9
National (See Note)	“At” sign Dollar sign Pound sign	@ \$ #
Special	Comma Period Slash Apostrophe Left parenthesis Right parenthesis Asterisk Ampersand Plus sign Hyphen Equal sign Blank	, . / ' ( ) * & + - =
<p><b>Note:</b> The system recognizes the following hexadecimal representations of the U.S. National characters; @ as X'7C'; \$ as X'5B'; and # as X'7B'. In countries other than the U.S., the U.S. National characters represented on terminal keyboards might generate a different hexadecimal representation and cause an error. For example, in some countries the \$ character may generate a X'4A'.</p>		

**Figure 2-2. Character Sets**

When coding any special characters, you must follow certain rules. The description and use of these rules follows.

## Coding Special Characters

You use special characters in the job control language to:

- Delimit parameters (the comma).
- Delimit fields (the blank).
- Perform syntactical functions. (For example, the appearance of && as the first two characters following DSNAMES= tells the system that a temporary data set name follows.)

Sometimes you can code a special character that does not satisfy one of the above uses of special characters. In most of these cases, indicate that you are using special characters by enclosing the item that contains the special characters in apostrophes, for example,

ACCT='123+456'. If one of the special characters is an apostrophe, you must code two consecutive apostrophes in its place, for example, 'O'NEILL'.

The following list contains those parameters that can have special characters as part of their variable information, and indicates when you do not have to code the apostrophes. Where applicable, this information is repeated for each parameter in "Part 4. Reference to Job Control Statements and Parameters."

- The *accounting information* on the JOB statement. The account number and additional accounting information can contain hyphens without being enclosed in apostrophes. For example:

```
//JOB D JOB D58-D04
```

- The *programmer's name* on the JOB statement. The programmer's name can contain periods and/or hyphens without being enclosed in apostrophes.

```
//JOB N JOB ,P.F.M,CLASS=...
//JOB X JOB ,S.M-TU,CLASS=...
//JOB C JOB ,M-T,CLASS=...
```

However, because a comma cannot immediately follow a period, the following is invalid:

```
//JOB Y JOB ,LEIGH.,TYPRUN=...
```

- The *checkid field* in the RESTART parameter on the JOB statement can contain an asterisk. For example:

```
//JOB Z JOB A709I,NAT,RESTART=*
```

- The *ACCT parameter* on the EXEC statement. The ACCT parameter can contain hyphens and plus zero (+0, an overpunch) without being enclosed in apostrophes. For example:

```
//STEP1 EXEC PRINT,ACCT=D58-LOC
//STEP2 EXEC PGM=PUB,ACCT=D57+0
```

- The *PARM parameter* on the EXEC statement may contain an ampersand for symbolic parameters. When coding the ampersand for symbolic parameters, you need not code an apostrophe. For example:

```
//STEPX EXEC MYPROC,PARM=&UNIT
```

- The *DSNAME parameter* on the DD statement.

- You can code *hyphens* in the DSNAME parameter without enclosing it in apostrophes. For example:

```
// DD DSN=NAT-SMT,...
```

- You can code *periods* in a qualified data set name without enclosing it in apostrophes. The use of periods within the data set name qualifies the DSNAME and eliminates the need to enclose it in apostrophes. For example:

```
// DD DSN=BPT.DATA.GROUP
```

However, the following combinations are invalid when you do not enclose the parameters within apostrophes.

- A period immediately after a left parenthesis or immediately before a right parenthesis, for example:

```
// DD DSN=(.ABC)
// DD DSN=(ABC.)
```

- A period immediately followed by a comma, for example:

```
// DD DSN=P.D.S.,
```

- You can code a *plus or minus* (hyphen) sign to identify a generation of a generation data group in the DSNNAME parameter without enclosing it in apostrophes. For example:

```
// DD DSN=PAYROLL(+1)
```

- You can code *ampersands* as the first two characters when defining a temporary data set in the DSNNAME parameter without enclosing them in apostrophes.

```
// DD DSN=&&ELM
```

- You can code *parentheses* in the DSNNAME parameter when defining a member of a partitioned data set, a generation of a generation data group, or an area of an indexed sequential data set. The parentheses that enclose the member name, generation number, or area name do not have to be enclosed in apostrophes. For example:

```
// DD DSN=PDS(MEM1)
```

- The *volume serial number* of the VOLUME parameter can contain hyphens without being enclosed in apostrophes. For example:

```
// DD VOL=SER=PUBS-RD
```

- *Subsystem-defined parameters* in the SUBSYS DD parameter can contain special characters without being enclosed in apostrophes.

- The *device type* subparameter of the UNIT parameter on the DD statement can contain hyphens without being enclosed in apostrophes.

```
// DD UNIT=3330-1
```

## Backward References

Many parameters in job control statements allow you to make use of the backward reference to fill in information. The backward reference lets you copy previously coded information or refer to DD statements that appear earlier in your job. Most backward references are coded as *\*.stepname.ddname*, where *stepname* is the name of an earlier step that contains the DD statement to which *ddname* is referring. The step you name *must* contain the DD statement to which you are referring.

For example:

```
//REFBAK1 JOB      ....
//STEP1   EXEC     ....
//POKDD   DD       DSN=D58.POK.PUBS01
.
.
//STEP2   EXEC     ....
//        DD       DSN=*.STEP1.POKDD
```

If the DD statement appears earlier in the same step as the backward reference, code the backward reference as *\*.ddname*. The DD statement to which you are referring *must* precede the backward reference.

For example:

```
//REFBAK2 JOB      ....
//STEP    EXEC     ....
//PUBSDD  DD       DSN=D04.POK.PUBS04
.
.
//        DD       DSN=*.PUBSDD
```

Do **not** make backward references to a DD statement that contains a SYSOUT parameter.

You can also refer to a DD statement that is contained in a cataloged or in-stream procedure step by coding *\*.stepname.procstepname.ddname*.

You must:

- Code an asterisk (\*) indicating to the system that this is a backward reference.
- Code the name of the step in your job that invokes the procedure.
- Code the name of the procedure step that contains the DD statement to which you are referring.
- Code the name of the DD statement to which you are referring.

The cataloged procedure PUBS contains:

```
      .  
      .  
//SPELL EXEC ....  
      4  .  
      .  
//WRITE DD ....  
      5  .  
      .
```

Your job contains:

```
//REFBAK3 JOB ....  
//STEPCALL EXEC PROC=PUBS  
      3  . 1  
      .  
//WKSTEP EXEC ....  
      .  
      .  
// DD DSN=*.STEPCALL.SPELL.WRITE  
      2 3 4 5
```

1. Step STEPCALL invokes the cataloged procedure named PUBS.

Later in your job, you wish to make a backward reference to a DD statement in the cataloged procedure PUBS.

2. The asterisk (\*) indicates to the system that this is a backward reference.
3. STEPCALL is the name of the previous step in your job that invoked the cataloged procedure that contains the DD statement to which you want to refer.
4. SPELL is the name of the step within the cataloged procedure that contains the DD statement to which you want to refer.
5. WRITE is the name of the DD statement in the cataloged procedure to which you want to refer.



## Symbolic Parameters

In order to be modified easily, cataloged and in-stream procedures can contain symbolic parameters. A symbolic parameter is a symbol preceded by an ampersand that stands for a parameter, a subparameter, or a value. In the following procedure step, the symbolic parameters are underlined:

```
//STEP1 EXEC PGM=UPDATE , ACCT=(PGMG, &DEPT)  
//DD1 DD DSNAME=INIT, UNIT=&DEVICE, SPACE=(CYL, (&SPACE, 10))  
//DD2 DD DSNAME=CHNG, UNIT=3400-6, DCB=BLKSIZE=&LENGTH
```

When this procedure is executed, every symbolic parameter must either be assigned a value or nullified on the EXEC statement calling the procedure; the changes are in effect only for the current execution of the procedure. Therefore, without being permanently changed, the procedure can be modified each time it is executed. Details on how to assign values to or nullify symbolic parameters are included under "Assigning Values to and Nullifying Symbolic Parameters." How to include symbolic parameters when writing a cataloged or in-stream procedure is described in the next section, "Defining Symbolic Parameters When Writing a Procedure."

## Defining Symbolic Parameters When Writing a Procedure

Any parameter, subparameter, or value in a procedure that can vary each time the procedure is called is a good candidate for definition as a symbolic parameter. For example, if different values can be passed to a processing program by means of the PARM parameter on one of the EXEC statements, you could define the PARM field as one or more symbolic parameters, for example,

```
PARM=&ALLVALS  
or  
PARM=&DECK&CODE
```

The symbolic parameter itself is one to seven alphanumeric and national (#,@,\$) characters preceded by a single ampersand. The first character must be alphabetic or national. Since a single ampersand defines a symbolic parameter, you code double ampersands to indicate to the system that you are not defining a symbolic parameter. For example, if you want to pass 543&LEV to a processing program by means of the PARM parameter, you must code PARM='543&&LEV'. The system treats the double ampersand as if a single ampersand had been coded, and only one ampersand appears in the results.

If you enclose a symbolic parameter with apostrophes, a symbolic parameter not enclosed in apostrophes must precede the one enclosed in apostrophes for correct substitution to occur.

Keyword parameters that you may code on the EXEC statement (such as ACCT, COND, and PARM) cannot be used as the name of a symbolic parameter. Also, you cannot use EXEC statement keyword parameters to define symbolic parameters in JCL procedures that you intend to start from the console using the START command.

For example, you cannot code &REGION=200K or REGION=&REGION on the EXEC statement, but you can code REGION=&SIZE.

The definitions used to signify symbolic parameters should be consistent in all the cataloged and in-stream procedures at an installation. For example, every time the programmer is to assign his department number to a symbolic parameter, no matter which procedure he is

calling, the symbolic parameter could be defined as `&DEPT`. In different procedures, you could code `ACCT=(43877,&DEPT)` and `DSNAME=LIBRARY.&DEPT.TALLY`. The programmer would assign his department number to the symbolic parameter wherever that symbolic parameter appears in a procedure.

The same symbolic parameter can appear more than once in a procedure, as long as the value assigned to the symbolic parameter is a constant in the procedure. Therefore, you could use `&DEPT` more than once in a procedure, if the department number to be assigned is the same in each use. But if you have two `DD` statements and include a symbolic parameter for the primary quantity of the space request on each `DD` statement, you would not want to use the same symbolic parameter, since the requests for primary quantity could be different for the two data sets.

Only one value can be assigned to each symbolic parameter used in a procedure; if you assign more than one value to a symbolic parameter, only the first value is used and that value is substituted wherever the symbolic parameter occurs.

### **Assigning Default Values to Symbolic Parameters**

You can assign default values to the symbolic parameters coded in the procedure on the `PROC` statement. The `PROC` statement must always appear as the first statement in an in-stream procedure; the `PROC` statement must be coded as the first statement in a cataloged procedure only if you want to assign defaults. Generally, you should assign defaults to every symbolic parameter in a procedure to limit the amount of coding necessary each time the procedure is called. For details, see the next section, "Assigning Values to and Nullifying Symbolic Parameters."

You can use symbolic parameters on `DD` statements that you are adding to a procedure. However, if you are adding a `DD` statement to the last step of a procedure — do **not** use symbolic parameters that are not used elsewhere in the procedure.

### **Assigning Values to and Nullifying Symbolic Parameters**

When a procedure containing symbolic parameters is used, each symbolic parameter must either be assigned a value or nullified. If the symbolic parameter is not assigned a value or nullified, the symbolic parameter remains in the `JCL` for that job.

Symbolic parameters are assigned values or nullified in one of two ways:

1. The programmer who uses the procedure codes the symbolic parameter on the `EXEC` statement that calls the procedure, either assigning it a value or nullifying it. Symbolic parameters specified on the `EXEC` statement calling the procedure must appear in the procedure.
2. The programmer who writes the procedure assigns a default value to or nullifies the symbolic parameter on the `PROC` statement, which must be the first statement in an in-stream procedure and can be the first statement in a cataloged procedure.

The default assigned to a symbolic parameter on a `PROC` statement is overridden when that symbolic parameter is assigned a value or nullified on the `EXEC` statement that calls the procedure.

Default values are not necessarily assigned to symbolic parameters in a procedure. Before using any procedure, find out what symbolic parameters are used, the meaning of each symbolic parameter, and what default, if any, is assigned. The PROC statement is optional in cataloged procedures; if the PROC statement is not included, no default values can be assigned to symbolic parameters in the procedure.

You need not code the symbolic parameters in any specific order when you assign values to or nullify them.

### Assigning a Value to a Symbolic Parameter

To assign a value to symbolic parameter, you code:

```
symbolic parameter=value
```

Omit the ampersand that precedes the symbolic parameter in the procedure. For example, if the symbolic parameter &NUMBER appears on a DD statement in the procedure, code NUMBER=value on the PROC statement (if you are writing the procedure and assigning defaults) or on the EXEC statement that calls the procedure (if you are using the procedure and want this value to be in effect only for the current execution of the procedure).

There are some rules for assigning values to symbolic parameters:

- The length of the value assigned is limited as follows:
  - The value cannot be continued on to another statement.
  - The length of the value you assign, combined with the length of all following parameters and delimiters in the operand field of a single statement, cannot exceed 120 characters. For example:

```
//INIT EXEC PGM=MYPROG,PARM='0
```

The length of the value assigned to &X must be less than 103 characters because the length of the remainder of the operand field “,TIME=10,REGION=5” is 17 characters.

*Note:* The PARM keyword on the EXEC statement is given special consideration when processing symbolic parameters. Any symbolic parameter appearing in a PARM keyword operand that is enclosed in apostrophes is replaced by its assigned value. (&X in the example above is replaced.) For any other use of a symbolic parameter in an operand enclosed in apostrophes, the symbolic parameter is not replaced. For example,

```
//DDX DD DSN='&PRM',DISP=(,SHR)
```

&PRM is not replaced by its assigned value.

If the character string that includes the ampersand is not enclosed in apostrophes, the ampersand causes a JCL error.

- If the value contains special characters, enclose the value in apostrophes (the enclosing apostrophes are not considered part of the value). If the special characters include apostrophes, each apostrophe must be shown as two consecutive apostrophes.

- If more than one value is assigned to a symbolic parameter as a default on the PROC statement, only the first value encountered is used; likewise, if more than one value is assigned to a symbolic parameter on an EXEC statement, only the first value encountered is used.
- If a symbolic parameter is a positional parameter followed by other parameters in the statement, it should be followed in the procedure by a period instead of a comma; for example:

```
//DEFINE DD  &POSPARM.DSN=ATLAS,DISP=OLD
```

- Do not use a value of literal blanks, that is, VALUE=' ', to nullify a symbolic parameter.

Symbolic parameters that are keyword subparameters should appear in the procedure without a preceding comma; for example:

```
VOLUME=SER=(111111&SERNO)
```

This is necessary so that, if the symbolic parameter is nullified, a leading or trailing comma will not cause a JCL syntax error. (For a more complete discussion of this, see "Caution Concerning Leading and Trailing Commas.")

In these cases, you must include a comma when you assign a value to the symbolic parameter; that is:

```
POSPARM='DUMMY,'
SERNO=' ,222222'
```

Since the comma is a special character, the value must then be enclosed in apostrophes.

- Two consecutive periods (..) indicate that a parameter consists of a symbolic parameter followed by other information; only part of the field is variable. For example, &DEPT..MACS is such a parameter. If &DEPT = D58, then the actual value is D58.MACS.

### Nullifying a Symbolic Parameter

To nullify a symbolic parameter, code:

```
symbolic parameter=
```

Omit the ampersand that precedes the symbolic parameter in the procedure and do not follow the equal sign with a value.

For example, a DD statement in an in-stream procedure named TIMES is:

```
//DD8 DD UNIT=3211,UCS=&UCSINFO
```

If you are writing the procedure and want to nullify &UCSINFO as a default on the PROC statement, code:

```
//TIMES PROC UCSINFO=
```

If you are calling the procedure, and no default was assigned to &UCSINFO, or if &UCSINFO was assigned a value on the PROC statement, nullify the parameter on the EXEC statement that calls the procedure by coding:

```
//CALL EXEC TIMES,UCSINFO=
```

If a symbolic parameter appears as the last parameter on a statement that is being continued, it cannot be nullified and must be assigned a value. An attempt to nullify such a parameter results in a JCL error.

### Caution Concerning Leading and Trailing Commas

All symbolic parameters must be assigned values or nullified before the procedure is executed. (When you write a procedure, you can assign default values to the symbolic parameters, or the programmer can assign values when he calls the procedure; for details, see "Assigning Values to and Nullifying Symbolic Parameters.") When a symbolic parameter is nullified, a delimiter, such as a leading or trailing comma, is not automatically removed. Only when the symbolic parameter is a positional subparameter followed by other subparameters should the comma remain. In other cases, the remaining comma will cause a syntax error.

For example, you code for a unit request:

```
UNIT=(3350,&MORE,DEFER)
```

If &MORE is nullified, the comma before it must remain, since the unit count subparameter is positional and a comma must indicate its absence if other subparameters follow. When &MORE is nullified, the parameter will appear as:

```
UNIT=(3350,,DEFER)
```

However, if you code:

```
VOLUME=SER=(111111,&SERNO)
```

and &SERNO is nullified, a leading comma will remain and cause a JCL syntax error. If a symbolic parameter is a positional parameter followed by other parameters in the statement, such as

```
//DEFINE DD &POSPARM,DSN=ATLAS,DISP=OLD
```

the comma will remain at the beginning of the operand field if &POSPARM is nullified and again cause a syntax error.

In these cases, do not code the comma. When a symbolic parameter follows information that does not vary, such as in `VOLUME=SER=(111111,&SERNO)`, you do not have to code any delimiter. The system recognizes the symbolic parameter when it encounters the single ampersand. For this example, you would code:

```
VOLUME=SER=(111111&SERNO)
```

When a value is assigned to the symbolic parameter, a comma must be included in the value, that is `SERNO='22222'`. (Because the comma is a special character, you must enclose the entire value within single apostrophes.)

When a symbolic parameter precedes information that does not vary, a period may be required after the symbolic parameter to distinguish the end of the symbolic parameter from the

beginning of the information that does not vary. A period is required after the symbolic parameter when the character following the symbolic parameter is:

- An alphabetic, numeric, or national (#,@,\$) character
- A period

The system recognizes the period as a delimiter and the period does not appear in the procedure after the symbolic parameter is assigned a value or nullified. (A period will appear after the value when two consecutive periods are coded.)

Therefore, place a period after a symbolic parameter that stands for a positional parameter followed by other parameters in the statement:

```
//DEFINE DD &POSPARM.DSN=ATLAS,DISP=OLD
```

If &POSPARM is nullified, the statement appears as:

```
//DEFINE DD DSN=ATLAS,DISP=OLD
```

When assigning a value to &POSPARM, you must include a comma:

```
POSPARM=' DUMMY, '
```

These rules are in effect whenever you are concatenating a symbolic parameter with information that does not vary. In the following examples, a symbolic parameter is placed after information that does not vary.

In these examples, the system recognizes the symbolic parameter when it encounters the &:

- DSNAME = LIBRARY(&MEMBER)
- DSNAME = USERLIB.&LEVEL

In the following examples, a symbolic parameter is placed before information that does not vary:

- PARM = '&OPTION + 15'. &OPTION is not followed by period because of the +.
- DSNAME = &QUAL.246. The period is required because a numeric character follows the symbolic parameter.
- DSNAME = &DOCNO..TXT. The period is required because a period follows the symbolic parameter. A single period will appear in the results.

You can also define two or more symbolic parameters in succession without including a comma, for example, PARM = &DECK&CODE. If you want a comma in the results, you must include a comma in the value assigned to the symbolic parameter.

## Example of an In-Stream Procedure Containing Symbolic Parameters

The following example illustrates the execution of an in-stream procedure to test symbolic parameters prior to placing the procedure in a procedure library.

The in-stream procedure named TESTPROC contains the following statements:

```
//TESTPROC PROC A=IMB406,B=ABLE,C=3330,D=WXYZ1,  
//           E=OLD,F=TRK,G='10,10,1'  
//STEP      EXEC PGM=&A  
//DD1       DD   DSN=&B,UNIT=&C,VOL=SER=&D,DISP=&E,  
//           SPACE=(&F,(&G))  
//           PEND
```

To execute the above in-stream procedure with certain overrides (change DSN to BAKER, PGM to IEFBR14, DISP to (NEW, KEEP) and leave the remainder of the parameters the same), code the following statements:

```
//STEPX     EXEC TESTPROC,A=IEFBR14,B=BAKER,E='(NEW,KEEP)'
```

After the symbolic substitution, the statements will look like this:

```
//STEP      EXEC PGM=IEFBR14  
//DD1       DD   DSN=BAKER,UNIT=3330,VOL=SER=WXYZ1,  
//           DISP=(NEW,KEEP),SPACE=(TRK,(10,10,1))
```

To execute the above in-stream procedure and change DD1 to resemble a temporary scratch space, code the following statement:

```
//STEPX     EXEC TESTPROC,A=IEFBR14,B=,C=3350,D=,E=
```

After the symbolic substitution, the statements will look like this:

```
//STEP      EXEC PGM=IEFBR14  
//DD1       DD   DSN=,UNIT=3350,VOL=SER=,DISP=,SPACE=(TRK,(10,10,1))
```





## **Part 2. Guide to Job and Step Control**

The operating system provides some resources for all the programs in a job. It provides other resources for a particular job step, that is, a particular program in a job.

You code parameters on a JOB statement to ask the operating system for resources for your entire job. There are also some JES control statements that you can use to request resources for the entire job. The parameters and control statements for resources for the entire job are discussed in Chapter 3, "Guide to Job Control."

You code parameters on an EXEC statement to ask the operating system for resources for a particular job step. The parameters for resources for a particular step are discussed in Chapter 4, "Guide to Step Control."

Some parameters that you can code on a JOB statement you can also code on an EXEC statement. Because these parameters have a relationship to each other when you code them on both the JOB and EXEC statements, they are discussed together in Chapter 5, "Guide to Job and Step Control."



## Chapter 3. Guide to Job Control

Normally the JOB statement is the first JCL statement of your job. Certain JES control statements can precede the JOB statements; these are discussed with the JOB statement parameter to which they relate. The JOB statement must contain a valid jobname in its name field and the word JOB in its operation field. The format of the JOB statement is:

```
//jobname JOB positional-parameters[,keyword-parameter]... comments
```

If you do not code any parameters on the JOB statement, you cannot code comments on the JOB statement.

The JOB statement parameters and JES control statements allow you to specify information in the following areas:

- Installation management information.
- Networking.
- Operating system messages.
- TSO (time sharing option).
- Remote job processing.
- Special job processing.
- JES3 spool partitioning.
- RACF-protected data sets.

In this chapter, each area is discussed; the discussions center around the JCL statements that you can code to direct the system's handling of your job. Where a JES2 or JES3 control statement also affects an area, the appropriate JES control statement and/or parameter is also discussed.

### Naming the Job

Your job must have a name. The job scheduler component of the operating system requires a jobname.

Code the jobname in the name field of the JOB statement. It can range from one to eight characters in length and can contain any alphanumeric characters. However, the first character of the name must be an alphabetic or national character and must begin in column 3.

The following are examples of jobnames in JOB statements:

```
//MYJOB      JOB
//MCS167     JOB
//R#123      JOB
//@5AB       JOB
```

## Installation Management Information

You can specify two types of installation management information in the JOB statement: job accounting information and programmer's name. These are positional parameters: You must code them before coding any other parameters on a JOB statement. The first positional parameter is for job accounting information; the second positional parameter is for the programmer's name:

```
//jobname JOB account,programmer
```

If you omit the job accounting information, you must indicate its absence with a comma:

```
//jobname JOB ,programmer
```

If you omit both the job accounting information and the programmer's name and you specify other parameters (such as MSGLEVEL), you do not need to indicate their absence with commas:

```
//jobname JOB MSGLEVEL=...
```

The installation can make mandatory the job accounting information or programmer's name or both. That is, your job will fail if you do not specify them on your JOB statement. Your manager or supervisor should inform you of this requirement.

### Job Accounting Information Parameter

The first positional parameter of the JOB statement allows you to supply job accounting information. It has the following format:

( [account-number] [,accounting-information]... )

Replace the term "account-number" with the account number to which you want to charge the job; replace the term "accounting-information" with other items your installation's accounting routines require. Your manager or supervisor should tell you exactly how to code the job accounting information parameter. The following are general rules for coding accounting information.

- The account number and each item of accounting information are subparameters; you must separate them with a comma.
- You can enclose the job accounting information with either parentheses or apostrophes. For example:

```
//JOBOZ JOB (12A75,DEPTD58,920)
or
//JOB0X JOB '12A75,DEPTD58,920'
```

If you use apostrophes, the system considers all accounting information within the apostrophes as one field.

- If a subparameter contains special characters (except hyphens), you must code it as follows:

- Enclose that subparameter in apostrophes and all the job accounting information in parentheses. For example:

```
//JOBON JOB (12A75,'DEPT/D58',920)
```

- Enclose all the job accounting information in apostrophes. For example:

```
//JOBOK JOB '12A75,DEPT/D58,920'
```

- If the special character is an apostrophe, you must code two consecutive apostrophes. For example, code DEPT'D58 as:

```
//JOBGO JOB (12A75,'DEPT''D58',920)
```

or

```
//JOBDO JOB '12A75,DEPT''D58,920'
```

- If your job accounting information consists of only an account number and the number does not contain special characters, you need not enclose the number in parentheses or apostrophes. If the account number contains special characters, enclose it in apostrophes. For example:

```
//JOB01 JOB 12A75  
//JOB02 JOB '12A.75'  
//JOB03 JOB '12A''75'
```

- If your job accounting information does not include an account number, you must indicate its absence by coding a comma preceding the additional accounting information. For example:

```
//JOB04 JOB (,DEPTD58,920)  
//JOB05 JOB (,'DEPT/D58',920)  
//JOB06 JOB ',DEPT/D58,920'
```

- Job accounting information can consist of up to 142 characters, including the commas that separate the subparameters. If your installation's parameter is long, you will have to continue it onto another statement. Keep the following in mind when you continue it:

- Enclose the job accounting information in parentheses.

- Do not continue a subparameter enclosed in apostrophes. Break for the continuation before or after such a subparameter.

- Follow the continuation conventions outlined in "Continuing Control Statements" on page 2-6. The following example shows job accounting information continued onto another statement.

```
//YOURJOB JOB (2G29,'DEPT/D58',  
// 920)
```

## JES2 Accounting Information

JES2 systems assume that the accounting information parameter contains information that you can, alternatively, specify on the JES2 `/*JOBPARM` statement. Specify the accounting information in the following format:

```
(pano,room,time,lines,card,forms,copies,log,linect)
```

JES2 will interpret and use the subparameters of this accounting information as explained in "Accounting Information Parameter" on page 10-3.

However, your installation might initialize JES2 to ignore accounting field subparameters that are illegal by JES2 standards. If your installation has initialized JES2 to terminate a job that has an accounting field subparameter that is illegal by JES2 standards, then you must code the first two subparameters (pano and room).

For a discussion of the JES2 scan of the accounting information parameter, refer to *SPL: JES2 Initialization and Tuning*.

## Network Accounting Information

Both job entry subsystems allow you to transmit a job to some other node in a network. Therefore, both JES2 and JES3 provide a means of specifying an account number that is available to all nodes in a network.

### JES2 NETACCT Control Statement

The JES2 NETACCT statement is an account number that JES2 makes available to all nodes in the JES2 network. The format of the JES2 NETACCT statement is:

```
/*NETACCT network-account-number
```

The "network-account-number" is from 1 to 8 alphanumeric characters that represent this job's accounting throughout the JES2 network. The JES2 NETACCT statement follows the JOB statement.

```
//JOBAB JOB  
/*NETACCT NETNUM9  
:
```

For complete coding information on the JES2 NETACCT statement, see "/\*NETACCT Statement" on page 16-10.

## JES3 NETACCT Control Statement

The JES3 NETACCT statement allows you to specify a variety of accounting information that JES3 makes available to other nodes in the JES3 network. The format of the JES3 NETACCT statement is:

```
//*NETACCT parameter[,parameter]...
```

The parameters are:

```
    PNAME=programmer's-name
    ACCT=nnnnnnnn
    BLDG=nnnnnnnn
    DEPT=nnnnnnnn
    ROOM=nnnnnnnn
    USERID=nnnnnnnn
```

The JES3 NETACCT statement follows the JOB statement. For example:

```
//JOBAC  JOB  ....
//*NETACCT PNAME=BETH,ACCT=D57706
```

For complete coding information on the JES3 NETACCT statement, see “//\*NETACCT Statement” on page 17-40.

## Programmer Information: The programmer-name parameter

The second positional parameter on the JOB statement indicates your name or identification to your installation's accounting routines. This parameter must follow the job accounting information. The following are rules for coding the programmer's name:

- The programmer's name must not exceed 20 characters, including all special characters.
- Enclose the name in apostrophes if it contains:
  - Special characters, other than hyphens. Blanks are special characters.
  - Leading or imbedded periods, including a period at the end of the name.

For example:

```
//JOBA1  JOB  ,T.JONES
//JOBA2  JOB  ,'T JONES'
//JOBA3  JOB  ,'SP/4 T.JONES'
//JOBA6  JOB  ,'L.NORRIS.'
```

- If the special character is an apostrophe, you must code it as two consecutive apostrophes. For example, specify the names O'Neill and J.O'Brien as:

```
//JOBA4  JOB  ,'O''NEILL'
//JOBA5  JOB  ,'J.O''BRIEN'
```

- If you do not code the job accounting information, you must indicate its absence by a comma preceding the programmer's name. For example:

```
//MCS  JOB  ,LORRAYN
```

- If you do not intend to code the programmer's name parameter, you do not have to indicate its absence with a comma. For example:

```
//JOBA8 JOB A709P,CLASS=A
```

## Networking

**JES2** networking lets you:

- Enter a job on one system and execute it on another system.
- Send output from one system to a remote device or an output device on another system.
- From a JES2 node, send data to nodes to which the sending node does not have a direct connection via an intermediate node.

For further information, see *JES2 Initialization and Tuning*.

**JES3** networking lets you:

- From one JES3 node, send jobs to another node for execution.
- From a JES3 node, send output to other nodes for processing.
- From a JES3 node, send data to nodes to which the sending node does not have a direct connection via an intermediate node.

For further information, see *JES3 SPL: Initialization and Tuning*.

Ask the system programmers at your installation and at other node(s) for the specific information required to properly prepare jobs for execution or output processing at the node.

Factors you should consider are:

- *The content and format of the JOB statement:* The receiving node might have different parameter requirements than the submitting node.
- *The job entry subsystem in use at the receiving node:* The receiving node, if using a different job entry subsystem, will have different control statement requirements.
- *The content of the procedure library (PROCLIB):* The procedure library might not be the same at the receiving node and at the submitting node.
- *Data set identification:* To use data sets at the receiving node, you need to know their names and the data set values to code in your JCL parameters.
- *Installation specific unit names* in use at the receiving node might include devices different from the submitting node.
- *SYSOUT classes* might have different meanings at the receiving node than at the submitting node.



- *Execution classes* might have different meanings at the receiving node than at the submitting node.

## Routing a Job in a Network (JES2)

In a JES2 system, jobs can be routed in two ways:

- An operator can issue a JES2 command through a console or input reader. Refer to *JES2 Commands*.
- JES2 control statements can be specified within the JCL.

Execution routing through JCL may be accomplished with either of two control statements:

- The ROUTE statement with the XEQ parameter. Specifying the XEQ option with the name of a node sends the job to that node for execution. The output returns to the node of origin, unless you code the PRINT or PUNCH option on another ROUTE statement.
- The XEQ statement with the name of a node. This statement sends the job to that node for execution. The output returns to the node of origin, unless you code the PRINT or PUNCH option on a ROUTE statement.

The job may be entered at an RJE terminal, a TSO terminal, an input reader, or an internal reader before being routed to some other node for execution. However, neither a started task, such as a task begun by an operator START command, nor a TSO logon can be executed at any node other than the node of entry.

## Transmitting Data in a Network (JES2)

By using the JES2 XMIT control statement you can submit and route a non-JES2 job or data through a JES2 node to a non-JES2 node (for example, a VM node or JES3 node) for processing. You can also use the XMIT statement to send jobs and data from a JES2 node to another JES2 node. When you use this control statement, JES2 will not check the information between the XMIT statement and the delimiter for JES2 syntax.

## Controlling Output Destination in a JES2 Network

JES2 allows you to submit jobs to a central computing center from a work station anywhere in the network; you can route output to any node or work station in the network.

Unless overridden by the system programmer or operator, the default output location is the submitting location, either a remote work station or the central site (destination of LOCAL). To receive the output at the submitting location, simply assign output data sets to any output class (with the SYSOUT parameter or the CLASS parameter on the OUTPUT JCL statement) and assign messages from your job to an output class (with the MSGCLASS parameter on a JOB statement or on the JESDS parameter, and the CLASS parameter on the OUTPUT JCL

statement). You can request at remote stations, most of the same JES2 options for writing data sets that you can request when submitting the job at the central computing center. You can request that:

- A data set be held until the operator requests that it be printed.
- A special output form be used by specifying a form-name in the SYSOUT parameter. You can also request special forms using the JES2 /\*OUTPUT statement or the OUTPUT JCL statement.
- Multiple copies of the data set be produced.

Whether at a remote station or at the central computing center, you can also request that a data set be routed to another destination. With JCL and JES2 control statements, you have the following ways of routing the output data set:

- JES2 ROUTE Statement (with PRINT and PUNCH options) — Allows the user to specify the destination of jobs or output for any node or any remote station. All output that has no other specific assignment is directed to the destination specified in the JES2 ROUTE statement.

Note that if you send a job to execute at a remote node and you have a ROUTE PRINT \RMTnnn statement in the job, JES2 returns the output to RMTnnn at the node of origin. To have JES2 print the output at RMTnnn at the executing node, use the DEST parameter on a JES2 OUTPUT control statement, OUTPUT JCL statement, or sysout DD statement with the form DEST = NnnnRmmm.

- DD SYSOUT Statement — Allows a specific data set assignment by the DEST parameter on a DD statement containing a SYSOUT parameter, thus routing that data set to a particular destination.
- OUTPUT JCL Statement — Permits another way of specifying a DEST parameter for a system output data set.
- SYSOUT DD Parameter — Allows the user to specify a code name that points to a JES2 OUTPUT statement, which in turn might contain a DEST parameter identifying up to four actual destinations. This method allows the user to send more than one (but not necessarily all) data sets to the same place without coding DEST on each SYSOUT statement. If the destination should change, only the JES2 OUTPUT statement need be altered.
- Default Output Destination — Defined implicitly: when the job enters the job entry network, the default destination is determined by the device upon which the job entered the system.

If you code a destination on the SYSOUT DD statement, the JES2 OUTPUT statement, or the OUTPUT JCL statement, it will override the destination in the ROUTE statement. Work stations are identified by a destination identification established by the system programmer. The DEST parameter causes JES2 to route output to local printers or punches or to any remote station, or any node.

## Example of Obtaining Output (JES2)

This example shows the use of JES2 and JCL statements to obtain output.

```
/*PRIORITY 5
//OUTJOB JOB BAKER,PERFORM=100,MSGCLASS=J
/*JOBPARM COPIES=2,LINECT=20,ROOM=233,FORMS=GRN1
/*OUTPUT PSET DEST=PRINTER8,FCB=STD3,FORMS=2PRT,UCS=TN
/*SETUP SCHLIB
//STEP1 EXEC PGM=TESTSYSO
//OUT1 OUTPUT JESDS=ALL
//DD1 DD DSN=DATA,UNIT=3350,VOL=SER=SCHLIB,
// DISP=(OLD,KEEP),SPACE=(TRK,(5,2))
//DD2 DD DSN=&&TEMP,UNIT=3350,DISP=(NEW,DELETE),
// SPACE=(TRK,(10,5))
//DD3 DD SYSOUT=(A,,PSET)
//DD4 DD SYSOUT=(A,,GRPH)
//DD5 DD SYSOUT=L,OUTPUT=* .OUT1,DEST=HDQ
```

1. The job will be selected at priority level 5.
2. The job will run in performance group 100; the meaning of 100 is defined by the installation. All system messages are to be written to output class J.
3. The JOBPARM statement indicates that:
  - a. Two copies of the entire job-related output will be printed.
  - b. No more than 20 lines per page will be printed (LINECT = 20). You can override this LINECT parameter by coding the LINECT parameter on the OUTPUT JCL statement.
  - c. The programmer's room number is 233. This appears on the separator page and is used for distributing output.
  - d. Forms name GRN1 is the name of the form to be used by all data sets unless a specific form is defined on a DD, JES2 /\*OUTPUT, or JCL OUTPUT statement.
4. The /\*OUTPUT statement indicates that:
  - a. PSET is the code that, when indicated on a SYSOUT DD statement, causes all parameters on the /\*OUTPUT statement to override default parameters, except those coded on the OUTPUT JCL statement(s). For further information, see "Processing System Output Data Sets Using the OUTPUT JCL Statement" on page 7-44.
  - b. The destination for the output is PRINTER8. PRINTER8 does not necessarily have to be defined as a printer, it can be defined as any output device.
  - c. If the printer has the forms control buffer feature, STD3 must be the name of a member of SYS1.IMAGELIB. STD3 defines the special forms control buffer image to be used for processing any data set that has PSET coded in the SYSOUT parameter.
  - d. Forms name 2PRT is the name of the form JES2 uses for printing any data sets that have PSET coded in the SYSOUT parameter (for example, DD3).
  - e. TN is the train or UCS used in output processing.

5. The **SETUP** statement indicates that volume **SCHLIB** should be mounted before this job begins processing.
6. **SYSOUT** data sets (except **DD3** and **DD4**) are printed on the form called **GRN1**. The **DD4** **SYSOUT** data set is printed on the form called **GRPH**; the **DD3** **SYSOUT** data set is printed on the form called **2PRT** because the code name subparameter of **DD3** contains the value **PSET** (referring to the **/\*OUTPUT** statement).
7. The output data set and the accompanying system data sets from **DD5** will be processed at **HDQ**.

*Note:* For more examples of obtaining output using the **OUTPUT JCL** statement, see Chapter 14, "Coding the **OUTPUT JCL** Statement." and "OUTPUT Parameter" on page 12-103.

### **Routing a Job in a Network (JES3 Networking)**

To prepare a job for execution at a remote node:

1. Code a **JOB** statement the same as the **JOB** statement you normally code to execute a job at your installation.
2. Code a **JES3 ROUTE XEQ** statement. This statement specifies the node that is to execute the job. It causes **JES3** to divert the incoming job stream to a special input service function. This special input service function transmits the data that follows the **ROUTE XEQ** statement to the specified node.

*Note:* Without the **ROUTE XEQ** statement, the job will execute at the submitting node.

3. Code a **second JOB** statement that allows the remote job to execute at the node specified in the **ROUTE XEQ** statement. Because procedures vary between installations, the format of this **JOB** statement might be different from the format you normally use at your installation. For example, the accounting field of this **JOB** statement might not contain the same parameters in the same format that you are accustomed to using.
  - An error on the **ROUTE XEQ** statement might cause the second **JOB** statement to signal the start of a job for local execution. To avoid this possibility, specify **NJB** in the second **JOB** statement in place of **JOB**. **NJB** is changed to **JOB** by **JES3** input service prior to transmitting the job through the network.

*Note:* TSO users submitting jobs through a **JES3** network **must** specify **NJB** in place of **JOB** on the second **JOB** statement.

4. Code the remainder of the **JCL** statements for the job that is to execute at the remote node.

## Example of Routing a Job Through a JES3 Network

```
//REMOTE JOB      D58-2951
//*ROUTE XEQ      MSC
//EXAM   NJB      (DLD1,2E44,12,8)
//*MAIN
/*
```

1. REMOTE is a valid JOB statement for the submitting JES3 location.
2. The ROUTE XEQ statement directs the following job to the remote node MSC.
3. EXAM is a valid JOB statement that allows the job to execute at the remote location, MSC, specified on the ROUTE XEQ statement.

*Note:* TSO users submitting jobs through a JES3 network **must** specify NJB in place of JOB on the second JOB statement.

## Controlling Job Execution Node Using JES3 Networking

To control routing within the JES3 network, you specifically designate any jobs to be sent to other nodes with a JES3 ROUTE XEQ control statement. A ROUTE XEQ statement encountered after a local JOB statement causes JES3 to route the next job to the node specified on the ROUTE XEQ control statement. A ROUTE XEQ control statement has the format:

```
//*ROUTE XEQ nodename[.vmguestid]
```

The nodename refers to an MVS JES3 system on a global processor. The nodename cannot refer to a local processor within a JES3 node. If the nodename specifies the node on which the job was entered, the first JOB statement and the ROUTE XEQ statement will be flushed and the job that follows will execute on the node of entry.

You can also specify an MVS JES2 system, a VSE POWER node, or a VM system in the nodename. If a VM system, then specify a VM guest system id as a qualifier for the nodename.

## Controlling Sysout Routing in a JES3 Network

You can route sysout data to other nodes in a JES3 network using one or more of the following:

- The ORG parameter on the JES3 MAIN statement.
- The DEST parameter on the JES3 FORMAT statement.
- The DEST parameter on the OUTPUT JCL statement.

In addition, you can have sysout data processed by an external writer at the destination node by specifying the EXTWTR parameter on the JES3 FORMAT control statement. See *SPL: Job Management* for further information on external writers.

## Controlling Output Destination Using JES3

JES3 allows you to submit jobs to a host processing center from a work station and to route output (submitted anywhere) to work stations.

When you submit a job from a local processor or a work station, the output is returned to the place where it is submitted unless you code the **ORG** parameter on a **MAIN** statement, or you specifically route the output.

JES3 offers most of the same options for writing data sets at remote stations that you can request when submitting the job at the central computing center.

You can request that:

- A data set be held until the operator requests that it be printed.
- A special output form be used by specifying a form name on the **sysout DD** or the **OUTPUT JCL** statement.
- Multiple copies of the data set be produced.

Whether at a remote station or at the host processing center, you can also request that a data set be routed to another destination. To route an output data set to another destination, code the identification of that destination in one of the following:

- the **DEST** parameter on the **DD** statement defining the data set.
- the **MAIN ORG** statement.
- the **DEST** parameter on a **JES3 FORMAT PR** statement.
- the **DEST** parameter on a **JES3 FORMAT PU** statement.
- the **DEST** parameter on an **OUTPUT JCL** statement.

Work stations are identified by a destination identification that the system programmer establishes. The **DEST** parameter on the **DD** statement, the **JES3 FORMAT PR** and **PU** statements, and the **OUTPUT JCL** statement routes individual data sets to a remote destination (work station), a local destination (central computing center), or a specific local device.

## Remote Job Processing in JES3

Jobs can be submitted to JES3 for processing from remote work stations using remote job processing (**RJP**). Any job submitted from a remote work station will, by default, have its output (print and punch) returned to the originating work station unless you have instructed JES3 to do otherwise using **JES3 FORMAT**, **MAIN ORG**, or **OUTPUT JCL** statements. The remote user has almost all the capabilities of the local JES3 user, except that he cannot use column binary input and output or uniquely specify printer overflow specifications.

You can route a job's output to a remote destination using the **DEST** parameter on the **DD** statement, the **OUTPUT JCL** statement, or the **JES3 FORMAT** statement. See "DEST Parameter" on page 12-53, "DEST Parameter" on page 14-23, and "/\*FORMAT PR Statement" on page 17-9.

## Example of Obtaining Output (JES3)

This example shows some of the JES3 and JCL statements that can be used to obtain output.

```
//OUTJOB JOB BAKER,PERFORM=100,MSGCLASS=J
// *FORMAT PR,DDNAME=,COPIES=2,FORMS=GRN1
// *FORMAT PR,DDNAME=DD3,DEST=PRINTER8,CARRIAGE=STD3,
// *FORMS=2PRT,TRAIN=TN
//STEP1 EXEC PGM=TESTSYSO
//DD1 DD DSN=DATA,UNIT=3350,VOL=SER=SCHLIB,
// DISP=(OLD,KEEP),SPACE=(TRK,(5,2))
//DD2 DD DSN=&TEMP,UNIT=3350,DISP=(NEW,DELETE),
// SPACE=(TRK,(10,5))
//DD3 DD SYSOUT=(A)
//DD4 DD SYSOUT=(A,,GRPH)
//DD5 DD SYSOUT=L
```

1. All system messages are to be written to output class J.
2. The first `//*FORMAT` statement indicates that:
  - a. All print data sets (according to class) that do not have `//*FORMAT` statements will be printed according to the parameters on this statement unless the output class defines specific processing characteristics because `DDNAME` is coded without a name (`DDNAME=,`) and applies to all output data sets for the job.
  - b. JES3 uses the form named `GRN1` and prints two copies of all data sets unless a specific form or number of copies is defined on a `DD` statement or for a class by the installation.
3. The second `//*FORMAT` statement indicates that:
  - a. The destination for the output is a printer that has an installation-defined name of `PRINTER8`.
  - b. If `PRINTER8` has the forms control buffer feature, `STD3` must be the name of a member of `SYS1.IMAGELIB`. `STD3` defines the special forms control buffer image or carriage tape to be used for processing the job.
  - c. Forms name `2PRT` is the name of the forms for `DD3`.
  - d. `TN` means text printing on a 1403, 3211, or 3203-5 printer.

*Note:* For examples of obtaining output using the `OUTPUT` JCL statement, see Chapter 14, "Coding the `OUTPUT` JCL Statement," and "OUTPUT Parameter" on page 12-103.

## Job Log

The system produces a job log, which is a record of job-related information for the programmer. It is a system output (sysout) data set written to the output listing for the job's message class, which is specified by the `MSGCLASS` parameter on the `JOB` statement. The job log can consist of:

- The `JOB` statement.
- Other JCL statements from the input stream and cataloged procedures.
- JCL messages.
- JES and operator messages about the job: the allocation of devices and volumes, disposition of data sets, and termination of job steps and the job.

By coding the `MSGLEVEL` parameter on the `JOB` statement, you inform the system of what statements and messages you want included in the job log. The notation used on the job log to identify cataloged and in-stream procedure statements is described in Chapter 9, "Guide to Cataloged and In-Stream Procedures" on page 9-1.

By coding the `MSGCLASS` parameter on the `JOB` statement, you assign the job log to an output class. The system uses an installation-defined default if `MSGCLASS` is not coded.

By coding the `NOLOG` parameter on the `JES2 JOBPARM` statement, you can control the printing of the JES2 job log.

By coding the `FETCH` parameter on the `JES3 MAIN` statement, you can control whether JES3 routes or displays the fetch messages from the JES3 main device scheduler. However, JES3 still sends the main device scheduler fetch messages to the `JESMSG` data set.

By coding the `DDNAME` parameter on the `JES3 /*FORMAT` statement, you can control the printing of messages that pertain to the output data set the system is processing.

You can also control the processing options of the system data sets by using the `JESDS` keyword on the `OUTPUT JCL` statement. For further information, see "Managing the System-Managed Data Sets: The `JESDS` Parameter" on page 7-49.

### **MSGLEVEL Parameter**

The `MSGLEVEL` parameter on the `JOB` statement tells the operating system what message information you want as output from your job. You can request the following output:

- The `JOB` statement
- All input JCL statements



- All cataloged procedure statements for procedures invoked by the job's steps, and the internal representation of procedure statements after symbolic parameter substitution
- Allocation and disposition (allocation/termination) messages.

The format of the MSGLEVEL parameter is:

```
MSGLEVEL=( [statements] [,messages] )
```

Replace "statements" with one of the following:

- 0 only print the JOB statement.
- 1 print all JCL statements in your job, cataloged procedure statements, and the internal representation of statements after symbolic substitution.
- 2 only print the JCL statements in your job (cataloged procedure statements will not appear).

Replace "messages" with one of the following:

- 0 no allocation/termination messages are to appear, unless the job abnormally terminates. If abnormal termination occurs, these messages will appear as output.
- 1 all allocation/termination messages are to appear.

If you omit the MSGLEVEL parameter or one of the subparameters, MVS assumes a default value established by your installation. Your manager or supervisor should tell you the defaults chosen in your installation.

For example, if you want MVS to display only the JOB statement and no allocation/termination messages, code:

```
MSGLEVEL=( 0,0 )
```

If you want MVS to display only the JCL statements in your job and all allocation/termination messages, code:

```
MSGLEVEL=( 2,1 )
```

If you want to use your installation's default for JCL statements and MVS is not to display any allocation/termination messages, code:

```
MSGLEVEL=( ,0 )
```

If you want all your JCL statements, cataloged procedures, and internal representation of statements, and your installation's default for allocation/termination messages, code:

```
MSGLEVEL=1
```

If you want your installation's defaults for both JCL statements and allocation/termination messages, omit the MSGLEVEL parameter.

For information on coding the MSGLEVEL parameter, see "MSGLEVEL Parameter" on page 10-16.

## MSGCLASS Parameter

The MSGCLASS parameter allows you to specify the output class to which MVS is to write the job log.

The format of the MSGCLASS parameter is:

```
MSGCLASS=class-name
```

Replace "class-name" with a letter (A-Z) or a number (0-9).

The system produces your job log on a device assigned to the class you selected. If you omit the MSGCLASS parameter, JES determines the default for the MSGCLASS parameter by the input source of the job. That is, the particular card reader or work station, or whether the job was submitted by a time-sharing user.

You can route the job log and output data sets to the same output class. To do this, code the same output class in both the MSGCLASS parameter on the JOB statement and the SYSOUT parameter on the DD statements for the data sets. Or, if you code SYSOUT=\* on all DD statements for the output data sets, the system uses the same output class you specified in the MSGCLASS parameter on the JOB statement. Note that the MSGCLASS parameter can be overridden on the OUTPUT JCL statement.

Your manager or supervisor should tell you which output classes are available in your installation. Some of these output classes are standard and some are reserved for special uses. You may have to notify the operator if you are using a special output class in your job because he has to start an output writer for that output class in order to obtain the output. For further information, see *SPL: Job Management*.

For additional information on coding the MSGCLASS parameter, see "MSGCLASS Parameter" on page 10-14.

## JES2 Hard-Copy Log

In addition to the job log, JES2 produces a hard-copy log for the job. The job's JES2 hard-copy log contains a list of job-related console messages and operator replies that JES2 produces while processing your job. You can request that JES2 suppress the hard-copy log output using the JES2 JOBPARM statement. The format of the JES2 JOBPARM statement is:

```
/*JOBPARM parameters
```

To suppress the job's JES2 hard-copy log, code:

```
/*JOBPARM NOLOG
```

Place the JES2 JOBPARM statement after the JOB statement:

```
//JOBXO JOB      ....  
/*JOBPARM NOLOG  
//S1     EXEC    ....  
:  
:
```

For additional information on coding the JOBPARM statement, see “/\*JOBPARM Statement” on page 16-4.

You can also control the job’s JES2 hard-copy log by using the JESDS parameter on the OUTPUT JCL statement. For further information on the JESDS parameter, see “JESDS Parameter” on page 14-37.

## JES3 Main Device Scheduler Messages

The JES3 main device scheduler (MDS) issues allocation (fetch), mounting, and deallocation messages for all JES3 and JES3/MVS (jointly) managed devices.

Using the FETCH parameter on the JES3 MAIN statement you can override the default specified for FETCH at JES3 initialization. The format of the JES3 MAIN statement is:

```
/*MAIN parameters
```

Using the FETCH parameter you can request that the MDS:

- Issue all fetch messages for all volumes in DD statements that use JES3 setup devices
- Issue no fetch messages
- Issue fetch messages for volumes in DD statements specified by the SETUP parameter on the JES3 MAIN statement
- Issue fetch messages for volumes on specific DD statements
- Not issue fetch messages for volumes on specific DD statements

The format of the FETCH parameter on the JES3 MAIN statement is:

```
FETCH={ALL|NONE|SETUP|[/](ddname[,ddname]...)}
```

For specific information on coding the FETCH parameter, see “/\*MAIN Statement” on page 17-23.

## JES3 System Messages

The JES3 FORMAT statement allows you to code the ddname of the DD statement that defines the output data set characteristics you want to specify. If you want system messages, code on the FORMAT statement:

```
//*FORMAT PR DDNAME=SYSMSG
```

If you want the JCL statements and messages, code on the FORMAT statement:

```
//*FORMAT PR DDNAME=JESJCL
```

If you want JES3 and system operator messages (job log), code on the FORMAT statement:

```
//*FORMAT PR DDNAME=JESMSG
```

You can also control JES3 job log messages, system messages, and JCL statement messages by using the JESDS parameter on the OUTPUT JCL statement. For further information on the OUTPUT JCL statement, see Chapter 14, "Coding the OUTPUT JCL Statement."

For additional information on coding the FORMAT PR statement, see "//\*FORMAT PR Statement" on page 17-9.

## TSO

TSO allows a number of users to execute programs concurrently and to interact with the programs during execution.

You can request that the system notify you when your background job completes. Under TSO, a background job is one that is entered through a time sharing terminal by means of the SUBMIT command or by executing a step to run TSO in the background. For more information, see *OS/VS2 TSO Command Language Reference*.

The JCL NOTIFY parameter, on the JOB statement, allows you to indicate to the system that you want automatic notification when your job completes.

The JES2 NOTIFY control statement allows you to indicate to the system that you want notification messages directed to the userid you specify.

The USER parameter on the JES3 MAIN control statement allows you to specify a userid that indicates, to JES3, a TSO user who can access, receive, or inquire about a data set.

## NOTIFY Parameter

By coding NOTIFY on the JOB statement, you are requesting that a message be sent to your terminal when your job completes.

The format of the NOTIFY parameter is:

NOTIFY=userid
---------------

Replace the term “userid” with a user identification expressed in 1 to 7 alphanumeric characters; the first character must be alphabetic. The user identification can be the same one you use when you start your terminal session; that is, the same user identification you use in the LOGON command must be the one used in the NOTIFY parameter, if you want to notify yourself. You can notify any other user using their valid ID in place of your own. For example, if a user submits a job named FORMS through the terminal and his user identification is JOHNS, and he wants to be notified upon completion of his job, then his JOB statement if he used NOTIFY should be:

```
//FORMS JOB NOTIFY=JOHNS
```

For more information on coding the NOTIFY parameter on a JOB statement, see “NOTIFY Parameter” on page 10-18. For a more detailed discussion of TSO, refer to *TSO Command Language Reference*.

## The JES2 NOTIFY Control Statement

Use the JES2 NOTIFY control statement to have the system direct a job’s notification messages to the userid you specify. The format of the NOTIFY statement is:

```
/*NOTIFY { .userid }
          { nodename{:userid } }
          { /userid }
          { (userid) }
          { userid }
```

To specify that the system is to send a job’s notification messages to a node other than the job’s node of origin, code:

```
/*NOTIFY nodename.userid
```

To specify that JES2 is to send a job’s notification messages to the job’s node of origin, code:

```
/*NOTIFY userid
```

*Note:* The userid you code on this statement overrides any specification you code in the NOTIFY parameter on the JOB statement.

For more information on coding the JES2 NOTIFY statement, see “/\*NOTIFY Statement” on page 16-11.

## The USER Parameter on the JES3 MAIN Control Statement

The USER parameter allows:

- A TSO user to access SYSOUT data sets via the TSO OUTPUT command
- A TSO user to inquire about the status of a job or to cancel the job
- Data sets to be sent to a main processor for use by a TSO user

The format of the USER parameter is:

```
//*MAIN USER=userid
```

To send SYSOUT data sets to an MVS main processor for use by a TSO user, code:

```
//*MAIN ACMAIN=main-name,USER=userid
```

For additional information on the MAIN statement, see “*//\*MAIN Statement*” on page 17-23 and TSO in *JES3 SPL: Initialization and Tuning*.

## Remote Job Processing

Remote job processing allows you to enter a job into the JES2 or JES3 job stream by way of a remote work station or device.

There are some differences between JES2 and JES3; therefore each is discussed under a separate heading.

### JES2 Remote Job Processing

The remote job entry (RJE) facility of JES2 allows a remote work station to use the job entry subsystem. JES2 processes remote jobs the same way it processes those received from local readers, printers, and punches.

Remote job entry is the ability to submit jobs and receive system output at remote facilities as if the jobs had been submitted at a local facility. JES2 supports both **systems network architecture (SNA)** and **binary synchronous communication (BSC)** remote stations as RJE facilities. The remote facilities may be attached to JES2 by synchronous data link control (SDLC) or by a point-to-point (BSC) communications link. The remote facility becomes a logical extension of the local computer facility and JES2 expects it to be under the control of a remote operator.

There are two types of remote facilities. The first type is a **remote terminal**, which does not have a processor. A remote terminal, for example a 2780 or 2770, can be used for entering jobs into and receiving data from JES2. The second type is a **remote work station**, which has a processor. A processor, for example, System/3 or System/370, executes a JES2-generated program that allows the processor to send jobs to, and receive data from, JES2. Also part of the remote work station are printers, punches, card readers, and a console.

#### SNA RJE for JES2

Remote job entry stations that use the facilities of a SNA network gain access to JES2 through VTAM. For more information, see remote job entry in *SPL: JES2 Initialization and Tuning* and *SPL: VTAM*.

#### BSC RJE for JES2

Communication between the local processor and BSC remote work stations use a JES2 facility called MULTI-LEAVING. Multi-leaving allows the transmission of multiple print and punch streams at the same time and allows JES2 to receive multiple console messages and input streams. For more information, see remote job entry in *SPL: JES2 Initialization and Tuning* and *SPL: VTAM*.

## Remote Job Entry Stations

Installations can configure remote lines as dedicated or nondedicated. This configuration is established during JES2 initialization. The following discussion pertains to nondedicated remote lines only.

**BSC remote work stations:** use the JES2 SIGNON control statement to notify JES2 of a connection request. See the discussion of the SIGNON statement below.

**SNA remote work stations:** must use the LOGON command instead of the SIGNON statement to notify JES2 of a connection request. For a discussion of the LOGON command, refer to *SPL: JES2 Initialization and Tuning* and *SPL: VTAM*.

### The JES2 SIGNON Control Statement

Use the SIGNON statement to indicate to the central processor that you wish to begin a remote job stream processing session. The format of the JES2 SIGNON statement is:

```
/*SIGNON      {REMOTEnnn}  
              {RMTnnnn } [password1]          [password2]  
              {RMnnnn  }
```

Place the SIGNON statement at the end of the JES2/RTP program deck for multi-leaving remote stations. For non-multileaving remote stations, JES2 transmits the SIGNON statement alone as part of the initial connection process.

REMOTEnnn, RMTnnnn, or RMnnnn specifies the identification number assigned to the remote station requesting to sign on.

*Note:* If you code REMOTEnnn on the SIGNON statement, and you code RMT on the ROUTE statement, you are restricted to coding RMT on the ROUTE statement with only three digits (RMTnnn).

*password1* is a communication line password assigned to a dial line that allows the remote station access to JES2 for remote job stream processing. This password guarantees a user or a group of users the use of a line and prevents unauthorized remote operators from using the line to gain access to JES2.

*password2* is a password that ensures that the remote station signing on is a valid RJE (remote job entry) station.

For more information on coding the SIGNON statement, see “/\*SIGNON Statement” on page 16-30.

### The JES2 SIGNOFF Control Statement

To terminate a remote job stream processing session, use the JES2 SIGNOFF statement. Both SNA and BSC remote work stations can use the SIGNOFF statement. SNA remote work stations can also use the LOGOFF command to end a session with JES2. For a discussion of the LOGOFF command, refer to *SPL: JES2 Initialization and Tuning* and *SPL: VTAM*.

The format of the SIGNOFF statement is:

```
/*SIGNOFF
```

For more information on coding the SIGNOFF statement, see “/\*SIGNOFF Statement” on page 16-29.

## JES3 Remote Job Processing

JES3 remote job processing (RJP) allows you to enter a job into the JES3 job stream using an input source connected to the global processor by a data line. JES3 processes the job as if it had been submitted locally.

Devices attached to a processor via channels are called **local devices**; devices attached to a processor via a data link are called **remote devices**. Any output from a remotely-entered job can, at your option, be retained at the host system, transmitted to the originating location, or sent to another location.

JES3 supports two types of remote devices for RJP. The first type are those attached using **binary synchronous communications (BSC)** protocols. The second type are those attached using **synchronous data link control (SDLC)** protocols within the **IBM systems network architecture (SNA)**.

### Remote Work Stations (JES3)

Installations can configure remote lines as dedicated or nondedicated. This configuration is established during JES3 initialization. The following discussion pertains to nondedicated remote lines only.

**BSC remote work stations** use the JES3 SIGNON control statement to notify JES3 of a connection request. See the discussion of the SIGNON statement below.

**SNA remote work stations** must use the LOGON command instead of the SIGNON statement to notify JES3 of a connection request. For a discussion of the LOGON command, refer to *JES3 SPL: Initialization and Tuning* and *SPL: VTAM*.

### The JES3 SIGNON Control Statement

Use the SIGNON statement to indicate to JES3 that you wish to begin a remote job stream processing session. The format of the JES3 SIGNON statement is:

```
/*SIGNON work-station-name {A|(blank)} {R|(blank)} passwd1 passwd2
```

The fields for the JES3 SIGNON statement have the following meanings:

- The *work-station-name* identifies the remote station requesting to sign on.
- An *A* in column 22 identifies the remote work station as programmable and specifies that, for the duration of this session, the processor automatically reads from the reader.



- An *R* in column 23 identifies the remote work station as nonprogrammable and specifies the output suspension feature. That is, if a print or punch data set is currently active, it can be suspended if the active device is not ready.
- *passwd1* is a line password assigned to a dial line that allows the remote station access to JES3 for remote job stream processing. This password guarantees a user or a group of users the use of a line and prevents unauthorized remote operators from using the line to gain access to JES3.
- *passwd2* is a password that ensures that the remote station signing on is a valid RJP (remote job processing) work station.

For more information on coding the SIGNON statement, see “/\*SIGNON Statement” on page 17-50.

### The JES3 SIGNOFF Control Statement

To terminate a remote job stream processing session, use the JES3 SIGNOFF statement. Both SNA and BSC remote work stations can use the SIGNOFF statement. SNA remote work stations can also use the LOGOFF command to end a session with JES3. For a discussion of the LOGOFF command, refer to *JES3 SPL: Initialization and Tuning* and *SPL: VTAM*.

The format of the SIGNOFF statement is:

```
/*SIGNOFF
```

For more information on coding the SIGNOFF statement, see “/\*SIGNOFF Statement” on page 17-49.

## Special Job Processing

Using JOB statement parameters and JES control statement parameters you can request special processing options of the operating system. The special processing options you can request are:

- Delaying Initiation of Your Job
- Delaying Initiation of Other Jobs in JES3
- Bypassing Job Initiation
- Testing JCL Without Execution
- Copying JCL Without Execution in JES2
- Reading Column Binary Input
- Deadline Scheduling
- Dependent Job Control for JES3

### Delaying Initiation of Your Job in JES2

Although you can influence a job’s selection by assigning a job class and priority to the job, you cannot predict whether JES2 will select a job in one job class for execution before another job in a different job class.

When jobs exist in the same job class, you cannot be certain that one job will complete execution before JES2 selects the other job, even if you assign a higher priority to the first job.

You must use the `TYPRUN=HOLD` parameter because the higher priority controls only the *selection* sequence and does not guarantee that the first job will complete execution before the second is selected. In some cases, when submitting two jobs, `JOBA` and `JOB B`, `JOBA` must complete execution before JES2 can initiate `JOB B`; `JOBA` might create records that `JOB B` uses. You will have to instruct JES2 to delay `JOB B`'s initiation until `JOBA` completes execution.

It is also possible that resources a job requires will not be available. Therefore, you will want to delay the job's initiation until required resources are available. The job remains on the execution queue or JCL conversion queue, but JES2 will not select the job for processing until the operator releases the job. Use the `/*MESSAGE` control statement to notify the operator to release the job. When the operator releases the job, it is again eligible for selection by JES2.

Delay a job's initiation by coding on the `JOB` statement:

```
TYPRUN=HOLD
OR
TYPRUN=JCLHOLD
```

Alternatively, you can specify a job class defined by the installation to force a `TYPRUN=HOLD` default.

For more information on coding the `TYPRUN` parameter on a `JOB` statement, see "TYPRUN Parameter" on page 10-39.

**JES2 users** can delay a job's initiation and have specific volumes mounted before the job executes by coding the `JES2 SETUP` statement. The `SETUP` statement notifies the operator which volumes the job requires.

If you code a `JES2 SETUP` statement, you can notify the operator what volumes to retrieve from the library. The operator will mount the requested volumes and then should release the job that has been held on the execution queue or on the JCL Conversion Queue.

For more information on coding the `JES2 SETUP` statement, see "/\*SETUP Statement" on page 16-28.

*Note:* JES3 does not support the use of `TYPRUN=HOLD`. Instead **JES3 users** can specify the `HOLD` parameter on the `JES3 MAIN` statement. This produces the same effect as coding `TYPRUN=HOLD` on the `JOB` statement. Additionally, JES3 users can use dependent job control to delay a job's initiation. For information on coding the `JES3 MAIN` statement, see "/\*MAIN Statement" on page 17-23. For information on using dependent job control, see "Dependent Job Control for JES3: The Job Net" on page 3-27.

### Delaying Initiation of Other Jobs (JES3)

Sometimes the function of your job is to update a member of a procedure library. To prevent other jobs from using the data set being updated, JES3 users should code the `UPDATE` parameter on the `JES3 MAIN` statement. This parameter identifies the procedure library data set(s) being updated. The parameter causes all jobs using this data set and all data sets concatenated to it to be held until the update is complete.

You can also use the `UPDATE` parameter when updating a private library. The installation must define any private library at JES3 initialization.

For information on coding the `JES3 MAIN` statement, see "/\*MAIN Statement" on page 17-23.

## Bypassing Job Initiation

Under certain conditions you may wish to scan the control statements for syntax errors without submitting the associated input data sets, or you may wish to produce a copy of your input deck without actually initiating any steps. To scan the control statements for syntax errors without initiating the job, code on the JOB statement:

```
TYPRUN=SCAN
```

Or you can select a job class that the installation has defined to force the TYPRUN=SCAN default. When you code this option, the system first scans the job for control statement syntax errors and then passes directly to the output stage for processing.

TYPRUN=SCAN causes the system to check for coding errors that it can detect at the converter phase. That is, invalid keywords, illegal characters, and parenthesis errors. The system does not check for errors that occur at the interpreter phase. For example, misplaced statements, the syntax of JCL statements in cataloged procedures, or the syntax of subparameters of JCL parameters.

For more information on coding the TYPRUN parameter on a JOB statement, see "TYPRUN Parameter" on page 10-39.

## Testing JCL Without Execution (JES3)

JES3 provides another method to test your job's JCL in addition to the TYPRUN=SCAN parameter on the JOB statement. JES3 users can code PGM=JCLTEST or PGM=JSTTEST on the EXEC statement. This causes the system to scan the JCL for syntax errors. These programs will check for invalid keywords, illegal characters, parentheses errors, and excessive parameters without processing the job or setting up devices. For further information about JSTTEST and JCLTEST, see *JES3 SPL: Diagnosis*.

## Copying JCL Input Without Execution in JES2

To produce a copy of the input deck without initiating any steps, code on the JOB statement:

```
TYPRUN=COPY
```

Or you can select a job class that the installation has defined to force the TYPRUN=COPY default. When you code this option, JES2 reads the input deck (as submitted) directly to a SYSOUT data set and schedules it for output processing. The class of the SYSOUT data set is the same as the message class of the job. Therefore, you can control the class using the MSGCLASS parameter on the JOB statement.

The SYSOUT data set generated can be processed by either the JES2 print/punch processor or by an external writer. It is available to the TSO OUTPUT command only if the output class is a held sysout class. Before adding control statements to the SYSOUT data set, JES2 interprets control statements that it encounters in the input stream. JES2 copies job control language (JCL) statements without any processing (that is, without JCL conversion).

When you specify TYPRUN=COPY on a JOB statement, JES2 ignores any OUTPUT JCL statements during output processing.

For more information on coding the TYPRUN parameter on a JOB statement, see "TYPRUN Parameter" on page 10-39.

## Reading Column Binary Input

Jobs that require input from column binary cards can receive this input directly from the DD statement using JES3. To do this, code the `MODE=C DCB` subparameter on the DD \* or DD DATA statement that precedes the column binary card input or notify the operator to read this job into a card reader for which he has specified mode C processing.

The DATASET statement can also be used to read column binary input for installation-written routines executed as part of nonstandard jobs. For a discussion of nonstandard jobs, see “`/*PROCESS Statement`” on page 17-44.

## The JES2 SETUP Statement

Use the JES2 SETUP statement to tell the operator what volumes your job needs to execute.

Use of the SETUP statement in a JES2 network generally requires that the SETUP statement follow any ROUTE or XEQ statement. This prevents JES2 from requesting the setup on a node other than the node of execution. If JES2 processes the SETUP statement prior to processing the ROUTE or XEQ statements, JES2 requests the setup on both the input and execution nodes.

The format of the JES2 SETUP statement is:

```
/*SETUP volume-serial-number[,volume-serial-number]...
```

- Place all SETUP statements after the JOB statement.
- Code as many SETUP statements as necessary.

For example:

```
/*SETUP 666321,149658
```

When the job enters the system, JES2 lists on the console the two volumes requested. JES2 then places the job in hold status awaiting release by the operator when the required volumes are available. The message informs the operator to mount the volumes before running the job.

For more information on coding the JES2 SETUP statement, see “`/*SETUP Statement`” on page 16-28.

## JES3 SETUP Parameter

Use the SETUP parameter on the JES3 MAIN statement to modify the standard algorithm JES3 uses in assigning devices to a job prior to job execution. The format of the SETUP parameter is:

```
SETUP={JOB|HWS|THWS|[/](ddname[,ddname]...)}
```

For a discussion of the JES3 SETUP parameter, see “`/*MAIN Statement`” on page 17-23. For a discussion of JES3 allocation, see “`Allocating Data Resources in a JES3 System`” on page 6-4.

## Deadline Scheduling for JES3

When a job must be scheduled by a certain time of the day, week, month, or year, specify the DEADLINE parameter on the JES3 MAIN statement. By indicating that there are time restrictions, you influence the priority of the job and help insure that the job will be scheduled when necessary. For example, a job must be scheduled every Friday by 2 p.m. to calculate the payroll. Request that the job be executed by that time by coding:

```
//*MAIN DEADLINE=(1400,A,6,WEEKLY).
```

The subparameter values mean the following:

### 1400

is 2 p.m. on a 24-hour clock.

### A

defines the deadline type that determines the periodic increment of the job's priority (the installation defines the meaning for A).

### 6

is the sixth day of the week - Friday (the first day is Sunday; the seventh day is Saturday).

### WEEKLY

is the cycle indicating the frequency of scheduling this job.

The purposes of deadline scheduling are to allow submission of a job at its true priority level and to have JES3 schedule it to best use the available resources. JES3 increases the priority level only if the job is not scheduled on time. For example, if you work first shift and submit a job at the end of the day, you do not need results until the next morning. Indicate that the job must be scheduled by 7 a.m. and assign an initial lower priority, then the job can be scheduled at any time. If it has not been scheduled a few hours before the 7 a.m. deadline, JES3 increases the priority periodically to increase the job's chances for being selected by 7 a.m.

If you have requested that a job be scheduled by a certain time on a certain day and the job is submitted after the deadline time, JES3 increases the priority of the job to the same level it would have been if the job had been submitted prior to the deadline and had not completed.

For more information on coding the DEADLINE parameter on the JES3 MAIN statement, see “//\*MAIN Statement” on page 17-23.

## Dependent Job Control for JES3: The Job Net

JES3 installations can use dependent job control (DJC) when jobs must be executed in a specific order. There are several reasons to execute one job before another. For example, JES3 fetches data set information from the catalog before scheduling a job. If JOBA changes or adds to the catalog that JOBB will refer to, use dependent job control to ensure that JOBA runs before JES3 processes JOBB allocation. Another reason to use DJC is to achieve better device utilization. If a job requires only one device for the first four steps but requires five devices for the fifth step, break the job into two jobs (one for the first four steps and one for the fifth step). Use DJC to make the second job dependent on the first; that is, the second job can run only after the first job has completed. DJC is also useful in controlling the scheduling of jobs that have data dependencies. When you use dependent job control, the group of jobs that depend on each other form a **dependent job net**. To indicate to JES3 the relationship of jobs to each other in a dependent job net, use a JES3 NET control statement.

## The JES3 NET Control Statement

To define a dependent job net, submit a NET statement with each job. The NET statement identifies a job's net and specifies the dependency that must be satisfied before the job can be scheduled.

Jobs normally must wait for scheduling until a predecessor job completes. Jobs that depend on one or more predecessor jobs to complete are called successor jobs. To specify the number of predecessor/successor relationships of a given job in a net, specify the number of predecessor jobs on the NHOLD parameter and the name of each successor job in the RELEASE parameter of the NET statement. The number of predecessors is the number of jobs immediately prior to the job that depends on other jobs completing; the number of successors is the total number of all jobs remaining to be processed in the net that depend on this job completing.

Some of the parameters you use on a JES3 NET statement are discussed in the following topics. For a discussion of all of the parameters you can code on a NET statement, see "/\*NET Statement" on page 17-35.

### Specifying System Action for Termination of a Job in a Net

A normal or abnormal predecessor completion can be the requirement established for going to the next job. For example, a job might not be requested unless the predecessor job abnormally terminates. The NORMAL and ABNORMAL parameters on the NET statement specify the kind of predecessor completion required for the successor dependent job to execute.

A job-net job that has previously completed normally can be resubmitted while the associated net is still active in the system. The resubmitted job does not enter the net nor does it affect the net's processing. If you want to resubmit the previously completed job as part of the net, you must first free the net from the system and then resubmit the entire net. Note that if you code the COND parameter on a JOB or EXEC statement, and the job terminates due to condition codes, JES3 treats the job as an abnormally terminated job.

### Placing a Job in a Net on Hold

Use the NHOLD parameter to specify the number of immediate predecessor jobs that must complete before a job is released for scheduling; the number can include jobs from another net that are predecessors to the dependent job. When this parameter is defined, the job is placed into dependent job control hold status when it enters the system. A job is made eligible for JES3 allocation and scheduling when its NHOLD count becomes zero. This count is decreased when each predecessor job completes execution or by the operator. However, the NHOLD value can be decreased before the predecessor job completes execution if you issue a DJC (dependent job control) WTO (write to operator) macro in the predecessor job problem program. Refer to *Supervisor Services and Macro Instructions*, for the format of the write to operator command and *JES3 Messages* for message text information.

To place jobs that are in a dependent job net in operator hold, code the NET OPHOLD parameter. This parameter prevents scheduling of the job until the operator explicitly releases it from hold.

Upon either normal or abnormal completion of a predecessor job, a successor job can have its NHOLD count decreased, can be flushed from the system, or can be retained pending operator action. If it is flushed, the job and all of its successor jobs (and their successor jobs) are canceled, printed, and flushed from the system. If it is retained in the held state,

the NHOLD count is not decreased and the job and all of its successor jobs are suspended from scheduling until either the predecessor is resubmitted or the operator decreases the NHOLD count. You can control external dependencies by setting the NHOLD value one greater than normally assigned and asking the operator to decrease the NHOLD count when the dependency is satisfied.

### **Specifying Early Setup of Resources for a Job in a Net**

Early setup of successor job resources is indicated using the RELSCHCT parameter. It allows a job to enter JES3 allocation before all predecessor jobs have completed. The job is then placed in a hold status until all of its predecessors complete processing. Early setup of successor job resources is invoked when the NHOLD count becomes less than or equal to the RELSCHCT count. Do not use this option with jobs that have catalog dependencies. Coding the RELSCHCT parameter can tie up devices and data sets for a long time, so use it carefully.

### **Establishing Dependencies Between Different Nets**

You can establish dependencies between jobs in different nets. To indicate that a job in one net is the predecessor to a job in another net, specify the NETREL parameter. For examples of using the NETREL parameter, see the dependent job control examples on the following pages.

### **Specifying Devices Dedicated to a Net**

You can dedicate devices to a dependent job net by coding the DEVPOOL parameter. When you code the DEVPOOL parameter in the first job in a net (it is ignored if not coded in the first job), the devices specified are dedicated for device allocation and volume mounting only by jobs in the same net.

To release these devices prior to all jobs completing in the network, code the DEVRELSE parameter. This parameter may be specified on one or more jobs in the net, except the first job. The first completing job that contains DEVRELSE = YES will cause the dedicated devices to be released. If no such job is encountered, the devices are released when the net is purged.

### **How to Code NET Statements**

When a job is part of a net, the number of predecessor jobs and the names of all successor jobs must be indicated on the NET statement. A diagram is a good way to graphically show the relationship of jobs in a net. Once you describe a net of dependent jobs in a diagram, you can list the dependencies in a table and then translate that into NET statements (see the following three examples). The following is a guideline for defining dependent job control nets:

1. Draw a diagram of the net, connecting dependent jobs with lines indicating the flow of job dependencies. Give the net a name (such as EXAM1) to identify the net; this becomes the NETID parameter value.
2. List the jobname of each job in the net in the order of their dependencies on one another. Note next to each jobname the number of predecessors to the job, including predecessors of other job-nets, if any. The number of predecessors becomes the NHOLD parameter value. If early setup scheduling is desired, specify RS=count (RELSCHCT = count) where count specifies setup of a dependent job's resources before all of its predecessors have completed execution.
3. List the disposition of each successor jobname based on normal or abnormal predecessor completion.

4. List the successor jobnames for each job in the diagram. If there is a successor in a different net, then list the successor jobname and successor net-id in parentheses. The successors become the RELEASE parameter values.
5. Construct the necessary NET statements based on the diagram.

One way to verify the net is to execute the IEFBR14 program for each job in the net, simulating normal and abnormal completions. The general format for each job of the net is:

```
//jobname JOB
//*NET your specific parameters
//STEP1 EXEC PGM=IEFBR14
/*
```

In this way, all DJC net functions and definitions can be tested without using actual jobs.

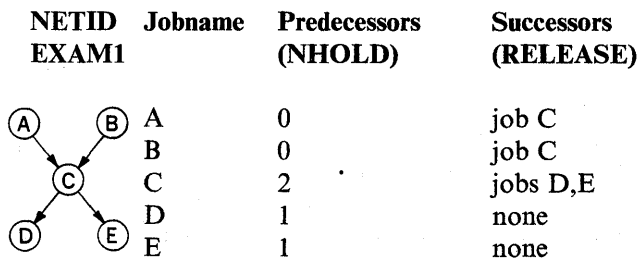
### Examples of Dependent Job Control

Instead of coding the full name of the parameters for every job, you can use the short form of the following parameters.

Parameter	Short Form
NETID	ID
NHOLD	HC
RELEASE	RL
NORMAL	NC
ABNORMAL	AB
OPHOLD	OH
RELSCHCT	RS
NETREL	NR

1. A simple net

Given: five jobs, A, B, C, D, and E.





**How to code EXAM1:**

Jobname	Control	Statement
A	//*NET	NETID=EXAM1,RELEASE=(C)
B	//*NET	NETID=EXAM1,RELEASE=(C)
C	//*NET	NETID=EXAM1,NHOLD=2,RELEASE=(D,E)
D	//*NET	NETID=EXAM1,NHOLD=1
E	//*NET	NETID=EXAM1,NHOLD=1

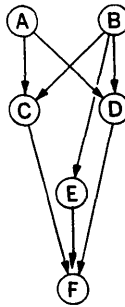
If the system scheduled this net of jobs with defined dependencies, you could achieve the desired sequence only through operator action. By using JES3 dependent job control, operator intervention is not required. Jobs A and B can run concurrently, followed by job C, and then jobs D and E can run concurrently.

2. Multiple predecessor jobs

Given: six jobs, A, B, C, D, E, and F.

the NETID is EXAM2.

NETID	Jobname	Predecessors (NHOLD)	Successors (RELEASE)	Disposition
EXAM2	A	0	jobs C,D	
	B	0	jobs C,D,E	AB = R (retain job) NC = D (decrease job)
	C	1	job F	AB = R (retain job) NC = D (decrease job)
	D	2	job F	AB = F (flush job) NC = D (decrease job)
	E	1	job F	AB = R (retain job) NC = D (decrease job)
	F	3	none	AB = R (retain job) NC = D (decrease job)



**How to code EXAM2:**

Jobname	Control	Statement
A	//*NET	NETID=EXAM2,RELEASE=(C,D)
B	//*NET	NETID=EXAM2,RELEASE=(C,D,E)
C	//*NET	NETID=EXAM2,RELEASE=(F),NHOLD=1
D	//*NET	NETID=EXAM2,RELEASE=(F),NHOLD=2,ABNORMAL=F
E	//*NET	NETID=EXAM2,RELEASE=(F),NHOLD=1
F	//*NET	NETID=EXAM2,NHOLD=3

If either job A or B abnormally terminates, job D is flushed from the system, thereby causing job F also to be flushed. Jobs C and E remain in the system. In this situation, correct the predecessor and resubmit the jobs to the system. When it completes normally, its successors, C and E, are made eligible for scheduling. Because job C has NHOLD=1 it requires that only job A or B complete normally. However, job D, which has NHOLD=2, requires that both jobs A and B complete normally.





### 3. Complex network

Given: two networks, EXAM4 and EXAM3.











EXAM4 contains four jobs, W,X,Y, and Z.

EXAM3 contains ten jobs, A,B,C,D,E,F,G,H,I, and J.

The net to be released (NETREL) for job I is EXAM4, the release jobname is Y.

EXAM4 Jobname	Predecessors (NHOLD)	Successors (RELEASE)	Disposition
 W	0	job X	
 X	1	job Y	AB = R (retain job) NC = D (decrease job)
 Y	2*	job Z	AB = F (flush job) NC = D (decrease job)
 Z	1	none	AB = F (flush job) NC = D (decrease job)

\*Job Y has one predecessor in this net and one predecessor in EXAM3.

EXAM3 Jobname	Predecessors (NHOLD)	Successors (RELEASE)	Disposition
 A	0	job C	
 B	0	jobs C,D	AB = R (retain job) NC = D (decrease job)
 C	2	job E	AB = R (retain job) NC = D (decrease job)
 D	1	jobs E,I	AB = R (retain job) NC = D (decrease job)
 E	2	jobs F,H	AB = R (retain job) NC = D (decrease job)
 F	1	job G	AB = R (retain job) NC = D (decrease job)
 G	1	none	AB = R (retain job) NC = F (flush job)
 H	1	none	AB = F (flush job) NC = D (decrease job)
 I	1	(EXAM4,Y)	AB = R (retain job) NC = D (decrease job)
 J	1	none	AB = R (retain job) NC = D (decrease job)

#### How to code EXAM4:

Jobname	Control	Statement
		(using short form of parameters)
W	//*NET	ID = EXAM4, RL = (X)
X	//*NET	ID = EXAM4, RL = (Y), HC = 1
Y	//*NET	ID = EXAM4, RL = (Z), HC = 2, AB = F
Z	//*NET	ID = EXAM4, HC = 1, AB = F

## How to code EXAM3:

Jobname	Control	Statements
		(using short form of parameters)
A	//*NET	ID = EXAM3,RL = (C)
B	//*NET	ID = EXAM3,RL = (C,D)
C	//*NET	ID = EXAM3,RL = (E),HC = 2
D	//*NET	ID = EXAM3,RL = (E,I),HC = 1
E	//*NET	ID = EXAM3,RL = (F,H),HC = 2,RS = 1
F	//*NET	ID = EXAM3,RL = (G),HC = 1
G	//*NET	ID = EXAM3,HC = 1,NC = F
H	//*NET	ID = EXAM3,HC = 1,AB = F
I	//*NET	ID = EXAM3,RL = (J),HC = 1,NR = (EXAM4,Y)
J	//*NET	ID = EXAM3,HC = 1

## JES3 Spool Partitioning

When the system reads a job, it initially places the job on a spool volume or volumes. An installation can divide its spool volumes into groups, known as partitions. Depending on how your installation defines its partitions, you can make the system allocate all the spool data for a particular job or all the spool data of a particular type, such as input, output, etc., to the spool volume or volumes in a specified spool partition. Thus, you can prevent JES3 from spreading a job's spool data sets across all spool volumes.

See *JES3 SPL: Initialization and Tuning* for details on how the installation initializes the spool partitions and how JES3 allocates a job's data sets to the partitions.

The following examples illustrate how to guide JES3's use of spool partitions during job execution.

```
//ONE JOB
//*MAIN
//STEP1 EXEC
//OUT1 DD SYSOUT=N
//OUT2 DD SYSOUT=S
```

No SPART parameter is specified on the MAIN statement. Therefore, this job's input spool data sets are allocated to the default spool partition (PARTA). Because there is no spool partition specified for SYSOUT=N, these output spool data sets are allocated to the default spool partition (PARTA). However, if this job executes on the processor named SY2, output spool data sets for SYSOUT=N are allocated to spool partition PARTC as specified on the JES3 MAINPROC initialization statement associated with the processor named SY2. Output spool data sets for SYSOUT=S are allocated to spool partition PARTD as specified on the SYSOUT initialization statement associated with the class name S.

---

```
//TWO JOB
//*MAIN CLASS=IMSBATCH
//STEP1 EXEC
//OUT1 DD SYSOUT=N
//OUT2 DD SYSOUT=S
```

No SPART parameter is specified on the MAIN statement. However, because a class is specified on the MAIN statement, this job's input spool data sets are allocated to the spool partition specified on the CLASS initialization statement associated with IMSBATCH (PARTB). Because there is no spool partition specified for SYSOUT=N, these output spool data sets are allocated to the spool partition specified on the CLASS initialization statement associated with IMSBATCH (PARTB). Output spool data sets for SYSOUT=S are allocated to spool partition PARTD as specified on the SYSOUT initialization statement associated with the class name S.

---

```
//THREE JOB
//*MAIN CLASS=IMSBATCH, SPART=PARTE
//STEP1 EXEC
//OUT DD SYSOUT=N
//OUT2 DD SYSOUT=S
```

This job's input spool data sets are allocated to the spool partition specified by the SPART parameter on the MAIN statement (PARTE) overriding the partition defined by the JES3 CLASS initialization statement. Because there is no spool partition specified for SYSOUT=N, these output spool data sets are allocated to the spool partition specified by the SPART parameter on the MAIN statement (PARTE). Output spool data sets for SYSOUT=S are allocated to spool partition PARTD as specified on the SYSOUT initialization statement associated with the class name S.

## Chapter 4. Guide to Step Control

The first JCL statement of each step in your job is the EXEC statement. The EXEC statement is also the first statement of each procedure step in a cataloged procedure. The format of the EXEC statement is:

```
//stepname EXEC parameters comments
```

Following the EXEC statement in the input stream are any DD statements and data that pertain to the step.

The principal functions of the EXEC statement are to:

- Identify the program the system is to execute.
- Identify the cataloged procedure the system is to use.

For example, Figure 4-1 shows the input deck for a three-step job.

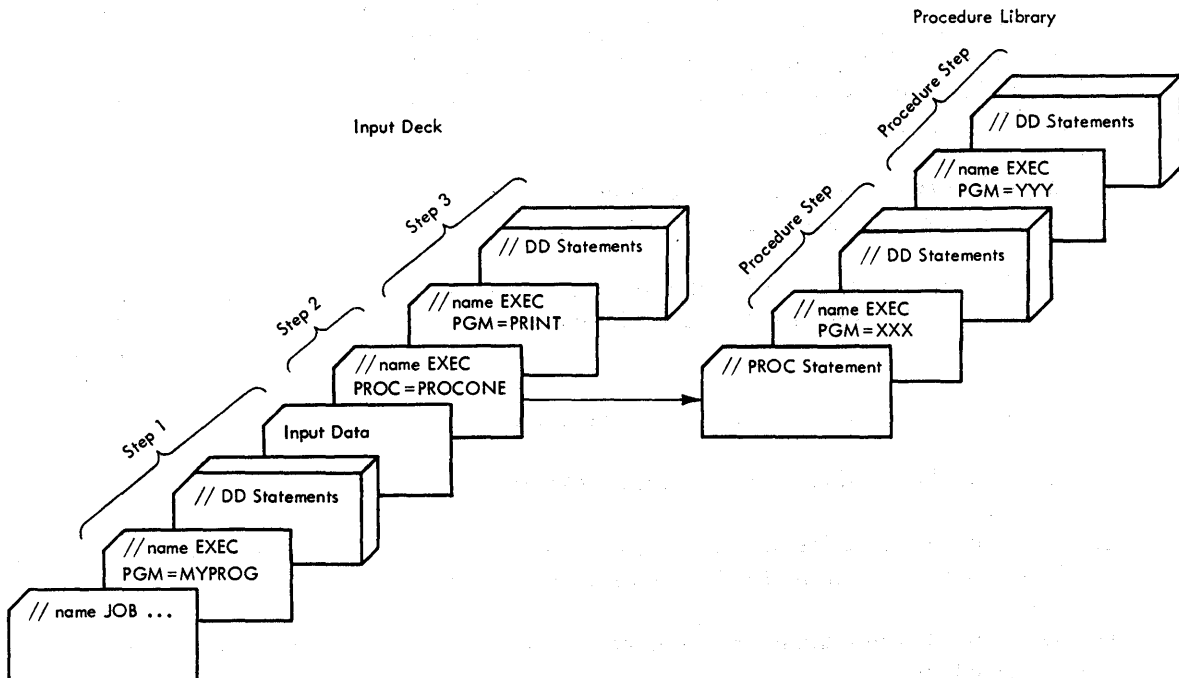


Figure 4-1. Using the EXEC Statement

- The EXEC statement of the first step requests a program named MYPROG. The DD statements and data that MYPROG requires follow the EXEC statement.
- The EXEC statement of the second step requests a cataloged procedure named PROCONE. When the system executes the second step, it uses PROCONE.
- PROCONE has two procedure steps.
  - The EXEC statement of the first procedure step requests a program named XXX.
  - The EXEC statement of the second procedure step requests a program named YYY.

After the system uses the cataloged procedure, it will execute the third step of the job.

- The EXEC statement of the third step requests a program named PRINT. The DD statements required by PRINT follow the third EXEC statement.

Note that you only supply the JCL statements for your job. The cataloged procedure already exists in the procedure library (SYS1.PROCLIB). Therefore, you do not have to code the JCL statements for the cataloged procedure unless you are actually writing the cataloged procedure to place it in the procedure library. You can, however, modify cataloged procedure statements by placing the corrections in the input deck for your job. The methods for modifying cataloged procedures are described in Chapter 9, "Guide to Cataloged and In-Stream Procedures."

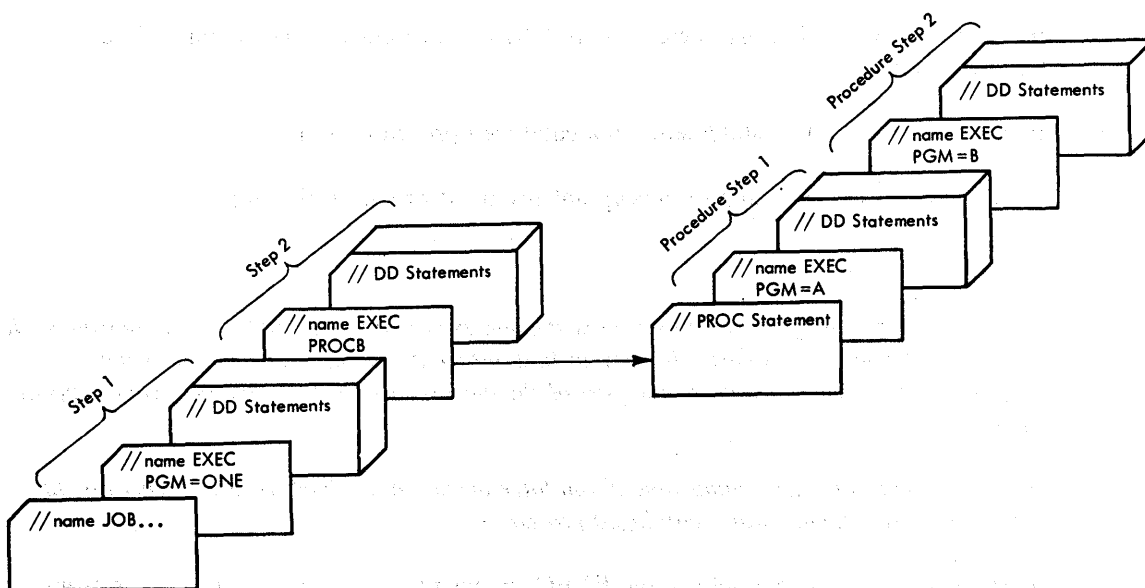


Figure 4-2. Modifying a Cataloged Procedure

The EXEC statement must contain the word EXEC in its operation field. The stepname (name field) and most parameters in the operand field are optional. The only required information in the operand field is either:

- The name of the program the system executes, or
- The name of the procedure the system invokes

The parameters on the EXEC statement that are not related to parameters on the JOB statement allow you to specify the following types of information:

- Processing program information
- Passing information to the program in execution
- Installation management information
- Dynamically allocating and deallocating data sets

The following paragraphs discuss the stepname and the optional parameters of the EXEC statement. For a discussion of EXEC statement parameters that are influenced by JOB statement parameters, see Chapter 5, "Guide to Job and Step Control."

## Naming a Job Step

The stepname identifies a step within a job. The stepname is optional. However, when you want to perform certain functions, you need a valid and unique stepname in the name field for each job step within the job. You **must** specify a stepname if:

- Later JCL statements refer to the step. See "Backward References" on page 2-13.
- The step is part of a cataloged procedure.
- You are going to override parameters on an EXEC statement or DD statement in a cataloged procedure step.
- You are going to add DD statements to a cataloged procedure step.
- You are going to perform step or checkpoint restart at or within the step.

*Notes:*

1. *Name each step in your job; the system uses stepnames in many operating system messages. If you supply a stepname, it is easier for you to find out what part of your job causes the messages. (If the step is unnamed, the part of the message where the stepname would appear is left blank.)*
2. *Using a stepname can save coding time if you later decide to use backward references or to turn your JCL statements into a cataloged procedure.*

Code the stepname in the name field of the EXEC statement. The stepname can range from one to eight characters in length and can contain any alphanumeric characters. The first character of the name must be an alphabetic or national character and must begin in column 3. Each stepname in a job or procedure must be unique.

The following are examples of stepnames in several EXEC statements:

```
//STEP1          EXEC...
//CHECK         EXEC...
//A$9          EXEC...
//LINKEDIT     EXEC...
```

## Processing Program Information

Use the PGM parameter to identify the program the system is to execute. Use the PROC parameter to identify the cataloged procedure the system is to invoke. You can also identify the cataloged procedure by name only as the first parameter in the operand field of the EXEC statement.

The PGM parameter and the PROC parameter (or procedure-name) are mutually exclusive. That is, you can code **only one** of them on an EXEC statement. However, you **must** code **one** of them in the EXEC statement.

Your manager or supervisor should give you the names of the programs and cataloged procedures available in your installation.

### Selecting a Processing Program

You must use the PGM parameter to indicate which processing program the system is to use for this job step and where this program resides. Processing programs can reside in three types of libraries (partitioned data sets):

- A system library.
- A private library.
- A temporary library.

You must code the PGM parameter as the *first* parameter in the operand field.

All IBM-supplied processing programs and, probably, the most frequently used programs written by your installation reside in a system library. The format of the PGM parameter for specifying programs that reside in a system library is:

```
PGM=program-name
```

Replace the term "program-name" with the name or alias associated with the program.

Not all programs have an alias. For example, your installation may have several levels of the linkage editor, but only one of them can have a particular alias. Your manager or supervisor should give you a list of the names and aliases of the processing programs in your installation.

### Identifying the Program to be Executed

All executable programs are members of partitioned data sets (libraries). The library that contains the program can be a temporary library or a private library. In order to execute a program contained in these libraries, code the PGM parameter as the first parameter on the EXEC statement.



## Temporary Library

*Temporary libraries* are temporary partitioned data sets that the system creates to store a program until it is used in a later job step *of the same job*. This type of library is particularly useful for storing the program output (load module) of a linkage editor run until it is executed by a later job step. The program stored in a temporary library is assigned a name by the system. This system-assigned name is not predictable by the programmer. Therefore, you use the PGM parameter to identify the program by location rather than by name. You do this using the backward-reference feature of JCL; see “Backward References” on page 2-13. The format of the PGM parameters for specifying programs that reside in temporary libraries is:

```
PGM=*.stepname.ddname
```

Replace the term “stepname” with the name of the EXEC statement of the job step (of the same job) that creates the temporary library. Replace the term “ddname” with the name of the DD statement that defined the library. The following example shows the EXEC statement for a link edit step, the DD statement in that step that defines the temporary library, and the EXEC statement that requests the execution of the program stored in the temporary library.

```
.  
. .  
. .  
//LINK      EXEC    PGM=IEWL      This EXEC statement requests  
                                     the linkage editor  
//SYSMOD    DD      ...          This DD statement defines  
                                     the temporary library  
. .  
. .  
//GO        EXEC    PGM=*.LINK.SYSMOD
```

When the temporary library is created in a cataloged procedure step (of the same job), you may want to call it in a later job step outside the procedure. In order to call it, you must give both the name of the job step that invokes the procedure and the procedure stepname. For example:

```
//jobstep EXEC PGM=*.stepname.procstepname.ddname.
```

The first part of the following example shows an EXEC statement that calls a cataloged procedure named ASMFCL (ASMFCL is a cataloged procedure that assembles and link edits a source program), and the EXEC statement that requests the execution of the program link edited with the cataloged procedure. The second part of the example shows the EXEC statement of the procedure step in ASMFCL where the temporary library is created, and the DD statement that defines the library.



## The IEFBR14 Program

When this program is called, it gives a return code of 0 and returns to the calling routine. This program causes the system to check the syntax of the control statements, allocate DASD space, and/or satisfy requests for disposition processing of data sets. To perform the preceding functions, substitute IEFBR14 for the program name on the EXEC statement.

For an example of using the IEFBR14 program, see “Using the COND Parameter to Force Step Execution” on page 5-15.

*Note:* When you use the IEFBR14 program to allocate data sets, the system does not perform any data set initialization. Therefore, any attempt to read from the data set will produce unpredictable results. Also, IBM does not recommend the use of the IEFBR14 program to allocate multi-volume data sets.

If you created a data set when using this program, the data set’s status will be old when you execute your program. For situations when a disposition of CATLG will not be satisfied, see “Cataloging a Data Set” on page 7-16.

When you use IEFBR14 to catalog or uncatalog a data set, a message to mount the volume is issued to the operator. If you do not want the volume mounted, code DEFER on the UNIT parameter.

JES will honor any JCL OUTPUT statements in an IEFBR14 program. This is especially important if you wish to give processing options to the system managed data sets. For more information, see the explanation for the JESDS keyword in Chapter 14, “Coding the OUTPUT JCL Statement.”

## Selecting a Cataloged Procedure Library

Instead of executing a particular program, a job step may use a cataloged or in-stream procedure. A cataloged or in-stream procedure can contain JCL statements for several steps, each of which executes a particular program. Cataloged procedures are members of the procedure library. (The IBM-supplied procedure library is named SYS1.PROCLIB; at your installation, there may be additional procedure libraries, which would have different names.)

The format of the PROC parameter is:

PROC=procedure-name
---------------------

Replace the term “procedure-name” with the name of the cataloged procedure or the name on the PROC statement of the in-stream procedure. The procedure-name must be from 1 to 8 alphanumeric characters, the first of which must be alphabetic or national. If you prefer, omit “PROC” and simply code the procedure-name. For example, you can request a cataloged procedure named COBFLG with either of the following EXEC statements:

```
//name EXEC PROC=COBFLG  
//name EXEC COBFLG
```

The PROC parameter or the procedure name must be the *first* parameter in the operand field.

Notes:

1. You can use subsequent parameters in the operand field to override EXEC statement parameters in the cataloged procedure.
2. For details on using and modifying cataloged or in-stream procedures, see Chapter 9, "Guide to Cataloged and In-Stream Procedures."

### In a JES2 System

Code the PROCLIB parameter on the JES2 JOBPARM control statement to choose which of the installation-specified cataloged procedure libraries JES2 uses for resolving catalog procedure references in the JCL. If this parameter is omitted, JES2 uses a cataloged procedure library associated with your job's class.

For more information on coding the JOBPARM statement, see "/\*JOBPARM Statement" on page 16-4.

### In a JES3 System

Code the PROC parameter on the JES3 MAIN control statement to choose which of the installation-specified cataloged procedure libraries JES3 uses for resolving catalog procedure references in the JCL. If this parameter is omitted, JES3 uses the installation standard library, denoted by ST.

If you want to update a cataloged procedure library, code the UPDATE parameter on the JES3 MAIN statement specifying the library to be updated by that job. You can code the UPDATE parameter whether or not JES3 used that library to resolve the job's library references. JES3 will effectively disable the use of that library, preventing all jobs that request it from entering the converter interpreter phase until the updating job terminates. This prevents the use of the library while the update occurs.

*Note:* The library update facility cannot be used for concatenated data sets.

For more information on coding the MAIN statement, see "/\*MAIN Statement" on page 17-23.

## Passing Information to the Program in Execution

Some information required by a program can vary from application to application, such as module attributes and options required by the compiler, assembler, and linkage editor programs. In order to provide this information to the program at the time it is executed, code the PARM parameter on the EXEC statement. The program must include instructions that can retrieve this information. (The exact location and format of the information passed to a processing program are included in *Supervisor Services and Macro Instructions*.)

The PARM parameter can also be coded on the EXEC statement of a cataloged or in-stream procedure step. This establishes fields in which you can pass information to the job. By coding the PARM parameter on the EXEC statement of the job calling a cataloged or in-stream procedure, you can override, add, or nullify parameters in the procedure, or define symbolic parameters. For more information on the PARM parameter, see Chapter 9, "Guide to Cataloged and In-Stream Procedures."

## PARM Parameter

The PARM parameter enables you to pass variable information to a program in execution. Additionally, some IBM-supplied processing programs allow you to select alternatives from a set of options. The PARM values are listed in the publication associated with the program you are using.

In many cases, default values can be selected for PARM values during system generation. That is, the system programmer will select one alternative or assign a fixed value to another. The system assumes this default option unless you specify the other alternative or change the fixed value.

Your manager or supervisor will tell you which default values were generated for the installation's operating system.

The format of the PARM parameter is:

```
PARM=information
```

Replace the term "information" with up to 100 characters of data. The following are general rules for coding the PARM parameter:

- If the information contains more than one expression separated by commas, you must enclose the information in either apostrophes or parentheses. For example:

```
PARM=(DECK,LIST,NOMAP)
PARM='DECK,LIST,NOMAP'
```

- If any of the expressions contain special characters, you can:

- Enclose that expression in apostrophes and the value in parentheses. For example:

```
PARM=(DECK,'NAME=FIRST',LIST)
PARM=(P1,167,'P*AA')
```

- Enclose the entire value in apostrophes. For Example:

```
PARM='DECK,NAME=FIRST,LIST'
PARM='P1,167,P*AA'
```

(The enclosing apostrophes and parentheses are not considered part of the information and do not count towards the maximum of 100 characters of data; commas within apostrophes are passed as part of the information.)

- If the special character is an apostrophe, code it as two consecutive apostrophes. For example, show NAT'L as:

```
PARM='NAT''L'
```

When you code two apostrophes, the system passes only one to the processing program.

- If the special character is an ampersand and you are not defining a symbolic parameter, code the ampersand as two consecutive ampersands. For example, show ABC&D as:

```
PARM='ABC&&D'
```

When you code two ampersands, the system passes only one to the processing program.

- If there is only one value and the value does not contain special characters, you need not enclose the value in parentheses or apostrophes. If the value contains special characters, enclose the value in apostrophes. For example:

```
PARM=LIST
PARM='L.24'
PARM='NAT' 'L'
```

Since the PARM value can consist of up to 100 characters, you may have to continue the value onto another statement. If you must continue the value, enclose it in parentheses. You cannot continue any value enclosed in apostrophes. To continue the value, follow the continuation conventions as outlined in the topic “Continuing Control Statements” in the section “Coding Conventions.” The continuation comma is considered part of the value field and counts towards the maximum of 100 characters of data. The following is an example of continuing the value onto another card.

```
//name EXEC      . . . , PARM=(DECK,LIST, 'LINECNT=80' ,
//              NOMAP)
```

When the job step uses a cataloged procedure, you can pass information to a step in the procedure by including the procedure stepname as part of the keyword PARM. The format is:

```
PARM[.procstepname]=value
```

This specification overrides the PARM parameter in the named procedure step, if one is present. You can code as many parameters of this form as there are steps in the cataloged procedure. For example, the following EXEC statement passes PARM information to two procedure steps of a cataloged procedure named PROCTWO. The names of the two procedure steps are STEP1 and STEP2.

```
//A1 EXEC  PROC=PROCTWO, PARM.STEP1=(ONE, 'TWO=B' ,
//        THREE) , PARM.STEP2=(FOUR, FIVE, SIX,
//        'SEVEN=G')
```

To pass information to the first step in a cataloged procedure and nullify **all** other PARM parameters in the procedure, code the PARM keyword without a procedure stepname. For example:

```
//A1 EXEC  PROCTWO, PARM=(ONE, 'TWO=B', EIGHT)
```

## Installation Management Information: The ACCT Parameter

Some installations have job step accounting routines in addition to the regular job accounting routines. Job step accounting is particularly useful in cases where a different programmer is assigned to write each step of a job, or where the installation's management wants to know how much time is spent on different functions such as compilation or link editing.

You can specify job step accounting information instead of, or in addition to, job accounting information. Specify job step accounting information with the ACCT parameter on the EXEC statement. Specify job accounting information with the accounting information positional parameter in the JOB statement; see Chapter 10, “Coding the JOB Statement.”

## ACCT Parameter

The ACCT parameter allows you to supply job step accounting information. It has the following format:

```
ACCT[.procstepname]=(accounting-information)
```

Replace the term “accounting-information” with one or more subparameters separated by commas. Your manager or supervisor should tell you exactly how to code this parameter. The following are general rules for coding the accounting information:

- The total number of characters of accounting information, plus the commas that separate the subparameters, cannot exceed 142.
- If the list contains only one subparameter, you need not enclose it in parentheses. For example:

```
ACCT=12345
```

- If any subparameter contains special characters (except hyphens), you can:
  - Enclose the subparameter in apostrophes and the value in parentheses. For example:

```
ACCT=(12345,'T/24')
```

- Enclose the entire value in apostrophes.

```
ACCT='12345,T/24'
```

Note that the system does not consider the apostrophes as part of the information.

- If the special character is an apostrophe, it must be shown as two consecutive apostrophes. For example, show O'DONNELL as:

```
ACCT=(12345,'O'DONNELL') or ACCT='12345,O'DONNELL'
```

When the job step uses a cataloged procedure, you can supply accounting information pertaining to a single procedure step by including, as part of the keyword ACCT, the procedure stepname, i.e., ACCT.procstepname. This specification overrides the ACCT parameter in the named procedure step, if one is present. You can code as many parameters of this form as there are steps in the cataloged procedure. For example, the following EXEC statement passes job step accounting information to two procedure steps of a cataloged procedure named PROC3. The name of the two procedure steps are ONE and TWO.

```
//BBB EXEC PROC3,ACCT.ONE=(COMPILE,'J.JONES',  
// '2/04/82'),ACCT.TWO=(LKED,'J.JONES','2/04/82')
```

To supply accounting information pertaining to **all** steps in a procedure, code the ACCT parameter without a procedure stepname. This specification overrides all ACCT parameters in the procedure, if any are present. For example:

```
//BBB EXEC PROC=PROC3,ACCT=('T.JONES','5/20/69')
```

## Dynamically Allocating and Deallocating Data Sets

Dynamic allocation allows you to acquire resources as they are needed. One reason to use dynamic allocation is that you may not know all of the device requirements for a job prior to execution. Another reason is that it allows the system to use resources more efficiently; that is, the system can acquire resources just before their use and/or release them immediately after use. (Resources, as used here, refer to a ddname-data set combination with its associated volumes and devices, if any.) The DYNAM DD statement parameter and DYNAMNBR EXEC statement parameter indicate the number of dynamic allocations to be held in anticipation of reuse. The system uses these indicators to establish a control limit for tracking resources that it is holding in anticipation of reuse.

Use the DYNAMNBR parameter on the EXEC statement to replace the DD DYNAM statements you would have to code. The format of the DYNAMNBR parameter is:

```
//stepname EXEC PGM=program-name,DYNAMNBR=n
```

Where n is the number of DD DYNAM statements you would otherwise have to code.

When you code the DYNAMNBR parameter and DD statements, the system uses the sum of the number of DD statements and the DYNAMNBR value to determine the limit of resources it is to hold in anticipation of reuse.

You can dynamically deallocate resources during the execution of a job step (at the time the data set is closed) by coding the FREE=CLOSE parameter on a DD statement.

There are some circumstances when you should not code the FREE parameter.

- The data set name is referenced in a subsequent step.
- The data set name is referenced in another DD statement in the same step.

Do not use the FREE parameter for a data set if a subsequent DD statement requests unit affinity to this DD statement. For example, do **not** code the following:

```
//DD1 DD DSN=dsname,DISP=OLD,UNIT=TAPE,VOL=SER=111111,FREE=CLOSE
//DD3DD DISP=(,KEEP),DSN=dsname2,UNIT=AFF=DD1
```

For more information on coding the FREE parameter, see "FREE Parameter" on page 12-84. If you do dynamically deallocate a resource at close time, it cannot be reopened in the same step. If you do not want to dynamically deallocate the resource, either specify nothing or specify FREE=END to let the system deallocate the resources at the end of the job step.

For more information on how to use dynamic allocation and deallocation and the control limit, see *SPL: Job Management*.



## Example of Dynamically Deallocating Data Sets

```
//PROS   JOB   CLASS=A,MSGLEVEL=(2,0),PERFORM=70
//STEP1  EXEC  PGM=TEST,DYNAMNBR=4,PARM=(P3,123,MT5)
//OUT1   DD   SYSOUT=C,FREE=CLOSE
//OUT2   DD   SYSOUT=A
//SYSIN  DD   *
.
.
.
data
.
.
.
/*
```

1. The JOB statement specifies that this job will be processed in class A in performance group 70. Only JCL statements will be printed.
2. The control limit is **the sum of the number of DD statements coded and the value coded in the DYNAMNBR parameter**; in this case, seven. If this control limit is exceeded and a request for another dynamic allocation is made, the request is not honored unless resources can be deallocated so that the control value is not exceeded.
3. When OUT1 is closed, it is immediately ready for printing.



## Chapter 5. Guide to Job and Step Control

This chapter discusses job and step control using parameters on the JOB statement, the EXEC statement, and the JES control statements. The discussion covers the following:

- Scheduling a job
- Selecting a processor
- Conditionally executing job steps
- Limiting job and step execution time
- Controlling job queuing through job classes and priorities
- Requesting storage for execution
- Restarting a job at a step or checkpoint

### Scheduling a Job

The scheduling of jobs depends on the job entry subsystem in use at your installation. The following paragraphs discuss job scheduling as it pertains to JES2 or JES3.

#### In a JES2 System

JES2 controls the selection of jobs for processing. As JES2 reads a job into the system, it places JCL statements and any input stream data in respective logical data sets. The system checks the JCL and JES2 statements for syntax errors and issues appropriate error messages. If the JCL statements are syntactically correct, JES2 places the job on an execution queue. The execution queue consists of job class queues; JES2 places jobs within each job class queue according to their priority. Installation programmers assign JES2 initiators to process job classes. The initiator selects jobs from the first class assigned to it according to the priority of the jobs until no more jobs exist in that class; it then selects jobs from the next class assigned.

Use the CLASS parameter on the JOB statement and the JES2 PRIORITY control statement to influence how a job is placed on the execution queue. The format of the CLASS parameter is:

```
CLASS=jobclass
```

For information on coding the CLASS parameter, see "CLASS Parameter" on page 10-9.

The format of the JES2 PRIORITY control statement is:

```
/*PRIORITY p
```

For information on coding the **PRIORITY** control statement, see “/\***PRIORITY** Statement” on page 16-22.

Note that in a multi-access spool environment, more than one JES2 system will be altering the queues. Due to conversion and processor timings, jobs of the same class and priority may be queued for execution out of their reader sequence. Therefore, to insure that one job is selected before another or that the desired volumes are mounted before a job is executed, delay the job's selection by coding **TYPRUN=HOLD** on the **JOB** statement, by coding a job class that will force **TYPRUN=HOLD**, or by coding a **SETUP** control statement.

For more information on the **TYPRUN** parameter and the **SETUP** statement, see “Special Job Processing” on page 3-23.

MVS includes support for controlling the processing rate of jobs and job steps. The installation defines a certain number of performance group definitions. Each of these defines a particular processing rate formula to use for associated jobs or job steps. To associate a job or job step with performance group definitions, code the **PERFORM** parameter on either the **JOB** or **EXEC** statements. The format of the **PERFORM** parameter on either the **JOB** or **EXEC** statement is:

<code>PERFORM=n</code>
------------------------

For information on coding the **PERFORM** parameter on the **JOB** statement, see “**PERFORM** Parameter” on page 10-22. For information on coding the **PERFORM** parameter on the **EXEC** statement, see “**PERFORM** Parameter” on page 11-19.

- If you specify **PERFORM** on the **JOB** statement, its value supersedes any **PERFORM** specifications on **EXEC** statements associated with the job.

### In a JES3 System

JES3 controls the selection of jobs for processing. When JES3 reads a job into the system, it initially places the job on a spooling volume. The system checks the **JCL** and **JES3** statements for syntax errors. If no errors are present, JES3 determines allocation requirements for the job. JES3 device selection takes place next.

JES3 selects devices based on the requirements for JES3-managed devices established in the **JCL**. JES3 requests that the operator mount any necessary volumes. More information on this subject is given in “Allocating Data Resources in a JES3 System” on page 6-4. Once all JES3-managed devices are selected and the first volume on each device is mounted (unless deferred mounting is requested or high watermark setup is used), JES3 places the job in the queue for execution. (High watermark setup allocates a minimum number of devices to run a job.)

When **JCL** or **JES3** statements have syntax errors, appropriate error messages are issued and the job is terminated. When the job has JES3 allocation errors, error messages are issued and JES3 bypasses execution. If the error is due to the operator mounting the wrong volume, JES3 issues a message requesting the correct volume.

The execution queue is logically divided into groups of job classes specified by the installation; within each job class group, jobs are placed according to their job priority. Normally, JES3 places jobs in the same job class group with the same job priority in the execution queue in the order they are read into the system. Deadline scheduling or operator intervention can cause

some jobs to move ahead of other jobs regardless of the order in which JES3 read the jobs into the system. The various job class groups are assigned priorities by the installation. JES3 starts system initiators on each processor and assigns them a job class group to process based on the installation priorities. It selects jobs from any class assigned to it. Jobs are selected by job class, processor eligibility, workload balancing, and priority order as described in the topics: "Assigning a Priority to a Job in JES3" on page 5-20, "Establishing job processing balance in JES3" on page 5-19, and "Assigning a Job to a Job Class in JES3" on page 5-19.

## Selecting a Processor in JES2

In a JES2 multi-access spool configuration, jobs enter the common queue from any input source (local, remote, or another node). If JES2 is not directed to take special actions, the jobs are eligible to execute in any system in the configuration and are selected by priority and the class of initiators, as in a single-system operation.

In a multi-access spool configuration, the JES2 job queue entries contain a system affinity for up to seven systems and may contain an independent mode affinity.

System affinity is useful for special processing requirements (for example, emulation) not available on all systems in the configuration. Independent mode is useful for testing new components with selected jobs while in a shared configuration.

You assign, to a job, affinity to one or more systems (less than the total configuration) and affinity for independent mode by using the SYSAFF parameter on the JES2 JOBPARM control statement. The format of the SYSAFF parameter is:

```
SYSAFF={*|ANY|cccc}[,IND]
```

A SYSAFF specification overrides any installation defined input device default.

You specify a specific system in the JES2 multi-access spool configuration by coding:

```
/*JOBPARM SYSAFF=cccc
```

Where cccc is the system-id of the system, in the JES2 multi-access spool configuration, that is to convert and process your job. The output for the job is eligible for processing by any member of the multi-access spool configuration.

To specify that more than one system can convert and process your job, code:

```
/*JOBPARM SYSAFF=cccc,cccc
```

You can repeat cccc up to the number of processors in the configuration.

To specify that a processor operating in independent mode is eligible to convert and process your job, code:

```
/*JOBPARM SYSAFF=cccc,IND
```

Where IND indicates that the system cccc, operating in independent mode, is to convert and process your job. This same system is to process the job's output.

When you specify that a job has affinity to a specific system (or systems) or to independent mode, only the system you specify can select the job for conversion and processing. The system selects the job only if the mode of the system (independent or not) matches that of the job.

If you submit a job with the NOTIFY parameter specified on the JOB statement or the job includes a JES2 NOTIFY statement, then the mode of the job (independent or not) must match that of the system at which the job is submitted. That is, for TSO submitted jobs, you cannot change the system affinity using the SYSAFF parameter.

For more information on the JES2 multi-access spool configuration, see *SPL: JES2 Initialization and Tuning*. For more information on coding the SYSAFF parameter, see “/\*JOBPARM Statement” on page 16-4.

## Selecting a Processor in JES3

JES3 automatically selects a processor for a job based on device, volume, and data set dependencies known to it. However, if any of the dependencies are not known to JES3, the job can be processed incorrectly or can fail. The section, “Allocating Data Resources in a JES3 System” on page 6-4, discusses these dependencies in more detail. There can also be processor dependencies: a special system feature such as an emulator, a nonstandard catalog, or a system-managed device that JES3 will not recognize unless you define which processor is required using the SYSTEM parameter on the JES3 MAIN statement. The format of the SYSTEM parameter is:

```
SYSTEM={ANY|JGLOBAL|JLOCAL|[ / ](main-name[,main-name]...)}  
}
```

The subparameters of the SYSTEM parameter, JGLOBAL and JLOCAL, request the global or a local MVS processor. To specify particular processors or exclude particular processors, code the main-name value on the MAIN SYSTEM parameter for each processor.

Not all classes are eligible to run on all processors; therefore, make sure that the job class for the job is eligible before selecting a specific processor.

JES3 flushes a job if it specifies a job class (on the JOB or MAIN statements) and a specific processor(s) (on the SYSTEM and TYPE subparameters on the MAIN statement) that are incompatible. A processor(s) is defined for each valid job class on the JES3 CLASS initialization statement during JES3 initialization. For example, if a job specifies CLASS=C and SYSTEM=SY1, then the processor SY1 must have been defined on the CLASS initialization statement for class C.

If you do not specify the SYSTEM parameter, or if you omit the MAIN statement, the job is eligible to run on those processors for which its class is eligible.

If any DD statement in the job contains a device address in the UNIT parameter and that device is either JES3-managed or jointly-managed (JES3/MVS), you must use the SYSTEM or TYPE parameters to restrict job eligibility to the processor that has a path to that device.

## Conditionally Executing Job Steps

Depending on the results of one step of a job, you may not wish to execute subsequent steps. For example, if a compilation fails, you would not want to waste computing time attempting subsequent link-editing or execution steps. You can specify tests to determine whether to bypass or execute job steps, based on the results from previous steps. To do this code the COND parameter on either a JOB or EXEC statement.

Programs indicate the results of a job step in a *return code*. Return codes range from 0 to 4095. You can code the COND parameter to test the return codes that the compiler, assembler, and linkage editor programs issue.

Some return codes are standard for certain programs; for example, a return code of 8 issued by a compiler or linkage editor indicates that serious errors were found and execution is likely to fail. In problem programs, assign a number as the return code to signify a certain condition. For example, if STEP1 of a job reads accounts that are processed in subsequent job steps, you might set a return code of 10 if no delinquent accounts are found. Before STEP3 executes to process delinquent accounts, test the return code from STEP1; if the return code from STEP1 is 10 - there are no delinquent accounts - you can skip STEP3. Specify the test to check the return code from STEP1 by coding the COND parameter.

*Note:* When JES3 determines the setup requirements for any given job, it does so without regard to any COND parameters that may be specified on the EXEC statements. All jobs are processed as though each step will be executed. This is necessary because setup requirements are determined in advance of job execution. The JES3 interpreter DSP (dynamic support program) has no way of predicting whether any given step will or will not execute, or what return code the program will produce.

### Specifying Return Code Tests on the JOB Statement

In the COND parameter, specify tests to determine if the system should bypass a job step. When you code the COND parameter on a JOB statement, and if the system determines that a comparison is true, it bypasses all remaining job steps.

For example, if you code COND=((10,GT),(20,LT)) on the JOB statement, you are asking, "Is 10 greater than the return code or is 20 less than the return code?" If either is true, the system skips all remaining job steps.

If any job step return code is 12, neither test is satisfied: no job step is skipped. All the tests you specify must be false if processing is to continue without skipping any job steps. If the return code is 25, the first test is still false, but the second test is satisfied: 20 is less than 25. The system will bypass all remaining job steps, if you code the COND parameter on the JOB statement.

*Note:* If any job step *abnormally* terminates, MVS bypasses all subsequent steps unless you code the COND parameter on the EXEC statement. (See the section "Specifying the COND Parameter on the EXEC Statement.") If you want to restart the same step that terminated abnormally you can use the checkpoint/restart facility of the operating system. (See "Restarting a JOB" in this chapter.)

The format of the COND parameter on the JOB statement is:

```
COND=((code,operator) [, (code,operator)] ...)
```

Replace "code" with any number from 0 to 4095. Replace the term "operator" with one of the following:

GT (greater than)  
GE (greater than or equal to)  
EQ (equal to)  
LT (less than)  
LE (less than or equal to)  
NE (not equal to)

If you coded COND=((50,GE),(60,LT)), it would read "if 50 is greater than or equal to a return code, or 60 is less than a return code, I want the remaining job steps bypassed." In other words, the job continues as long as return codes range from 51 through 60. If you want to make only one return code test, you need not code the outer parentheses. For example, COND=(8,NE). A maximum of eight conditions can be established. For example, if you code:

```
COND=((5,GT),(8,EQ),(12,EQ),(17,EQ),(19,EQ),(21,EQ),(23,LE))
```

Your job will continue *only* if the return codes are: 5, 6, 7, 9, 10, 11, 13, 14, 15, 16, 18, 20, or 22.

The system applies the tests you specify with the COND parameter against the return code, if any, that each step in your job produces. You can best take advantage of this parameter when the return codes of each job step have compatible meanings. For example, a return code of 4 from the COBOL compiler indicates that the source program was compiled and some minor errors were found; the same return code of 4 from the linkage editor indicates that a load module was produced, but an error that may cause failure at execution time has been found. If you want to chance processing even if small errors occur, code COND=(4,LT); that is, the job terminates if the return code of any step is greater than 4. If you only want to continue processing when no errors occur, code COND=(4,LE); that is, the job terminates if the return code of any step is greater than or equal to 4. (All codes greater than 4 indicate major errors for both the COBOL compiler and the linkage editor.)

If the same return code has different meanings in different job steps, or if you want to take different actions according to which job step produced the return code, use the COND parameter of the EXEC statement to set up conditions for individual job steps. Using the COND parameter on both the JOB and EXEC statements allows you to set some conditions that apply to all steps in the job and other conditions that apply only to particular job steps.

#### **Relationship to the COND Specification on the EXEC Statement**

If you code the COND parameter on the JOB statement and on one or more of the job's EXEC statements, and the return code test requested on the JOB statement is satisfied, the job terminates. The job terminates regardless of whether or not the return code test requested on the EXEC statement is satisfied.

If the test requested on the JOB statement is not satisfied and the return code test requested on the EXEC statement is satisfied, the step is bypassed.



If you omit the COND parameter from the JOB statement, no return code tests are performed for the entire job. If you want return codes tested for a given job step, use the COND parameter on the EXEC statement for that job step. If you do not code the COND parameter on either the JOB or the EXEC statements, the system does not perform any return code tests but tries to execute each step in the job.

*Note:* The COND parameter of the EXEC statement is slightly different from the COND parameter of the JOB statement. See the following section, which also contains examples of using the COND parameter in both the JOB and EXEC statements.

## Specifying Return Code Tests on the EXEC Statement

The COND parameter of the EXEC statement lets you:

- Make as many as eight tests on return codes issued by preceding job steps or cataloged procedure steps, which completed normally. If any one of the tests is satisfied, the job step is bypassed.
- Specify that the system execute the job step even if one or more of the preceding job steps abnormally terminates or *only* if one or more of the preceding job steps abnormally terminated.

The system performs the tests you specify with the COND parameter of the EXEC statement in addition to the tests you specify with the COND parameter of the JOB statement. That is, the system first performs the tests in the JOB statement. If any conditions you specify on the JOB statement are met, the job is discontinued regardless of what you specify in the EXEC statements.

Abnormal termination of a job step normally causes the system to bypass subsequent steps and to terminate the job. However, by means of the COND parameter on the EXEC statement, you can specify execution of a job step after one or more preceding job steps have abnormally terminated.

For the system to act on the COND parameter, a job step must abnormally terminate while the program has control. If a job step abnormally terminates during scheduling, due to failures such as JCL errors or inability to allocate space, the system bypasses the remaining job steps, no matter what you specified in any COND parameter.

The format of the COND parameter on the EXEC statement is:

```
COND=( (code,operator[,stepname][.procstepname])  
      [, (code,operator[,stepname][.procstepname]) ]... [,EVEN] )  
      [,ONLY] )
```

You can write the term (code,operator[,stepname][.procstepname] up to eight times, or you can write either EVEN or ONLY *and* the term (code,operator[,stepname][.procstepname]) up to seven times.

Replace the term “code” with any decimal integer from 0 through 4095.

Replace the term "operator" with one of the following:

GT (greater than)  
GE (greater than or equal to)  
EQ (equal to)  
LT (less than)  
LE (less than or equal to)  
NE (not equal to)

Replace the term "stepname" with the name of the previous job step that issues the return code to be tested. If you do not code a "stepname," the test indicated is performed on all preceding steps.

For example, if you write:

```
COND=( (20,GT,STEP1) , (60,EQ,STEP2) )
```

it would read "Bypass this step if 20 is greater than the return code STEP1 issues, or if STEP2 issues a return code of 60."

If you write:

```
COND=( (20,GT,STEP1) , (60,EQ) )
```

it would read "Bypass this step if 20 is greater than the return code STEP1 issues, or if any of the preceding steps issues a return code of 60."

If you want only one test made, omit the outer parentheses. For example:

```
COND=(10,LT) or COND=(15,NE,STEP5)
```

When the return code is issued by a cataloged procedure step, you may want to test it in a later job step outside of the procedure. In order to test it, you must give both the name of the job step that invokes the procedure and the procedure stepname, for example:

```
COND=((code,operator,stepname.procstepname),...).
```

If you write:

```
COND=(7,LT,STEP4.LINK)
```

it would read "Bypass this step if 7 is less than the return code issued by a procedure step named LINK in the cataloged procedure called by an EXEC statement named STEP4." For additional information on using the COND parameter with cataloged procedures, see "Using the COND Parameter within Cataloged Procedures" on page 5-13.

### Using the COND Subparameters EVEN and ONLY

The EVEN subparameter causes the system to execute the step even if one or more of the preceding job steps abnormally terminated. However, if any return code tests specified in this job step are satisfied, the system bypasses this step.

The ONLY subparameter causes the system to execute the step only if one or more of the preceding job steps abnormally terminated. However, if any return code tests specified in this job step are satisfied, the system bypasses the step.

The EVEN and ONLY subparameters are mutually exclusive. You can code whichever subparameter you select in combination with up to 7 return code tests, and the subparameter can appear before, between, or after return code tests, for example:

```
COND=(EVEN,(4,GT,STEP3))
COND=((8,GE,STEP1),(16,GE),ONLY)
COND=((15,GT,STEP4),EVEN,(30,EQ,STEP7))
```

When a job step abnormally terminates, the system scans the COND parameter on the EXEC statement of the next step for the EVEN or ONLY subparameter. If neither is present, the system bypasses the job step. The system then scans the EXEC statement of the next step for the EVEN or ONLY subparameter. If EVEN or ONLY is specified, the system makes any return code tests on all previous steps specified that executed and did not abnormally terminate. The step is bypassed if any test is satisfied, or if any previous job step abnormally terminated because it exceeded the time limit for the job. Otherwise, the job step is executed.

For example, if you write:

```
COND=EVEN
```

it would read, "Execute this step even if one or more of the preceding steps abnormally terminated during execution."

If you write:

```
COND=((10,LT,STEPA),(20,EQ),ONLY)
```

it would read, "Execute this step only if one of the preceding steps terminated abnormally; but bypass it if 10 is less than the return code STEPA issues or if any of the steps that terminated normally issued a return code of 20."

If you write:

```
COND=((10,LT,STEPA),(20,EQ),EVEN)
```

it would read, "Bypass this step if 10 is less than the return code STEPA issues, or if any of the preceding steps issues a return code of 20; otherwise execute this step even if one of the preceding steps terminated abnormally."

If you omit the COND parameter, the system makes no return code tests and bypasses the step if any of the preceding job steps abnormally terminated.

### Examples of using the COND Parameter in a Job

Any tests specified via the COND parameter of the JOB statement take precedence over those specified via EXEC statements. For example, Figure 5-1 on page 5-11 shows an input deck with nine steps and the return codes produced by those steps that executed. The following tests are performed:

- Before STEP2 (STEP1 produced a return code of 6):
  1. Is 10 less than 6? No.
  2. Is the return code 2 or 4? No. Execute STEP2

- Before STEP3 (STEP2 produced a return code of 2):
  1. Is 10 less than 2 or 6? No.
  2. Did one or more of the preceding steps terminate abnormally? No. Bypass STEP3.
  
- Before STEP4:
  1. Is 10 less than 2 or 6? No.
  2. Is 5 greater than 6? No.
  3. Is one of the preceding return codes equal to 2? Yes. Bypass STEP4.
  
- Before STEP5:
  1. Is 10 less than 2 or 6? No. Execute STEP5.
  
- Before STEP6 (STEP5 produced a return code of 9):
  1. Is 10 less than 9, 2, or 6? No.
  2. Is 8 greater than 9? No.
  3. Did one of the preceding steps terminate abnormally? No. Execute STEP6.
  
- Before STEP7 (STEP6 produced a return code of 10):
  1. Is 10 less than 10, 9, 2, or 6? No.
  
  2. Is 4 greater than return code of STEP4? STEP4 was bypassed and did not produce a return code so this test is ignored. Execute STEP7.
  
- Before STEP8 (STEP7 produced a return code of 12):
  1. Is 10 less than 12, 10, 9, 2, or 6? Yes. Bypass STEP8 and STEP9.

//MYJOB	JOB EXEC	A.SMITH,COND=(10,LT)	Return Code
//STEP1	EXEC	PGM=AAA	6
.	.	.	.
//STEP2	EXEC	PGM=BBB,COND=((2,EQ),(4,EQ))	2
.	.	.	.
//STEP3	EXEC	PGM=CCC,COND=ONLY	-
.	.	.	.
//STEP4	EXEC	PGM=DDD,COND=((5,GT,STEP1),(2,EQ))	-
.	.	.	.
//STEP5	EXEC	PGM=EEE	9
.	.	.	.
//STEP6	EXEC	PGM=FFF,COND=((8,GT,STEP5),EVEN)	10
.	.	.	.
//STEP7	EXEC	PGM=GGG,COND=(4,GT,STEP4)	12
.	.	.	.
//STEP8	EXEC	PGM=HHH	-
.	.	.	.
//STEP9	EXEC	PGM=III,COND=ONLY	-

Figure 5-1. Using the COND Parameter

Figure 5-2 on page 5-13 is another example of the use of the COND parameter. This figure shows an input deck with nine steps and the return codes produced by those steps that were executed. The following tests are performed:

- Before STEP2 (STEP1 produced a return code of 4):
  1. Is 5 equal to 4? No.
  2. Is 7 less than 4? No. Execute STEP2.
- Before STEP3 (STEP2 terminated abnormally):
  1. Is 5 equal to 4? No.
  2. Is EVEN or ONLY specified in STEP3? Yes.
  3. Is 20 greater than 4? Yes. Bypass STEP3.
- Before STEP4:
  1. Is 5 equal to 4? No.
  2. Is EVEN or ONLY specified in STEP4? Yes.
  3. Are any of the preceding return codes equal to 3? No. Execute STEP4.
- Before STEP5 (STEP4 produced a return code of 6):
  1. Is 5 equal to 6 or 4? No.
  2. Is 2 less than the return code of STEP3? STEP3 was bypassed and did not produce a return code, so this test is ignored.
  3. Is EVEN or ONLY specified in STEP5? No. Bypass STEP5.
- Before STEP6:
  1. Is 5 equal to 6 or 4? No.
  2. Is EVEN or ONLY specified in STEP6? No. Bypass STEP6.
- Before STEP7:
  1. Is 5 equal to 6 or 4? No.
  2. Is EVEN or ONLY specified in STEP7? Yes.
  3. Is 6 equal to the return code of STEP5? STEP5 was bypassed and did not produce a return code, so this test is ignored. Execute STEP7.
- Before STEP8 (STEP7 produced a return code of 5):
  1. Is 5 equal to 5, 6, or 4? Yes. Bypass STEP8 and STEP9.

//ABC	JOB	12345,COND=(5,EQ)	Return Code
//STEP1	EXEC	PGM=A	4
.			
.			
//STEP2	EXEC	PGM=B,COND=(7,LT)	ABEND
.			
.			
//STEP3	EXEC	PGM=C,COND=((20,GT,STEP1),EVEN)	-
.			
.			
//STEP4	EXEC	PGM=D,COND=((3,EQ),ONLY)	6
.			
.			
//STEP5	EXEC	PGM=E,COND=(2,LT,STEP3)	-
.			
.			
//STEP6	EXEC	PGM=F	-
.			
.			
//STEP7	EXEC	PGM=G,COND=((6,EQ,STEP5),ONLY)	5
.			
.			
//STEP8	EXEC	PGM=H,COND=EVEN	-
.			
.			
//STEP9	EXEC	PGM=I	-

Figure 5-2. Using the COND Parameter within a Failing Step

### Using the COND Parameter within Cataloged Procedures

When the job step uses a cataloged procedure, you can establish return code tests and the EVEN or ONLY subparameter for a procedure step by including, as part of the keyword COND, the procedure stepname. For example:

```
COND.procstepname=condition codes
```

This specification overrides the COND parameter in the named procedure step, if one is present. You can code as many parameters of this form as there are steps in the cataloged procedure. For example, the following EXEC statement passed COND parameters to two procedure steps of a cataloged procedure named PROC4. The name of the two procedure steps are STEP4 and STEP6.

```
//TEST EXEC PROC=PROC4,COND.STEP4=((7,LT,STEP1),
//          (5,EQ),EVEN),COND.STEP6=((2,EQ),
//          (10,GT,STEP4))
```

To establish one set of return code tests and the EVEN or ONLY subparameter for all steps in a procedure, code (on the EXEC statement that invokes the procedure) the COND parameter





You can direct the system to bypass all steps in a procedure by coding the COND parameter without a procedure stepname. In the above example, if you want the system to bypass the entire PROCB cataloged procedure when 10 is less than the return code issued by step EDIT of cataloged procedure PROCA, code:

```
//THREE EXEC PROCB,COND=(10,LT,TWO.EDIT)
```

*Notes:*

1. *When a job step that contains the EVEN or ONLY subparameter references a data set that was to be created or cataloged in a preceding step, the data set (1) will not exist if the step creating it was bypassed, or (2) may be incomplete if the step creating it abnormally terminated.*
2. *It is meaningless to specify the COND parameter for the first step of a job.*

### Using the COND Parameter to Force Step Execution

Normally, when you code the COND parameter on an EXEC statement, you are instructing the system to bypass execution of this step if the return code tests are satisfied.

There might be a job step that you want the system to **execute** (rather than bypass) if the return code tests are satisfied. For example, you might want to execute a step that cleans up data sets if an earlier step executes but issues a return code that indicates there might be a problem that could cause job failure in a later step.

You can use the COND parameter on an EXEC statement to force the system to execute (rather than bypass) a step if any return code tests are satisfied by adding an extra step to your job. To do this,

1. code an additional EXEC statement that executes program IEFBR14.
2. code on this EXEC statement a COND parameter with condition codes that specify the conditions under which you want the system to execute the original step. (These codes will cause the system to bypass this step.)
3. On the step you want to execute, code: COND=(0,EQ,stepname) where stepname is the name of the step that executes program IEFBR14.

The following example illustrates using the COND parameter to instruct the system to **execute** a job step if return code tests are satisfied.

```
//jobname JOB
//STEP1 EXEC
.
//STEP2 EXEC
.
.
//STEP3 EXEC
.
.
//TESTCODE EXEC PGM=IEFBR14,COND=(8,LE)
//CLEANUP EXEC PGM=PROB,COND=(0,EQ,TESTCODE)
.
.
```

1. The COND parameter on job step TESTCODE specifies: if any preceding step issues a return code less than 8, execute this step. (If 8 is less than or equal to the return code issued by any preceding step, bypass this step.)
2. The COND parameter on step CLEANUP specifies: if step TESTCODE issues a return code equal to 0, bypass this step.
3. Step CLEANUP will execute only if step TESTCODE is bypassed because program IEFBR14 always issues a return code of 0 when it executes.
4. If step 1, 2, or 3 issues a return code of 8 or greater, step TESTCODE is bypassed and step CLEANUP will execute.
5. If step 1, 2, or 3 issues a return code of less than 8, step TESTCODE will execute issuing a return code of 0 and the system bypasses step CLEANUP.

### Limiting Job and Job Step Execution Time

The TIME parameter specifies the maximum amount of time a job may use the processor. Two benefits of the TIME parameter are that it allows you to find out through messages how long the job uses the processor (processor time used appears on the output listing), and it helps limit the processor time wasted by a step that goes into a loop. Normally, the system terminates a job that exceeds its time limit.

The format of the TIME parameter on the JOB and EXEC statements is:

```
TIME={1440|([minutes][,seconds])}
```

Code 1440 if the job can use the processor for 24 hours or more, or code 1440 if any of the job's steps should be allowed to remain in a wait state for more than the established time limit.

Coding TIME = 1440 also lifts the restrictions on the amount of time a job step may remain in a wait state. With System Management Facilities, the installation determines this time limit. In this case, a job step remaining in a wait state for more than the established time limit causes termination of the job unless a user-provided exit routine extends the wait-state time limit for that step.

Replace the term "minutes" and "seconds" with the maximum number of minutes and seconds that the step can use the processor. The number of minutes must be less than 1440 (24 hours); the number of seconds must be less than 60. That is, the maximum time you can specify is TIME = (1439,59).

If you code the processor time limit in minutes only, you need not code the parentheses. For example, code twelve minutes as

```
TIME=12
```

If the processor time limit is given in seconds only, you must code both the parentheses and a comma to indicate the absence of minutes. For example, code "thirty seconds" as:

```
TIME=( , 30)
```

Because the processor time-used field is checked at intervals of about 10.5 seconds, the actual amount of time that a job uses the processor can exceed the time specified on the TIME parameter by up to 10.5 seconds.

A job that exceeds the specified limit causes termination of the job unless you use a user exit routine to extend the time.

TIME=0 is not supported for the JOB statement. The results are unpredictable.

If you code TIME=0 on an EXEC statement, the step will fail **after** the unexpired time from the previous step is used up.

If the TIME parameter is coded on the JOB statement with a value other than 1440, the time limit for each step is set to the step time limit (the value coded on the TIME parameter of the EXEC statement or the limit specified by the installation) or the remaining job time, whichever is smaller.

If the TIME parameter is not coded on the JOB statement, each job step is timed individually according to the value coded on the TIME parameter of the EXEC statement or the limit specified by the installation.

The time limit specified for a job or the time remaining for successive steps in a multistep job is converted, by the system, to seconds and then rounded to the nearest unit (1 unit = 1.048576 seconds). Thus a step can begin execution with up to one-half unit more or one-half unit less time than expected. For example, if the time remaining for the job is less than one-half unit, a step will begin execution with zero time, resulting in an abnormal termination.

If you omit the TIME parameter on the JOB statement, there is no processor time limit assigned to the job; however, each job step is still timed. You can specify different processor time limits for each step in the job by coding the TIME parameter on the EXEC statement associated with each step, as described below.

## Using the TIME Parameter for Cataloged Procedures

When the job step uses a cataloged procedure, you can set a processor time limit for a single procedure step by including, as part of the TIME parameter, the procedure stepname. For example:

```
//stepname EXEC PGM=program-name,TIME.procstepname=1440
```

This specification overrides the TIME parameter in the named procedure step, if one is present. You can code as many parameters of this form as there are steps in the cataloged procedure. For example, the following EXEC statement sets a time limit for two procedure steps of a cataloged procedure named PROC5. The name of the procedure steps are ABC and DEF.

```
//AAA EXEC PROC5,TIME.ABC=20,TIME.DEF=(3,40)
```

## Examples of Coding the Time Parameter on JOB and EXEC Statements

```
//FIRST JOB      TIME=2  
//STEP1 EXEC    TIME=1  
//STEP2 EXEC    TIME=1
```

In this example the total job is allowed 2 minutes of execution time and each step is allowed 1 minute. Should either step attempt to execute beyond 1 minute the entire job will terminate beginning with that step.

```
//SECOND JOB     TIME=3  
//STEP1 EXEC    TIME=2  
//STEP2 EXEC    TIME=2
```

In this example the total job is allowed 3 minutes of execution time. Each step is allowed 2 minutes of execution time. Should either step attempt to execute beyond 2 minutes, the entire job will terminate beginning with that step. Should STEP1 execute for 1.74 minutes and STEP2 attempt to execute beyond 1.26 minutes, the job will be terminated because of the 3-minute time limit specified on the JOB statement.

To set a processor time limit for an entire procedure, code the TIME parameter without a procedure stepname. This specification overrides all TIME parameters in the procedure if any are present. For example:

```
//AAA EXEC PROC5,TIME=20
```

### Specifying the TIME parameter on the JES2 JOBPARM Statement

When you code the TIME parameter on a JES2 JOBPARM control statement, you instruct JES2 to issue messages to the operator when the job exceeds the estimated execution time.

## Controlling Job Queuing through Job Classes and Priorities

One of the most important features of the operating system is its ability to balance the job mix by recognizing the classes and priorities assigned to jobs.

There can be up to 36 job classes in your installation (two additional classes are reserved for started tasks and time sharing users). The type of job assigned to each class is arbitrary and should be determined by each installation. For example, some installations may assign a class to each of the following types of jobs:

- I/O-bound jobs.
- Processor-bound jobs.
- Jobs that are being debugged.
- Jobs that use a particular resource. For example, if there are relatively few tape drives in your installation, two programs that use that tape drives should not be multiprogrammed. Therefore, those programs should be assigned to the same job class to avoid their simultaneous selection. Similarly, if there is a data base that programs must access serially, Therefore, those programs should be assigned to the same job class to avoid their simultaneous selection.

In general, all jobs of the same characteristics should be in the same class.

The priority assigned to each job determines the order of execution within each class. There can be up to 15 priorities in each class. The higher the priority (the higher the number), the sooner your job will be executed.

Your manager or supervisor should tell you which class and priority to assign to your job.

## Establishing job processing balance in JES3

The IORATE parameter on the JES3 MAIN statement specifies a value for the job to determine the mix of jobs for each processor. It defines the relationship between processor-bound processing and I/O-bound processing for that job. The I/O rate for a job is expressed as being high, medium, or low. JES3 attempts to provide a balance of processor-bound and I/O-bound jobs to improve the scheduling of jobs for execution.

The IORATE parameter regulates how a job is scheduled. In contrast, the PERFORM parameter on either the JOB or EXEC statement regulates how a job executes. The PERFORM parameter is discussed in "Performance of Jobs and Job Steps in JES3" on page 5-22.

## Assigning a Job to a Job Class in JES2

An installation establishes job classes to group jobs. By assigning jobs to job classes, the installation tries to avoid contention between jobs that require the same resources by preventing them from running concurrently; in short, grouping jobs is an attempt to provide a better mix of jobs for more efficient system use. The installation determines which characteristics are most important in achieving a good balance of jobs in the computing system.

Assign a job to a job class by coding the CLASS parameter on the JOB statement. The format of the CLASS parameter is:

```
CLASS=jobclass
```

Replace the term "jobclass" with a letter from A through Z or a number from 0 through 9. For example, if your job belongs to class C, code

```
CLASS=C
```

If you omit the CLASS parameter the default job class is determined by the input device from which the job entered the system.

## Assigning a Job to a Job Class in JES3

A job class describes the type of job being submitted, that is, production, testing, and so forth. The installation establishes it; the class has no inherent meaning except as the installation has defined it. It is used by the installation for scheduling jobs on eligible parameter on the JOB statement, as discussed above, or the CLASS parameter on the JES3 MAIN control statement. If neither of these parameters is coded, the job will be assigned an installation-defined standard class default.

## Assigning a Priority to a Job for JES2

Within a job class, jobs are selected for execution from the execution queue according to job priority. Jobs with the same class and priority are placed in the execution queue in the execution queue in a first-in first-out order. In most cases, JES2 calculates the job's priority. However, for certain jobs, you or the operator can assign different priorities. Specify job priority by coding a JES2 PRIORITY statement or by coding the PRTY parameter on the JOB statement.

Priority is explicitly stated on a PRIORITY statement. When you do not specify a priority, JES2 uses the estimated number of cards, lines of output, and the time for job execution, according to installation algorithms, to calculate the priority. JES2 also uses these factors to monitor job execution time and output. If you do not code these estimates, JES2 assumes installation defaults. If your job exceeds any of these estimates, JES2 issues warning messages to the operator. In some cases, the installation can specify that the operator cancel the job. For example, an installation might specify that the lower the estimated execution time and output, the higher the priority. This can enforce programmers specifying the correct amounts or the job is canceled. JES2 sets a job's selection priority to 1 following execution if the job's execution priority was 12 or less. If the execution priority was greater than 12, JES2 sets the selection priority to 15.

If you wish to assign a different priority to your job, use the PRTY parameter.

The format of the PRTY parameter on the JOB statement for JES2 is:

```
PRTY=priority
```

Replace the term "priority" with an integer from 0 through 15. The highest priority number is 15.

*Note:* To assign a different priority to a job step, code the DPRTY parameter on the EXEC statement associated with the step, as described in "Assigning a Dispatching Priority to Job Steps" on page 5-21. The priority assigned to the job applies to any step that does not use the DPRTY parameter.

## Assigning a Priority to a Job in JES3

Within a job class group, jobs are selected for execution according to job priority. Jobs with the same priority are placed in a first-in first-out order. Specify job priority by coding the PRTY parameter on the JOB statement.

The format of the PRTY parameter on the JOB statement for JES3 is:

```
PRTY=priority
```

Replace the term "priority" with an integer from 0 through 13. The highest priority number is 13.

*Note:* To assign a different priority to a job step, code the DPRTY parameter on the EXEC statement associated with the step, as described in the next section. The priority assigned to the job applies to any step that does not use the DPRTY parameter.

The operator can change the priority order for jobs by priority aging or by deadline scheduling. How the operator changes priority is discussed in *JES3 Operator's Library*.

Priority aging allows JES3 to increase the priority of a job after JES3 passes over it an installation-specified number of times. A job can be bypassed because of an insufficient number of devices or contention for a volume or data set or because there is not enough main storage on an MVS processor. The installation defines priority aging; if you cannot specify it using JCL.

Deadline scheduling allows you to specify a time of day when the job should be scheduled. If the job is not scheduled by this time, JES3 will increase the priority of the job at installation-defined intervals until it is scheduled. For more information on deadline scheduling, refer to "Deadline Scheduling for JES3" on page 3-27.

In addition to job selection, raising a job's priority will cause the job to be given preferential treatment in JES3 device selection. For more information on JES3 device selection, see "Allocating Data Resources in a JES3 System" on page 6-4.

## Assigning a Dispatching Priority to Job Steps

In most jobs, you will want the job's dispatching priority to default to an automatic priority group (APG) instead of assigning your own dispatching priority. The automatic priority group function is an algorithm that the system resources manager uses to attempt to increase system throughput by dynamically adjusting the dispatching priority of associated address spaces.

If you do assign a dispatching priority, code the DPRTY parameter on the EXEC statement. In the DPRTY parameter, you can code two values. The system substitutes these values in the following formula to form the dispatching priority:

$$(\text{value1} \times 16) + \text{value2} = \text{step's dispatching priority}$$

If you omit the DPRTY parameter completely, the job step is assigned the APG priority. If value1 is omitted or it is equal to the APG value, the step is assigned the APG priority and any value you code for value2 is ignored. In this case, value2 is obtained from the Installation Performance Specification (IPS) using the performance group associated with the job step. (See *SPL: Initialization and Tuning Guide* for information on IPS.) If value2 is not specified in the IPS, a value of 6 is assigned to value2.

## DPRTY Parameter

The format of the DPRTY parameter is:

```
DPRTY=( [value1] [,value2] )
```

Replace both "value1" and "value2" with a number from 0 through 15.

If you do not assign a number to “value1,” a default value of 0 is assumed. If you omit “value1” you must code both the parentheses and a comma preceding “value2” to indicate the absence of “value1.” For example, if you code:

```
DPRTY=( , 5)
```

a value of DPRTY=(0,5) is assumed.

If you omit “value2” you need not code the parentheses. For example, if you code:

```
DPRTY=7
```

If you omit the DPRTY parameter, the job step is assigned the priority you specified for the entire job either with the PRTY parameter of the JOB statement, or by default.

When this step uses a cataloged procedure, you can assign a dispatching priority to a single procedure step by including, as part of the DPRTY parameter, the procedure stepname, that is, DPRTY.procstepname. This specification overrides the DPRTY parameter in the named procedure step, if one is present. You can code as many parameters of this form as there are steps in the cataloged procedure. For example, the following EXEC statement is used to establish a dispatching priority for two procedure steps of a cataloged procedure named PROC6. The names of the procedure steps are UP and DOWN.

```
//STEP9 EXEC PROC6,DPRTY.UP=( , 8),DPRTY.DOWN=( 4, 6)
```

To assign a single dispatching priority to an entire cataloged procedure, code the DPRTY parameter without a procedure stepname. This specification overrides all DPRTY parameters in the procedure, if any are present. For example:

```
//STEP9 EXEC PROC=PROC6,DPRTY=( 5, 9)
```

## Performance of Jobs and Job Steps in JES2

You can associate a job or job step with any one of several performance group definitions. Performance group definitions that the installation supplies describe the workload-dependent processing rate the system should afford to an associated job or job step. Most performance group definitions prescribe good processing rates under light system workload conditions. However, when the system workload is moderate or heavy, some performance group definitions will specify significantly lower processing rates than for other performance groups.

The installation defines the number and definition of performance groups needed to meet the response requirements of its various users and will probably publish this information for your use. Make the performance group association with the job or job step by coding an appropriate performance group number on the PERFORM parameter of the JOB or EXEC statement.

For further information concerning performance, refer to *SPL: Initialization and Tuning Guide* and to *SPL: JES2 Initialization and Tuning*.



## Performance of Jobs and Job Steps in JES3

To regulate the execution performance of a job in JES3, associate a job or job step with a performance group. The installation defines performance groups that determine the rate at which a given job will have access to the processor, storage, and I/O channels. Most performance groups designate good processing rates under light system workload conditions. However, when the system workload is moderate or heavy, some performance groups will have significantly lower processing rates than others. The installation defines the performance groups needed to meet the response requirements of its various users and will probably publish this information for your use. Associate the performance group with a job or job step by coding a performance group number on the PERFORM parameter on the JOB or EXEC statements. The PERFORM parameter regulates how a job executes as contrasted with the MAIN IORATE parameter that regulates how a job is scheduled. The IORATE parameter is described in the section, "Establishing job processing balance in JES3" on page 5-19.

For further information concerning system performance, refer to *SPL: Initialization and Tuning Guide* and to *JES3 SPL: Initialization and Tuning*.

## Requesting Storage for Execution

In MVS, the storage available for a program consists of real storage and virtual storage:

- **Real storage** is the storage from which the central processing unit can directly obtain instructions and data and to which it can directly return results.
- **Virtual storage** is addressable space that appears to the user as real storage, from which instructions and data are mapped into real storage locations. The user address space is 16 million bytes. The user address space consists of the commonly addressable system storage, the nucleus, and the private address space (which includes the user's region).

When a program is selected, it is brought into virtual storage and divided into pages (a page is 4K bytes). The supervisor component of MVS is responsible for transferring pages of a program into real storage for execution. This paging is done automatically by the supervisor; to you, it appears as if the entire program exists in real storage.

Usually you assign a region size through the REGION parameter of the JOB statement, as described below. In this case, each step of the job will be executed in that region. You can, however, specify a different region size for each step in the job using the REGION parameter of the EXEC statement. This is desirable in cases where different steps need a greatly different region size.

## When to Request Real Storage

For most programs, the supervisor transfers pages of a program to real storage as they are required for execution; not all pages of a program are necessarily in real storage at one time and the pages that are in real storage at once do not necessarily occupy contiguous space. Certain programs, however, must have all their pages in contiguous real storage while they are executing; they cannot be paged during execution. Such programs include:

- Programs that modify a channel program while it is active
- Programs that are highly time dependent

These programs must be put into an area of virtual storage called the nonpageable dynamic area, whose virtual addresses are identical to real addresses; they are the only programs for which you should request real storage. If a job or job step must not be paged during execution, identify it by coding `ADDRSPC=REAL` on either the `JOB` or the `EXEC` statement. Request the amount of real storage needed via the `REGION` parameter.

## Specifying Storage Requirements with the `REGION` Parameter

The meaning of the `REGION` parameter differs depending on whether the program can be paged during execution (if `ADDRSPC=VIRT` is coded or implied) or cannot be paged during execution (if `ADDRSPC=REAL` is coded).

### Virtual Storage Requirements

When `ADDRSPC=VIRT` is coded or implied, two values are established internally from either the `REGION` parameter or an installation-supplied default. When `ADDRSPC=REAL` is coded, one value is established internally from either the `REGION` parameter or the installation-supplied default. These internal values are used to limit all `GETMAINs`. (For further information, see *SPL: Job Management, Supervisor Services and Macro Instructions*, and the sections on the `ADDRSPC` and `REGION` parameters in this publication.)

The amount of space requested must include any additional requests the program makes during its execution (for example, a request made with the `GETMAIN` macro instruction). Also, the amount of storage requested must include sufficient space for the task termination function. Task termination invokes certain system resource managers that can issue `GETMAIN` macro instructions for space in the user's region. The region must have enough unallocated storage during task termination to allow the task termination function to complete.

If you do not specify the `REGION` parameter when `ADDRSPC=VIRT` is coded or implied, the installation provides a default region size. That value, or if coded, the value specified on the `REGION` parameter, sets the upper boundary to limit region size for variable-length `GETMAINs`. In addition, an IBM- or installation-supplied routine (`IEALIMIT`) uses the region value to establish a second value, which the system uses to limit fixed-length and variable-length `GETMAINs` when the space remaining in the region is less than the requested minimum. When the minimum requested length for variable-length `GETMAINs` causes this second value to be exceeded, the job or job step abnormally terminates. For further information, see *Supervisor Services and Macro Instructions*.

### Real Storage Requirements

When `ADDRSPC=REAL` is coded, the minimum region size must be 8K if the program to be executed is reenterable and resides in an authorized library. In all other cases the minimum region size must be 12K. Note that this is the minimum region for successful execution, but not necessarily the minimum region size for successful job completion. If you are going to run programs in an `ADDRSPC=REAL` environment, have them perform as much clean-up as possible before terminating.

When a job step uses a cataloged procedure, you can request a region size for a single procedure step by including, as part of the region parameter, the procedure stepname. For example:

```
REGION.procstepname=valueK
```

This specification overrides the REGION parameter in the named procedure step, if one is present. You can code as many parameters of this form as there are steps in the cataloged procedure. For example, the following EXEC statement is used to assign region sizes to two procedure steps of a cataloged procedure named PROC8. The names of the procedure steps are FIRST and SECOND.

```
//STEP EXEC PROC8,REGION.FIRST=750K  
// REGION.SECOND=700K
```

To request a single region size for an entire cataloged procedure, code the REGION parameter without a procedure stepname. This specification overrides all REGION parameters in the procedure, if any are present. For example:

```
//STEP EXEC PROC=PROC8,REGION=650K
```

*Note:* If you have specified a REGION parameter on the JOB statement, REGION parameters on the job's EXEC statements are ignored.

### The REGION Parameter

The REGION parameter allows you to request:

- The maximum amount of storage to be allocated to the job. This figure must include the size of those components that are required by your program and are not resident in storage.

The format of the REGION parameter is:

REGION=valueK
---------------

- Code an even number for valueK. If you code an odd number, the system treats it as the next highest even number.
- When you want to specify a different region size for each job step, code the REGION parameter on the EXEC statements, instead of the JOB statement.
- If you code the REGION parameter on the JOB statement, REGION parameters coded on the job's EXEC statements are ignored.

REGION=0K will get you all the storage available in the private area, that is, from the top of the system region to the bottom of the common service area (CSA). The resulting size of the region is unpredictable.

### Using the JES3 LREGION Parameter to Define Logical Storage

The LREGION parameter of the JES3 *//\*MAIN* statement allows you to specify the approximate size of the largest step's working set in real storage. JES3 uses the LREGION to improve scheduling on an MVS main processor. However, you should consult your system programming staff before using LREGION. LREGION values that are too small can cause a performance degradation. For more information, see the JES3 SELECT initialization statement in the *JES3 SPL: Initialization and Tuning*.

## Example of Requesting Storage

The purpose of this example is to indicate how to request storage for a program when it is important that it not be paged.

```
//JOB      JOB      BROWN,CLASS=D,MSGLEVEL=1
//STEP1   EXEC     PGM=REAL,REGION=20K,ADDRSPC=REAL
//DD1     DD       DSN=DISK1,DISP=OLD
//STEP2   EXEC     PGM=VIRT,REGION=75K,ADDRSPC=VIRT
//DD2     DD       DSN=DISK2,DISP=OLD
```

1. The JOB statement assigns the job to class D and requests the printing of all JCL statements and messages.
2. STEP1 is to be executed in real storage.
3. STEP2 is to be executed in virtual storage.

## Restarting a Job at a Step or Checkpoint

When a job step abnormally terminates, you may have to resubmit the job for execution; this means lost computer time and a delay in obtaining the desired results. The operating system provides restart facilities to reduce the effects of abnormal termination.

If a job step abnormally terminates or if a system failure occurs, the restart facilities allow you to request that the job step be restarted either at the beginning of the step (step restart) or at some point within the step (checkpoint restart). Furthermore, restart can occur automatically after abnormal termination, or it can be deferred until the job is resubmitted. Automatic restarts are specified with the RD parameter, and deferred restarts with the RESTART parameter. For detailed information on the checkpoint/restart facilities, see *Checkpoint/Restart*.

There are two types of restarts:

- **Step restart**, from the beginning of a job step.
- **Checkpoint restart**, from a checkpoint within a job step. You establish checkpoints in a job step by coding the CHKPT macro instruction for each checkpoint. The CHKPT macro is described in *Data Management Macro Instructions*. See also the DD CHKPT parameter. It specifies that checkpoints are to be taken at end of volume for the data set defined by the DD statement on which it is coded.

Whether you use step restart or checkpoint restart, the restart facility can be automatic or deferred.

**Automatic restart:** To use automatic restart, code the RD (restart definition) parameter on the JOB or EXEC statement. If you use this facility, the presence of a job journal is required. (A **job journal** is established at JES2 initialization, or by coding JOURNAL=YES on the `//*MAIN JES3` statement for JES3, program in execution.) When a system failure occurs or a job step abnormally terminates, and you have a job journal, the restart facility allows you to have automatic restart by coding RD=R on the JOB or EXEC statements. If you have taken checkpoints, the system restarts the job at the last checkpoint regardless of whether you have coded the RD parameter.

**For JES2**, when you code `RD=R` or `RD=RNC`, JES2 forces journaling. When a job step abnormally terminates or a system failure occurs while the job is in execution and you do not have a job journal, these jobs are ineligible for automatic restart regardless of whether or not the RD parameter is coded.

**For JES3**, when a job step abnormally terminates or a system failure occurs while the job is in execution and you do not have a job journal, these jobs are ineligible for automatic restart regardless of whether or not the RD parameter is coded.

**Deferred restart:** To use deferred restart, code the `RESTART` parameter on the `JOB` statement. This required parameter specifies a job step or a step of a cataloged procedure and can specify a checkpoint identifier if you are using deferred checkpoint restart. The effect of the parameter is simply to restart the job at the beginning of the specified step or checkpoint. The `SYSCHK DD` statement is required when a job is being submitted for deferred checkpoint restart and must specify explicit `UNIT` and `VOLUME` information if the checkpoint data set is not cataloged.

Refer to *Checkpoint/Restart*. for a complete description of planning for and using the checkpoint restart facility.

## The RD Parameter on the JOB Statement

A job can be automatically restarted at the beginning of the job step that abnormally terminated (step restart) or within the step (checkpoint restart). In either case, *automatic* restart can occur only if all of the following are true:

1. You use the RD parameter to request restart,
2. The completion code returned during abnormal termination indicates that the step is eligible for restart, and
3. The operator authorizes restart.

When you code the RD parameter on the `JOB` statement, the RD parameter applies to all steps of the corresponding job and overrides the RD parameter you code on any `EXEC` statements of the job.

A request for an automatic checkpoint restart (issuing the `CHKPT` macro instruction) overrides a request for automatic step restart (coding `RD=R` on a `JOB` or `EXEC` statement).

Use the `RD` (restart definition) parameter to specify that the operator is to perform automatic step restart if job failure occurs. You can also use the `RD` parameter to suppress, partially or totally, the action of the `CHKPT` macro instruction. The format of the `RD` parameter on the `JOB` statement is:

<code>RD={R RNC NC NR}</code>
-------------------------------

When you do not code the `RD` parameter, the job is eligible for automatic checkpoint restart if your program has requested checkpoints using the `CHKPT` macro instruction. However, the job is not eligible for automatic step restart.

When you code RD=R or RD=RNC on either the JOB or EXEC statements JES2 forces journaling.

## The RESTART Parameter on the JOB Statement

Use the RESTART parameter to perform a deferred restart of a job. You can specify that the system restart at a step (deferred step restart), or within the step at a checkpoint taken with the CHKPT macro instruction (deferred checkpoint restart). The format of the RESTART parameter is:

```
RESTART=({*|stepname|stepname.procstepname}[ ,checkid])
```

### Using the RESTART parameter.

Before resubmitting a job,

- Check all backward references to steps that precede the restart step. Eliminate all backward references for the following keywords:
  - PGM on EXEC statements, and
  - VOLUME = REF = reference on DD statements.
- Carefully review all EXEC statements that contain the COND parameter. If any of the COND parameters contain values that refer to a step preceding the restart step, be aware that the system ignores the COND parameters for these steps.

### Using the RESTART parameter With Generation Data Sets

In the restart step or in steps following it, do not use the original relative generation numbers to refer to generation data sets that were created and cataloged in steps preceding the restart step. Instead, refer to a generation data set by its present relative generation number. For example, if the last generation data set created and cataloged was assigned a generation number of +2, refer to it as 0 in the restart step and in steps following the restart step. If generation data set +1 was also created and cataloged, you would refer to it as -1.

If generation data sets created in the restart step were kept instead of cataloged (that is, DISP=(NEW,CATLG,KEEP) was coded and abnormal termination occurred), refer to generation data sets during checkpoint restart by the same relative generation numbers that you used to create them.

## The RD Parameter on the EXEC Statement

Use the RD (restart definition) parameter to specify how the step restart facilities are used with the CHKPT macro instruction, and whether you want to permit or suppress automatic restart.

For detailed information on the checkpoint/restart facilities, refer to *Checkpoint/Restart*.

The format of the RD parameter on the EXEC statement is:

```
RD[.procstepname]={R|RNC|NC|NR}
```

## Using the RD Parameter

Automatic restart will not be honored if you do not have a job journal. The journal data set in JES3 is used if one of the following exists:

- RD= is specified on the JOB or EXEC statement
- JOURNAL= YES is specified on the JES3 MAIN statement
- JOURNAL= YES is specified on the CLASS initialization statement for this job, and the JES3 MAIN statement did not override it

If you code the RD parameter on the JOB statement, any RD parameters coded on the job's EXEC statements are ignored and the value coded on the JOB statement is effective for all steps.

You can use the RD parameter values NC and RNC to suppress the action of the CHKPT DD parameter.

## The JES2 RESTART Parameter

If your job is executing before a re-IPL and you warm start JES2, and your job cannot restart from a step or checkpoint, code the RESTART parameter on the JES2 JOBPARM control statement.

## The JES3 FAILURE Parameter

The FAILURE parameter on the JES3 MAIN control statement tells JES3 what action to take if a system failure occurs.





## Chapter 6. Guide to Data Allocation Control

This chapter discusses how to direct the system in its allocation of data resources. The discussion covers the following:

- Using JES3 spool partitioning
- Controlling access to RACF-protected data sets
- Dynamically allocating and deallocating data sets
- Allocating data resources in a JES3 system

### Using JES3 Spool Partitioning

When the system reads a job, it initially places the job on a spool volume or volumes. An installation can divide its spool volumes into groups, known as partitions. Depending on how your installation defines its partitions, you can make the system allocate all the spool data for a particular job or all the spool data of a particular type, such as input, output, etc., to the spool volume or volumes in a specified spool partition. Thus, you can prevent JES3 from spreading a job's spool data sets across all spool volumes.

See *JES3 SPL: Initialization and Tuning* for details on how the installation initializes the spool partitions and how JES3 allocates a job's data sets to the partitions.

The following examples illustrate how to guide JES3's use of spool partitions during job execution.

```
//ONE JOB
//*MAIN
//STEP1 EXEC
//OUT1 DD SYSOUT=N
//OUT2 DD SYSOUT=S
```

No SPART parameter is specified on the MAIN statement. Therefore, this job's input spool data sets are allocated to the default spool partition (PARTA). Because there is no spool partition specified for SYSOUT=N, these output spool data sets are allocated to the default spool partition (PARTA). However, if this job executes on the processor named SY2, output spool data sets for SYSOUT=N are allocated to spool partition PARTC as specified on the JES3 MAINPROC initialization statement associated with the processor named SY2. Output spool data sets for SYSOUT=S are allocated to spool partition PARTD as specified on the SYSOUT initialization statement associated with the class name S.

---

```
//TWO JOB
//*MAIN CLASS=IMSBATCH
//STEP1 EXEC
//OUT1 DD SYSOUT=N
//OUT2 DD SYSOUT=S
```

No SPART parameter is specified on the MAIN statement. However, because a class is specified on the MAIN statement, this job's input spool data sets are allocated to the spool partition specified on the CLASS initialization statement associated with IMSBATCH (PARTB). Because there is no spool partition specified for SYSOUT=N, these output spool data sets are allocated to the spool partition specified on the CLASS initialization statement associated with IMSBATCH (PARTB). Output spool data sets for SYSOUT=S are allocated to spool partition PARTD as specified on the SYSOUT initialization statement associated with the class name S.

---

```
//THREE JOB
//*MAIN CLASS=IMSBATCH,SPART=PARTE
//STEP1 EXEC
//OUT DD SYSOUT=N
//OUT2 DD SYSOUT=S
```

This job's input spool data sets are allocated to the spool partition specified by the SPART parameter on the MAIN statement (PARTE) overriding the partition defined by the JES3 CLASS initialization statement. Because there is no spool partition specified for SYSOUT=N, these output spool data sets are allocated to the spool partition specified by the SPART parameter on the MAIN statement (PARTE). Output spool data sets for SYSOUT=S are allocated to spool partition PARTD as specified on the SYSOUT initialization statement associated with the class name S.

## Controlling Access to RACF-Protected Data Sets

The IBM Resource Access Control Facility (RACF) is a program product that helps installations achieve data security by controlling access to data sets. When the data sets a job uses are RACF protected, the USER and PASSWORD parameters may be required on the JOB statement to gain access to protected data sets.

For example:

```
//MINE JOB D58,TOM,USER=userid,PASSWORD=pswd
```

The USER parameter identifies the RACF-defined user and the PASSWORD parameter identifies the user's current password.

Depending on the RACF options an installation has chosen, a user may also be required to specify a RACF group name on the JOB statement in order to access some RACF-protected resources. When an installation determines that this is necessary, you must code the GROUP parameter in addition to the USER and PASSWORD parameters.

For example:

```
//YOURS JOB D58,CHERI,USER=userid,PASSWORD=pswd,GROUP=groupname
```

Depending on the installation's procedures, the USER, PASSWORD, and GROUP parameters may be omitted for jobs submitted through the internal reader by RACF/TSO users or directly by other jobs.

For more information about coding these parameters, see “GROUP Parameter” on page 10-12, “PASSWORD Parameter” on page 10-20, and “USER Parameter” on page 10-41. For more information about RACF, see *Resource Access Control Facility (RACF) Security Administrator’s Guide*.

## Dynamically Allocating and Deallocating Data Sets

Dynamic allocation allows you to acquire resources as they are needed. One reason to use dynamic allocation is that you may not know all of the device requirements for a job prior to execution. Another reason is that it allows the system to use resources more efficiently; that is, the system can acquire resources just before their use and/or release them immediately after use. (Resources, as used here, refer to a ddname-data set combination with its associated volumes and devices, if any.) The DYNAM DD statement parameter and DYNAMNBR EXEC statement parameter indicate the number of dynamic allocations to be held in anticipation of reuse. The system uses these indicators to establish a control limit for tracking resources that it is holding in anticipation of reuse.

Use the DYNAMNBR parameter on the EXEC statement to replace the DD DYNAM statements you would have to code. The format of the DYNAMNBR parameter is:

```
//stepname EXEC PGM=program-name,DYNAMNBR=n
```

Where n is the number of DD DYNAM statements you would otherwise have to code.

When you code the DYNAMNBR parameter and DD statements, the system uses the sum of the number of DD statements and the DYNAMNBR value to determine the limit of resources it is to hold in anticipation of reuse.

You can dynamically deallocate resources during the execution of a job step (at the time the data set is closed) by coding the FREE=CLOSE parameter on a DD statement.

There are some circumstances when you should not code the FREE parameter.

- The data set name is referenced in a subsequent step.
- The data set name is referenced in another DD statement in the same step.

Do not use the FREE parameter for a data set if a subsequent DD statement requests unit affinity to this DD statement. For example, do **not** code the following:

```
//DD1 DD DSN=dsname,DISP=OLD,UNIT=TAPE,VOL=SER=111111,FREE=CLOSE
      .
      .
//DD3DD DISP=(,KEEP),DSN=dsname2,UNIT=AFF=DD1
```

For more information on coding the FREE parameter, see “FREE Parameter” on page 12-84. If you do dynamically deallocate a resource at close time, it cannot be reopened in the same step. If you do not want to dynamically deallocate the resource, either specify nothing or specify FREE=END to let the system deallocate the resources at the end of the job step.

For more information on how to use dynamic allocation and deallocation and the control limit, see *SPL: Job Management*.

## Example of Dynamically Deallocating Data Sets

```
//PROS   JOB   CLASS=A,MSGLEVEL=(2,0),PERFORM=70
//STEP1  EXEC  PGM=TEST,DYNAMNBR=4,PARM=(P3,123,MT5)
//OUT1   DD   SYSOUT=C,FREE=CLOSE
//OUT2   DD   SYSOUT=A
//SYSIN  DD   *
.
.
.
data
.
.
.
/*
```

1. The JOB statement specifies that this job will be processed in class A in performance group 70. Only JCL statements will be printed.
2. The control limit is **the sum of the number of DD statements coded and the value coded in the DYNAMNBR parameter**; in this case, seven. If this control limit is exceeded and a request for another dynamic allocation is made, the request is not honored unless resources can be deallocated so that the control value is not exceeded.
3. When OUT1 is closed, it is immediately ready for printing.

## Allocating Data Resources in a JES3 System

Data resources, that is, the devices, data sets, and volumes required for each DD statement request, are allocated either by JES3 or by the system according to the DSNAME, DISP, UNIT, and VOLUME parameters on the DD statement. Allocation is handled differently for existing and new data sets and for devices managed by the system, JES3, or jointly. Allocation requires access to the catalog.

**Existing data sets:** If you request an existing data set, data resources are allocated differently according to how the required or specified device is managed:

- A JES3-managed device: JES3 allocates the request before the job executes. For this allocation, JES3 examines the request in relation to other data requests in this and other jobs.
- A MVS-managed device: The system allocates the request as the step enters execution.

JES3 does **not** allocate direct access storage (DASD) space.

For a JES3-managed device, you can change the way JES3 handles the allocation by specifying the SETUP parameter on the JES3 MAIN statement. See "Types of JES3 Setup" on page 6-5.

**New data sets:** If you request a new nonspecific data set and you require or specify a JES3-managed unit:

- JES3 allocates all tape and MSS requests.
- If you code PRIVATE in the VOLUME parameter, JES3 allocates all DASD requests.

**Device Management:** In a JES3-controlled complex, devices are managed in three ways: MVS-managed, jointly managed (JES3/MVS), or JES3-managed. The following chart shows how each type of device can be managed.

How Managed	Attribute of Device	
	Permanently Resident	Removable
MVS managed only	X	X
Jointly managed (JES3/MVS)	X	
JES3-managed only		X

JES3 allocates JES3-managed devices and jointly managed devices. MVS allocates MVS-managed and jointly managed devices. The system programmer defines how each device is managed. Refer to "Requesting Units and Volumes" on page 7-24 for a brief discussion of MVS allocation. Refer to *SPL: Job Management* for additional information on MVS allocation and to *JES3 SPL: Initialization and Tuning* for additional information on JES3 allocation.

**Catalog Access:** To allocate data resources, JES3 accesses the catalog at job setup time, whereas MVS accesses the catalog at step execution time. After job setup and before step execution, the catalog can be changed by, for example, an IBM utility, user utility, or SVC routine. Because JES3 and MVS access the catalog at different times, such catalog changes can cause unpredictable results. Therefore, you must make sure that the catalog remains the same from job setup until job execution.

### Types of JES3 Setup

JES3 allocates devices in three different ways: job setup, high watermark setup, and explicit setup.

**Job setup:** Job setup results in allocation of all the JES3-managed and jointly-managed devices required in the job before the job executes.

To obtain job setup, specify `SETUP=JOB` on the `MAIN` statement. If you specify `MSS=JOB` on the `MAIN` statement, JES3 allocates all mass storage system (MSS) requests. However, JES3 never mounts or demounts MSS volumes.

JES3 mounts the initial volumes necessary to run all steps before the job executes.

When volumes are no longer needed, they will be demounted and the devices deallocated, that is, made available for use by another job. If you specify the `FREE=CLOSE DD` parameter, JES3 deallocates the device when the data set is closed. If you are using the dequeue at demount facility (early volume release) for multivolume data sets, JES3 deallocates volumes when they are demounted. For information on the dequeue at demount facility, see the `TYPE=J OPEN` macro option in *SPL: Data Management*.

**High watermark setup::** High watermark setup results in JES3 reserving for a job the highest number of devices of each type needed for any one job step. High watermark setup does not cause premounting of all mountable volumes. When you must use fewer devices for a job, high watermark setup is better than job setup.

To obtain high watermark setup, specify one of the following:

- For tape, direct access devices, graphics, and unit record devices, SETUP=HWS on the MAIN control statement for the job.
- For tapes only, SETUP=THWS on the MAIN control statement for the job.
- For direct access devices only, SETUP=DHWS on the MAIN control statement for the job.
- For MSS devices, MSS=HWS on a MAIN control statement for the job.
- SETUP=THWS, SETUP=DHWS, or SETUP=HWS on the STANDARDS initialization statement, if the SETUP or MSS parameter is not specified on a MAIN statement.
- MSS=HWS on the SETPARM initialization statement, if the SETUP or MSS parameter is not specified on a MAIN statement.

High watermark setup, like job setup, causes devices, volumes, and data sets to be returned to JES3 for use by other jobs as soon as the resource is deallocated in the last step using it.

In the high watermark setup shown in Figure 6-1 on page 6-7, volume A is mounted for use in STEP1 and then demounted until needed in STEP4. Volume K is mounted for use in STEP1 and STEP2 and then demounted until needed in STEP4. When needed in STEP4, volumes A and K are mounted on any available device.

**Explicit setup:** Explicit setup is user directed. It uses the number of devices required by job setup, but premounts volumes according to the explicit setup specifications.

To obtain explicit setup, specify one of the following:

- SETUP=ddname on the MAIN statement, where ddname is the request that JES3 is to use for this setup.
- SETUP=/ddname on the MAIN statement, where ddname is the request that JES3 is to remove from consideration for this setup.

An advantage of explicit setup over high watermark setup is that you can force volumes to stay mounted on devices until they are no longer needed. A disadvantage of explicit setup is that JES3 does not deallocate devices early: JES3 allocates a certain number of devices before job execution and does not deallocate any until the job completes execution. In contrast, JES3 using job setup and high watermark setup can deallocate devices at the end of any step, if the devices are no longer needed.

In the explicit setup shown in Figure 6-1 on page 6-7, four devices are allocated for both tape and disk instead of the three allocated using high watermark setup. When you explicitly request that JES3 mount certain volumes, the volumes you specify, for example volumes A and K, are not deallocated and remounted for the last step.

Devices and Volumes to be Allocated	Three Types of JES3 Setup																								
	Job Setup (SETUP=JOB)					High Watermark Setup <sup>1</sup> (SETUP=HWS)				Explicit Setup (SETUP=ddname)															
	Tape			Disk		Tape		Disk		Tape			Disk												
Volumes on Devices Set Up Prior to Execution	A	B	C	D	E	F	K	L	M	N	O	A	B	D	K	L	N	A	B	C	D	K	L	M	N
Steps in a Job <sup>2</sup>																									
STEP1 tape volume=A, B disk volume=K, L																									
STEP2 tape volume=B, C, D disk volume=K																									
STEP3 tape volume=D disk volume=L, M, N																									
STEP4 tape volume=A, E, F disk volume=K, N, O																									
Total devices used by the job for setup	6 Tape			5 Disk		3 Tape		3 Disk		4 Tape			4 Disk												

**LEGEND:**

- The device is allocated and in use
- The device is allocated but not in use
- The device is no longer needed and can be deallocated

<sup>1</sup>High watermark setup can express combinations of tape and disk allocations.  
*HWS* request allocation of the minimal number of devices required to run the job.  
*THWS* requests high watermark setup for tapes and job setup for disks.  
*DHWS* requests high watermark setup for disks and job setup for tapes.

<sup>2</sup>Volumes mounted after STEP1 are indicated by placing the volume name in the box for the step in which it is allocated. For example, in high watermark setup, volume C is mounted at STEP2.

**Figure 6-1. Types of JES3 Setup**

**Altering JES3 Device Allocation:** To prevent JES3 from allocating devices before the first job step begins execution and holding them until a later job step needs them, you can break a multiple-step job into several smaller, dependent jobs. “Dependent Job Control for JES3: The Job Net” on page 3-27 tells how to split a job into smaller, dependent jobs.





## Part 3. Guide to Data Set Control

The primary functions of JCL DD statements are to describe the characteristics of data sets and to indicate their location to the system. These functions allow you a great deal of freedom in writing your programs. For example, if you are writing a program to process paid bills, you do not have to indicate in your program the size of the input records, or the type of device where the records are located. You can postpone these definitions until you run the program. At that time you code the DD statements for the input data sets.

You can debug your program, and then run it several times with different DD statements for the input record set. In this way you can determine which record size is most efficiently processed, and whether the input should come from a card reader or a magnetic tape unit. All your program needs to know to refer to the data set is the name of the DD statement (ddname) that describes the data set. Each time you execute the program you can use the DD statement to describe a different data set as long as the ddname remains constant.

You can define data set characteristics within your program so that you will not have to specify those characteristics that remain constant each time you use a data set. The number and type of data set characteristics you can specify in your program, rather than in the DD statement, depends on the language you are using for writing your program. However, regardless of the facilities of the language you are using, you should only specify **in your program** those requirements essential to processing and leave the rest for the DD statement. This gives you more flexibility in writing the program and places fewer restrictions on any future changes you may have to make to the program.

All job steps in your job (except those steps that use a cataloged procedure) require DD statements because every program must have either an input data set, or an output data set, and, in many cases, work data sets in order to operate. The names of the DD statements required for IBM-supplied programs, such as compilers and utilities, are predefined and you must code their parameters according to the rules stated in the publications associated with the programs.

Only you can determine the DD statements required for your own program. There must be a DD statement for each data set that you use in your job step that is not dynamically allocated. DD statements follow the EXEC statement that marks the beginning of the job step. You can include a maximum of 1635 DD statements in each job step.

If the job step uses a cataloged procedure, you can use a DD statement either to override parameters in a DD statement in the procedure, or to add a new DD statement to the procedure. In either case, the modification remains in effect only for the duration of the job step, it does not change the procedure permanently.

A DD statement must contain the term DD in its operation field. Although all parameters in the DD statement's operand field are optional, a blank operand field is invalid except when you are overriding DD statements defining concatenated data sets in a cataloged procedure.

The parameters in the operand field allow you to specify the following:

- Data set information
- Unit and volume requests
- Data sets for mass storage systems
- Space for non-VSAM data sets
- Data set processing options

Not all DD statement parameters are needed to define a data set. In fact, you cannot use some combinations of parameters in the same DD statement.

The valid combinations of DD statement parameters allow you to perform the following functions:

- Create a data set
- Retrieve an existing data set
- Extend an existing data set
- Define special data sets
- Postpone definition of a data set

This section describes the ddname and the parameters you need for each of the above functions. Also discussed are any JES control statement parameters that you can code for a particular function.

## Chapter 7. Guide to Specifying Data Set Information

You must provide certain data set information to enable the system to deal with your data sets.

### Specifying the DDNAME Parameter

You use the DDNAME parameter most often in cataloged procedures and in job steps that call procedures. It is used in cataloged procedures to postpone defining data in the input stream until a job step calls the procedure. (Procedures cannot contain DD statements that define data in the input stream; that is, DD \* or DD DATA statements). In job steps that call procedures it is used on an overriding DD statement to postpone defining data in the input stream until the last overriding DD statement for a procedure step. Overriding DD statements must appear in the same order as the corresponding DD statements in the procedure.

### When You Code the DDNAME Parameter

When the system encounters a DD statement that contains the DDNAME parameter, it saves the ddname of that statement. The system also temporarily saves the name specified in the DDNAME parameter so that it can relate that name to the ddname of a later DD statement. Once a DD statement with that corresponding name is encountered, the name is no longer saved. For example, if the system encounters this statement

```
//XYZ DD DDNAME=PHOB
```

the system saves XYZ and, temporarily, PHOB. Until the system encounters the ddname PHOB in the input stream, it treats the data set as a dummy data set.

When the system encounters a statement whose ddname has been temporarily saved, it does two things. It uses the information contained on this statement to define the data set; it associates this information with the name of the statement that contained the DDNAME parameter. The value that appeared in the DDNAME parameter is no longer saved by the system. To continue the above example, if the system encounters this statement

```
//PHOB DD DSN= NIN, DISP=(NEW,KEEP), UNIT=3400-5
```

the system uses the data set name and the disposition and unit information to define the data set; it also associates the ddname of the statement that contained the DDNAME parameter with this information. In this example, the ddname used is XYZ; the ddname PHOB is no longer saved. The data set is now defined just as it would be if you had coded

```
//XYZ DD DSN= NIN, DISP=(NEW,KEEP), UNIT=3400-5
```

The system associates the ddname of the statement that contains the DDNAME parameter with the data set definition information. It does not use the ddname of the later statement that defines the data set. Therefore, any references to the data set, before or after the data set is

defined, must refer to the DD statement that contains the DDNAME parameter, not the DD statement that defines the data set. The following sequence of control statements illustrates this:

```
//DD1 DD DDNAME=LATER
.
.
//LATER DD DSN=SET12,DISP=(NEW,KEEP),UNIT=3350,
// VOLUME=SER=46231,SPACE=(TRK,(20,5))
.
.
//DD12 DD DSN=SET13,DISP=(NEW,KEEP),VOLUME=REF=*.DD1,
// SPACE=(TRK,(40,5))
```

DD1 postpones defining the data set until the system encounters DD statement LATER. DD12 must do a backward reference to DD1 because the system associates the data set information with the DD statement that contains the DDNAME parameter.

When you want to concatenate data sets, the unnamed DD statements must follow the DD statement that contains the DDNAME parameter, not the DD statement that defines the data set. The following sequence of control statements illustrates this:

```
//DDA DD DDNAME=DEFINE
// DD DSN=A.B.C,DISP=OLD
// DD DSN=SEVC,DISP=OLD,UNIT=3350,VOL=SER=52226
.
.
//DEFINE DD *
data
/*
```

You can use the DDNAME parameter up to five times in a job step or procedure step. However, each time the DDNAME parameter is coded, it must refer to a different ddname.

## Specifying the DSNAME Parameter

When creating a data set, use the DSNAME parameter to assign a name to the data set. The data set name is part of the information stored with the data set on a volume. Later, when another job step or job wants to use the data set, it identifies the data set name in the DSNAME parameter; the system uses the data set name to locate the data set on the volume.

How you code the DSNAME parameter depends on the type of data set and whether the it is nontemporary or temporary.

Note that if you code a data set name ending in .GnnnnVnn (where n=0 to 9) for a tape, your data set is always treated as part of a generation data group by data management routines. For more information on generation data groups, see "Creating and Retrieving Generation Data Sets" on page 8-25.

## Creating or Retrieving a Nontemporary Data Set

If the data set is nontemporary, you can identify:

- A permanent data set by coding `DSNAME=dsname`
- A member of a nontemporary partitioned data set by coding `DSNAME=dsname(member name)`
- A generation of a nontemporary generation data group by coding `DSNAME=dsname(number)`
- An area of a nontemporary indexed sequential data set by coding `DSNAME=dsname(area name)`

### Nontemporary Data Sets

When a nontemporary data set is created, it is assigned a name in the `DSNAME` parameter and is assigned a disposition of `KEEP` or `CATLG`. (A data set assigned a disposition of `KEEP` may be assigned a disposition of `CATLG` by a later job step or job). All other steps and jobs that want to use the data set must specify the `DSNAME` parameter using either the data set's assigned name or its backward reference.

A nontemporary data set name can be either an unqualified or qualified name. An unqualified data set name consists of 1 through 8 characters. The first character must be an alphabetic or national (`@,#,$`) character; the remaining characters can be any alphanumeric or national characters, a hyphen, or plus zero (`+0`). Note that national characters are invalid for ISO/ANSI/FIPS Version 3 tape data set names.

A qualified data set name consists of 1 through 44 characters (including periods), except when the qualified name identifies a generation data group. In this case, the data set name may consist of only 1 through 35 characters (including periods). For each eight characters or less there must be a period, and the first character of the name and the character following a period must be an alphabetic or national (`@,#,$`) character.

When you request a data set that is cataloged on a control volume or a private catalog, the system attempts to mount this control volume if it is not already mounted. After the system obtains the pointer to the requested data set, the control volume or private catalog can then be demounted by the system if the unit on which it was mounted is required by another volume. The control volume or private catalog is assigned to the job step and is available for disposition processing when the job step ends.

In the following cases, the control volume or private catalog is not mounted when disposition is processed:

- The job fails or abnormally terminates and data sets with a conditional disposition of `CATLG` or `UNCATLG` have been passed but not received.
- A job step is deallocated during system warm start.

## Members of a Partitioned Data Set

A partitioned data set consists of independent groups of sequential records, each identified by a member name in a directory. When you want to add a member to a partitioned data set or retrieve a member, specify the partitioned data set name and follow it with the member name. The member name is enclosed in parentheses and consists of 1 to 8 characters. The first character must be an alphabetic or national (@,\$,#) character, the remaining characters can be any alphanumeric or national characters.

## Generations of a Generation Data Group

A generation data group is a collection of chronologically related data sets that can be referred to by the same data set name. When you want to add a generation to a generation data group or retrieve a generation, specify the generation data group name and follow it with the generation number. The generation number is enclosed in parentheses and the number is a zero or a signed integer. A zero represents the most current generation of the group, a negative integer (for example, -1) represents an older generation; a positive integer (for example, +1) represents a new generation that has not as yet been cataloged.

To retrieve all generations of a generation data group (up to 255 generations), code only the group name in the DSNAME parameter and the DISP parameter.

A complete discussion of creating and retrieving generation data sets is contained in "Creating and Retrieving Generation Data Sets."

## Areas of an Indexed Sequential Data Set

The areas used for an indexed sequential data set are the index, prime, and overflow areas. When you are creating the data set and define any of these areas on a DD statement, you must identify the data set name and follow it with the area name you are defining. The area name is enclosed in parentheses and is either PRIME, INDEX, or OVFLOW. If you are using only one DD statement to define the entire data set, code DSNAME=dsname or DSNAME=dsname(PRIME). When you retrieve the data set, you code only the data set name; you do not include the terms PRIME, INDEX, or OVFLOW.

## Creating or Retrieving a Temporary Data Set

If the data set is temporary, you can identify:

- A temporary data set by coding DSNAME=&&dsname
- A member of a temporary partitioned data set by coding DSNAME=&&dsname(member name)
- An area of a temporary indexed sequential data set by coding DSNAME=&&dsname(area name)

## Temporary Data Sets

Any data set that is created and deleted within the same job is a temporary data set. A DD statement that defines a temporary data set need not include the DSNNAME parameter; the system generates one for you.

If you do include the DSNNAME parameter, the temporary data set name can consist of 1 through 8 characters and is preceded by two ampersands (&&). The character following the ampersands must be alphabetic or national (@, #, \$) characters; the remaining characters can be any alphanumeric or national characters. (A temporary data set name that is preceded by only one ampersand is treated as a temporary data set name as long as you do not assign a value to it either on the EXEC statement for this job step when it calls a procedure, or on a PROC statement within the procedure. If a value is assigned to it by one of these means, it is treated as a symbolic parameter).

The system generates a qualified name for the temporary data set that begins with SYS and includes the Julian date, the time, the jobname, the temporary name assigned in the DSNNAME parameter if specified (or an identifying name and number if not specified), and other identifying characters.

*Note:* The time in the system-generated qualified name is the time that the converter/interpreter is invoked. Because the system invokes the converter/interpreter only once per job, if you use the same temporary data set name more than once per job, you might get a JCL error.

If you attempt to keep or catalog a temporary data set (by specifying a disposition of KEEP or CATLG in the DISP parameter), the system changes the disposition to PASS and the data set is deleted at job termination. However, this change is not made for a data set on a tape volume when the following conditions exist:

- The data set is new
- The data set is not assigned a name
- You specify a status of OLD or SHR in the DISP parameter
- You specify DEFER in the UNIT parameter

The data set is deleted at job termination, but the system tells the operator to keep the volume on which the data set resided during the job. If you code a conditional disposition for temporary data sets, it is ignored.

To simplify processing of temporary data sets, see “Using Virtual Input/Output (VIO) for Temporary Data Sets.”

### Members of a Temporary Partitioned Data Set

When adding a member to a temporary partitioned data set or retrieving a member during the job, specify the partitioned data set's temporary name and follow it with the member name. The member name is enclosed in parentheses and consists of 1 through 8 characters. The first character must be an alphabetic or national (@, \$, #) character; the remaining characters can be any alphanumeric or national characters.

## Areas of a Temporary Indexed Sequential Data Set

The areas you specify for indexed sequential data set are the index, prime, and overflow areas. When you are creating a temporary indexed sequential data set and define any of these areas on a DD statement, you must identify the data set's temporary name and follow it with the area name you are defining. The area name is enclosed in parentheses and is either PRIME, INDEX, or OVFLOW. If you are using only one DD statement to define the entire temporary data set, code DSNAME=&&dsname or DSNAME=&&dsname(PRIME). If you want to retrieve the temporary data set on the same job, you code only the data set's temporary name; you do not include the term PRIME, INDEX, or OVFLOW.

## Associated Data Sets (3540 Diskette)

Associated data sets are data sets on 3540 diskette volumes that are separate from the job stream data set and are to be spooled as SYSIN data sets. Associated SYSIN data sets are identified by specifying a data set identifier (on the DD DSID parameter) and, optionally, a volume identifier on the DD \* or DD DATA statement in the job stream.

To have associated data sets merged into the job stream, the job stream containing the diskette associated data set requests must be processed by the diskette reader program; neither JES2 nor JES3 can read it.

Data sets are created on 3540 diskette volumes only by using SYSOUT. The SYSOUT DD statement must contain the DSID parameter and a sysout class (or classes) designed by the installation to be used by data sets on a 3540 diskette. The diskette writer must be started to the sysout class to transfer the data sets to diskettes.

For more information on the 3540 diskette, refer to "OS/VS2 IBM 3540 Programmer's Reference."

## Copying the Data Set Name from an Earlier DD Statement

The name of a data set that is used several times in a job can be copied after its first use in the job. You can copy the data set name whether specified in the DSNAME parameter or assigned by the system. This allows you to easily change data sets from job to job and eliminates your having to assign names to temporary data sets. To copy a data set name, refer to an earlier DD statement that identifies the data set.

*Note:* When copying a data set's name from an earlier DD statement, you may also copy other information from the DD statement. The other information is:

- Whether or not the data set is a partitioned data set (PDS).
- Whether or not the data set is a temporary data set.

Do not copy data set names of subsystem data sets created by a DD \* or a DD DATA statement.



When the earlier DD statement is contained in an earlier job step, code,

```
DSNAME=*.stepname.ddname
```

When the earlier DD statement is contained in the same job step, code,

```
DSNAME=*.ddname
```

When the earlier DD statement is contained in a cataloged procedure step called by an earlier job step, code,

```
DSNAME=*.stepname.procstepname.ddname
```

## Specifying the DSNAME Parameter in Apostrophes

Sometimes, it may be necessary or desirable to specify a data set name that contains special characters. If the name contains special characters, you must enclose the name in apostrophes, for example, `DSNAME='DAT+5'`. If one of the special characters is an apostrophe, you must identify it by coding two consecutive apostrophes in its place, for example, `DSNAME='DAY''SEND'`. A data set name enclosed in apostrophes can consist of 1 through 44 characters.

There are cases when the data set name must contain required special characters, which tell the system something about the data set (for example, `&&` in `DSNAME=&&name` are required special characters that tell the system that this is a temporary data set). In these cases, the data set name must not be enclosed in apostrophes because the system will not recognize the required special characters as having any special significance. The following data set names contain special characters that tell the system something about the data set and, therefore, cannot be enclosed in apostrophes:

- `DSNAME=name(member name)`
- `DSNAME=name(area name)`
- `DSNAME=name(generation number)`
- `DSNAME=&&name`
- `DSNAME=*.stepname.ddname`
- Part of, or the entire data set name, that is to be symbolically substituted

Keep the following rules in mind:

- If the data set name ends with a blank character, the blank is ignored.
- If the only special character is a period used to create a qualified data set name, a hyphen, or plus zero (0-12 punch), you need not enclose the data set name in apostrophes.

## Specifying the LABEL Parameter

The operating system uses labels to identify volumes and the data sets they contain, and to store data set attributes. Data sets residing on magnetic tape volumes usually have data set labels. If data set labels are present, they precede each data set on the volume. Data sets residing on direct access volumes always have data set labels. These data set labels are contained in the volume table of contents of the direct access volume.

A data set label may be a standard or nonstandard label. Standard labels can be processed by the system; nonstandard labels must be processed by nonstandard label processing routines,

which the installation includes in the system. Data sets on direct access volumes must have standard labels. Data sets on tape volumes usually have standard labels, but can have nonstandard labels or no labels.

The LABEL parameter must be coded if:

- You are processing a tape data set that is not the first data set on the reel; in this case, indicate the data set sequence number.
- The data set labels are not IBM standard labels; you must indicate the label type.
- You want to specify what type of labels a data set is to have when it is written on a scratch volume; indicate the label type.
- The data set is to be password protected; specify PASSWORD when creating the data set.
- The data set is to be processed only for input or output and this conflicts with the processing method indicated in the OPEN macro instruction; specify IN, for input, or OUT, for output.
- The data set is to be kept for a specific period of time; indicate a retention period (RETPD) or expiration data (EXPDT).

#### **The Data Set Sequence Number Subparameter**

When placing a data set on a tape volume that already contains one or more data sets, specify where the data set is to be placed, that is, whether the data set is to be the second, third, fourth, etc., data set on the volume. The data set sequence number causes the tape to be positioned properly so that the data set can be written on the tape or retrieved.

The data set sequence number subparameter is a positional subparameter and is the first subparameter that you can code in the LABEL parameter. The data set sequence number is a 1- to 4-digit number. The data set sequence number is ignored for the following types of data sets:

- For data sets passed from a previous step, the system obtains the data set sequence number from the passing step.
- For GDG ALL requests, the system always retrieves the data set sequence number from the catalog.

If you omit the data set sequence number subparameter or code 0, the system assumes 1 (this is the first data set on the tape) unless the data set is cataloged. If the data set is cataloged, the system obtains the data set sequence number from the catalog.

#### **Specifying the Label Type**

The label type subparameter tells the system the type of labels associated with the data set. The label type is a positional subparameter and must be coded second in the LABEL parameter, after the data set sequence number subparameter. You can omit this subparameter if the data set has IBM standard labels.

The label type is specified as:

- SL — if the data set has IBM standard labels.
- SUL — if the data set has both IBM standard and user labels.
- AL — if the data set has ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels.
- AUL — if the data set has ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels, and ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 user labels.
- NSL — if the data set has nonstandard labels.
- NL — if the data set has no labels.
- BLP — if you want label processing bypassed.
- LTM — if you want the system to bypass a leading tape mark on unlabeled tape (OS/DOS interchange).

SL or SUL is the only label type that can be specified for data sets that reside on direct access volumes. SL, SUL, AL, AUL, NSL, and NL are the only label types that can be specified for data sets that reside on tape volumes. BLP and LTM are label type subparameters that can also be coded for tape.

When SL or SUL is specified, or the label type is omitted and the data set has IBM standard labels, the system can ensure that the correct tape or direct access volume is mounted.

When referring the operating system to an earlier tape volume request (by receiving a passed data set or by using VOL = REF = reference), you should specify SL or SUL as the label type. Specifying any other label type causes the operating system to copy the label type from the referenced request — overriding the label type you specify on the DD statement making the reference.

When specifying NSL, installation-provided nonstandard label processing routines must ensure that the correct tape volume is mounted.

When specifying NL or BLP, the operator must ensure that the correct tape volume is mounted. If you specify NL, the data set must have no standard labels.

When specifying AL or AUL, the system ensures that the correct tape is mounted; to be correct, the tape must have a ISO/ANSI Version 1 or ISO/ANS/FIPS Version 3 label.

Specifically, if your installation has specified ASCII = INCLUDE during system generation, then the specification of LABEL = (,AL) or LABEL = (,AUL) requests translation. You can also request translation by specifying OPTCD = Q. If the tape is not labeled, LABEL = (,NL), you **must** specify OPTCD = Q for translation to occur.

For cataloged and passed data sets, label type information is not kept. Therefore, when referring to a cataloged or passed data set that has other than standard labels, code the LABEL parameter and specify the label type.

BLP is not a label type, but a request that the system bypass label processing. This specification allows you to use a blank tape or overwrite a seven-track tape that differs from the

current parity or density specifications. If the bypass label processing option is not selected by the installation and you have coded BLP, the system assumes NL.

When retrieving data sets from each of several NL or BLP tape volumes and you are coding the data-set-sequence-number subparameter, you must set up a concatenation with one tape volume for each DD statement and you must **repeat** the LABEL parameter on each DD statement.

*Note:* When you request the system to **bypass label processing** (LABEL = BLP) and the tape volume has labels, the system treats anything between tapemarks as a data set.

Therefore, in order for a tape with labels to be positioned properly, you **must** code the data-set-sequence-number subparameter of the LABEL parameter and the subparameter must reflect all labels and data sets that precede the desired data set. The *Tape Labels* publication illustrates where tapemarks appear.

**Nonspecific volume request:** The label type subparameter can also be specified when making a nonspecific volume request for a tape volume (that is, no volume serial numbers are specified on the DD statement) and when having a certain type of label. If the volume that is mounted does not have the corresponding label type desired, you may be able to change the label type.

When you specify NL or NSL and the operator mounts a tape volume that contains standard labels, you can use the volume if **all** the following are true:

1. The expiration data of the existing data set on the volume has passed;
2. The existing data set on the volume is not password protected;
3. You make a nonspecific volume request.

If they are not all true, the system requests the operator to mount another tape volume.

If you specify SL and make a nonspecific volume request, but the operator mounts a tape volume that contains other than IBM standard labels, the system asks the operator to identify the volume serial number and the volume's new owner before the IBM standard labels are written. If the tape volume has ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels, the system asks the operator for permission to destroy the labels.

If you specify SL and make a specific volume request, but the volume that is mounted does not contain IBM standard labels:

- The system rejects the tape and requests the operator to mount the tape volume specified, or
- If the volume currently mounted is not labeled, the operator has the option of labeling the tape volume or rejecting it.

### **The PASSWORD and NOPWREAD Subparameters**

The PASSWORD and NOPWREAD subparameters tell the system that you want the data set to be password-protected.

- If you specify PASSWORD, the data set cannot be read from, written into, or deleted by another job step or job unless the operator can supply the system with the correct password.

- If you specify NOPWREAD (no password read), the data set can be read without the operator supplying the password, but the password is still required for writing or deleting data sets.

The PASSWORD and NOPWREAD subparameters are positional and must be coded third, after the data set sequence number subparameter and the label type subparameter; if you do not code the preceding positional subparameters, you must code commas to indicate their absence.

For example,

```
//EX1 DD LABEL=(data-set-sequence-number,SL,PASSWORD)
//EX2 DD LABEL=(data-set-sequence-number,,PASSWORD)
//EX3 DD LABEL=(,AL,PASSWORD)
//EX4 DD LABEL=(,,PASSWORD)
```

If you want the data set password-protected, specify PASSWORD when the data set is created. Password-protected data sets must have standard labels, either IBM standard or ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels.

### Overriding OPEN Macro Options: The IN and OUT Subparameters

The basic sequential access method (BSAM) permits a specification of INOUT, OUTIN, or OUTINX in the OPEN macro instruction as the processing method.

The basic direct access method (BDAM) permits a specification of UPDAT in the OPEN macro instruction as the processing method.

OPEN macro specification	LABEL subparameter specification	System treats the open as if
INOUT (BSAM)	IN	INPUT was specified in OPEN macro. See Note 1.
OUTIN	OUT	OUTPUT was specified in OPEN macro. See Note 1.
OUTINX	OUT	EXTEND was specified in OPEN macro. See Note 2.
<p><b>Note 1:</b> When you code the IN subparameter, and the OPEN for INOUT or UPDAT is in effect, any attempt by the processing program to process the data set for <b>output</b> results in abnormal termination.</p> <p><b>Note 2:</b> When you specify the OUT subparameter, and the OPEN for OUTIN or OUTINX is in effect, any attempt by the processing program to process the data set for <b>input</b> results in abnormal termination.</p>		

The IN and OUT subparameters are positional subparameters and must appear as the fourth subparameter in the LABEL parameter. That is, IN or OUT must follow the data-set-sequence-number subparameter, the label-type subparameter, and the PASSWORD

subparameter — or the commas that indicate the absence of a preceding subparameter. For example;

```
//DD1 DD LABEL=(data-set-sequence-number,SL,PASSWORD,IN)
//DD2 DD LABEL=(,AL,NOPWREAD,OUT)
//DD3 DD LABEL=(,PASSWORD,IN)
//DD4 DD LABEL=(,,OUT)
```

The IN subparameter allows a program that opens for INOUT to read a password protected data set. Because the OPEN does not allow output processing, the read password and not the write password allows the data set to be read.

If the data set is protected with the NOPWREAD (read without password) option, no password is required to read the data set.

You can also use the IN subparameter to avoid operator intervention when reading a data set that has an unexpired expiration date.

*Note:* When you specify OUTINX or EXTEND in the OPEN macro instruction or if you specify the OUT subparameter, the system adds records to the end of the data set regardless of what you specify on the DISP parameter of the DD statement.

### The RETPD and EXPDT Subparameters

When a data set must be kept for some time, you can tell the system how long when you create the data set, by means of the LABEL parameter. As long as the time period has not expired, a data set that resides on a direct access volume cannot be deleted by or overwritten by another job step or job; this is true even if the job or step has specified a disposition of DELETE for the data set. If it is necessary to delete a data set before the expiration date or retention period has passed, use one of the following methods:

- For data sets cataloged in a VSAM catalog use the DELETE command; this makes the space occupied by the data set available for reallocation. See *Access Method Services*.
- To delete the catalog entry for data sets that are not cataloged in a VSAM catalog, use the IEHPROGM utility as described in *Utilities*.
- To delete the data set control block use the SCRATCH macro with the "OVRD" parameter; this makes the space occupied by that data set available for reallocation. See *SPL: Data Management*.

When the expiration date of a data set is the current date, the data set is considered expired and another data set can delete or write over it.

There are two different ways to specify a time period:

1. Tell the system how many days you want the data set kept, through the RETPD subparameter, or
2. Tell the system the exact date after which the data set need not be kept, through the EXPDT subparameter.

If you code the RETPD subparameter, you specify a 1- to 4-digit number, which represents the number of days the data set is to be kept. If you code the EXPDT subparameter, you specify a 2-digit year number and a 3-digit day number (for example, January 1 would be 001, July 1

would be 182); the number you code represents the date after which the data set need not be kept. When neither the RETPD or EXPDT subparameter is specified for a new data set, the system assumes a retention period of zero days.

The RETPD or EXPDT subparameter must follow all other subparameters of the LABEL parameter. If no other subparameters are coded, you can code LABEL = RETPD = nnnn or LABEL = EXPDT = yyddd.

## Example of Identifying Data Sets to the System

This job shows how to use the DSNNAME parameter.

```

/*PRIORITY      8
//DATASETS     JOB  FREEMAN,MSGLEVEL=1
//STEP1        EXEC PGM=IEFBR14
//D1           DD   DSN=ABC,DISP=(NEW,CATLG),UNIT=3350,
//             VOL=SER=333001,SPACE=(CYL,(12,1,1),CONTIG)
//D2           DD   DSN=&&NAME,UNIT=3330,SPACE=(TRK,(10,1))
//D3           DD   DSN=SYSLIB,DISP=(OLD,KEEP)
//D4           DD   *
.
.
.
data
.
.
.
/*

```

1. This job runs in priority 8, the meaning of which is defined by the installation.
2. The job statement specifies that system messages and JCL statements are to be printed (MSGLEVEL = 1).
3. D1 catalogs a newly created data set. The space request is for 12 primary cylinders, 1 secondary, 1 directory, and the space is to be contiguous.
4. D2 creates a temporary data set on a 3330. The space request is for 10 primary tracks and 1 secondary.
5. D3 defines an old cataloged data set.
6. D4 defines a SYSIN data set. This will be followed by data in the input stream.

## Disposition Processing of Non-VSAM Data Sets

Processing of data sets at the end of a job step or when closed and FREE = CLOSE is specified, is known as disposition processing. You request disposition processing for non-VSAM data sets by coding the DISP parameter on the DD statement defining the data set. (VSAM data sets are handled differently. For information on VSAM, refer to *VSAM Programmer's Guide*.)

In the DISP parameter, you can code:

- Data set status as the first subparameter, indicating whether the data set is new, is old, can be shared with other jobs, or can be lengthened.

- Normal disposition as the second subparameter, indicating how the data set should be handled if the job step terminates normally.
- Conditional disposition as the third subparameter, indicating how the data set should be handled if the job step terminates abnormally.

If you do not code one of the subparameters, or omit the DISP parameter entirely, the system supplies default values, as described under “Default Disposition Processing.” Refer to Figure 18-7 on page 18-9 for further information on disposition processing.

## Specifying Data Set Status

Indicate a data set’s status by coding one of the following:

- NEW — the data set is being created in this job step.
- OLD — the data set existed before this job step.
- SHR — the data set existed before this job step and can be read simultaneously by other jobs.
- MOD — the system first assumes that the data set exists. However, if the system cannot find volume information for the data set on the DD statement, in the catalog, or passed with the data set from a previous step, the system then assumes that the data set does not exist and the data set is created for the job step. Specifying MOD for a new sequential data set causes the read/write mechanism to be positioned after the last record in the data set. The read/write mechanism is positioned after the last record each time the data set is opened for output when being created.

When coding SHR, you are requesting shared control of the data set and the job should be reading the data set only. When coding NEW, OLD, or MOD, you are requesting exclusive control of the data set. Shared and exclusive control are described in this chapter under “Insuring Data Set Integrity.”

## Specifying a Disposition for the Data Set

You can specify one disposition, called a normal disposition, to be used when the job step terminates normally (successfully) and another disposition, called the conditional disposition, to be used when the job step terminates abnormally. (You can specify a conditional disposition for nontemporary data sets only.)

For normal disposition, you can request as the second subparameter that the data set be:

- Deleted by coding DELETE
- Kept by coding KEEP
- Cataloged by coding CATLG
- Uncataloged by coding UNCATLG
- Passed by coding PASS

*Note:* The disposition of a data set is solely a function of the DISP parameter; however, the disposition of the volumes on which the data set resides is a function of the volume status when the volume is demounted.



For conditional disposition (the third subparameter of the DISP parameter), you may code the same dispositions as for a normal disposition except for PASS. You should consider using conditional disposition every time you create or use a data set. Conditional disposition can be used to keep data sets after a program failure, when they might be needed to determine the cause of the failure. Conditional disposition can also be used to delete data sets in case of program failure, thereby restoring the system environment to what it was before the error. This allows the failing job to be rerun without an intervening clean-up job.

Data sets allocated to steps that have abnormally terminated and that do not have automatic restart, are disposed of as specified by the conditional disposition. If a job step abnormally terminates during execution and a conditional disposition is not specified, the normal disposition is processed.

If a job step fails during step allocation:

- A data set created in that job step is deleted.
- A data set that existed before that job step is kept.

If you are accessing a data set that was cataloged or kept in a step prior to the step that abnormally terminates, and you have not coded a disposition, MVS processing follows the disposition specified in the earlier step.

Disposition processing differs for data sets on direct access volumes and data sets on magnetic tape volumes. A direct access volume contains a volume table of contents (VTOC), which consists of control blocks describing the non-VSAM data sets and available space on the volume. The handling of tape and direct access volumes when specifying a particular disposition is described below.

### Deleting a Data Set

Specifying DELETE requests that the data set's space on the volume be released at the end of the job step (normal disposition) or if the step abnormally terminates (conditional disposition). If the data set resides on a public tape volume, the tape is rewound and the volume is available for use by other job steps. If the data set resides on a private volume, the tape is rewound and unloaded. In this case, it is rewound and unloaded and a KEEP message is issued. If the data set exists on a direct access volume, the control block describing the data set is removed from the VTOC and the space on the volume is then available to other data sets.

In the following case, however, a data set on a direct access volume will not be deleted.

If a data set previously existed and has an unexpired expiration date or retention period, a disposition of DELETE does not delete the data set if the step abnormally terminates.

Specify a length of time that a data set must be kept by assigning a retention period or expiration date in the LABEL parameter on the DD statement. Specifying a retention period or expiration date is described under "Specifying the LABEL Parameter" on page 7-7.

If you are deleting a cataloged non-VSAM data set, the entry for the data set in the system catalog is also removed, provided the system obtained volume information for the data set from the catalog (that is, the volume's serial number was not coded on the DD statement).

If the system did not obtain volume information from the catalog, the data set is still deleted but its entry in the catalog remains. If an error is encountered while attempting to delete a data set, its entry in the catalog remains. (The data set will or will not be deleted, depending on

when the error occurs). To delete an entry from a VSAM catalog, use the DELETE command as described in *VSAM Programmer's Guide*, which makes the space occupied by the data set available for reallocation.

To delete the catalog entries for data sets that are not cataloged in a VSAM catalog, use the UNCATLG statement of IEHPROGM as described in *Utilities*.

DELETE is the only valid conditional disposition for a data set that has no name or that has a temporary name. If you specify a disposition other than DELETE, the system assumes DELETE.

### Keeping a Data Set

Specifying KEEP instructs the system to keep a data set intact until a subsequent job step or job requests that the data set be deleted or at least until the expiration date or retention period is passed. You can specify an expiration date or retention period, indicating the length of time a data set must be kept, in the LABEL parameter on the DD statement. If you do not specify a time period, the system assumes a retention period of zero days. Coding an expiration date or retention period is described under "Specifying the LABEL Parameter" on page 7-7.

If you are assigning a final disposition of KEEP to a passed data set, make certain that you follow the rules for receiving a passed data set. See the discussion under "Passing a Data Set" on page 7-17.

For data sets on direct access devices, the entry in the VTOC describing the data set and the data set itself are kept intact. For data sets on tape, the volume is rewound and unloaded, and a KEEP message is issued to the operator.

### Cataloging a Data Set

Cataloging allows you to keep track of and retrieve data sets. You can catalog data sets in the system master catalog or in user (private) catalogs. A private catalog can be either a VSAM user catalog or an Integrated Catalog Facility (ICF). When retrieving a cataloged data set, you do not have to specify volume information; you need only code the DSNAMES parameter and a status other than NEW in the DISP parameter.

To catalog a non-VSAM data set, code CATLG as the disposition; the system creates an entry in the catalog that points to the data set. The disposition CATLG implies KEEP.

You can specify a disposition of CATLG for an already cataloged data set. Do this when lengthening the data set with additional output (a status of MOD is coded) and the data set can exceed one volume. If the system obtained volume information for the data set from the catalog (that is, the volume's serial number was not coded on the DD statement) and you code DISP=(MOD,CATLG), the system updates the entry to include the volume serial numbers of any additional volumes.

We define a collection of cataloged data sets that are kept in chronological order as a generation data group (GDG). The entire GDG is stored under a single data set name; each data set within the group, called a generation data set, is associated with a generation number that indicates how far removed the data set is from the original generation. For more information on defining and creating generation data groups, see "Generation Data Groups" in this publication, and *VSAM Programmer's Guide*.

*Note:* There are instances when the system will not catalog a data set. The system does not catalog a data set if the DD statement describing that data set is not opened by the problem program and;

- You request a nonspecific tape volume (scratch volume is assumed), or
- You request a tape volume for a tape unit with dual density options and you did not specify the density (DEN subparameter of the DCB parameter) on the DD statement.

### Uncataloging a Data Set

To remove the entry describing a non-VSAM data set from the catalog, code UNCATLG as the disposition. Specifying UNCATLG does not request the initiator to delete the data set; only the reference in the catalog is removed. When you request use of the data set in a subsequent job or job step, you must include volume information on the DD statement.

### Passing a Data Set

If more than one step in a job requests the same data set, each step using the data set can pass the data set for a later step to use. A data set can only be passed within a job.

To pass a data set, code PASS as the normal disposition; PASS cannot be the conditional disposition. You continue to code PASS each time the data set is referred to until the last time it is used in the job. At this time, you assign it a final disposition.

Specifying the data set name of a passed data set without specifying volume serial number or a volume reference is called “receiving” the data set. Identical data set names (whether or not the same data set is referred to) can be passed at the same time. Such identical data set names are received in the same order in which they are passed. A data set name that has been passed *n* times can be received no more than *n* times. A data set cannot be passed and received within the same step.

**Considerations for Passed Data Sets:** Consider the following when you pass data sets:

A data set may be passed more times than it is received. However, a problem can occur when the same data set is passed more times than it is received in a procedure that is called multiple times in a job.

For example, the following procedure is called in a job step:

```
//STEP1 EXEC PGM=IEFBR14
//DD1 DD DSNAME=&A,DISP=(NEW,PASS),
// SPACE=(TRK,(1,1)),UNIT=SYSDA
//DD2 DD DSNAME=* .DD1,DISP=(OLD,PASS),
// VOL=REF=* .DD1
//STEP2 EXEC PGM=IEFBR14
//DD3 DD DSNAME=&A,DISP=(OLD,DELETE)
```

In this example:

- DD1 and DD2 pass data set &A.
- DD3 receives data set &A.
- After the procedure has been called the first time, one entry for data set &A remains unreceived.

- If the procedure is called a second time, DD3 receives data set &A from the first execution of the procedure and this can result in incorrect data or an abnormal termination.
- If data set &A is not received twice in the job, data set &A is processed as an unreceived passed data set at the end of the job.

If a job step containing a passed data set abnormally terminates during execution, the passed data set is passed at the end of the job step. This allows you to receive and process the passed data set on a following job step (for example, when COND = EVEN or ONLY is coded). If the passed data set remains unreceived at the end of the job, then the conditional disposition (if specified) for the passed data set occurs.

**In a JES3 system:** If the data set was extended to additional volumes, code UNIT = AFF = ddname in the step to receive the data set. This makes JES3 aware of the additional unit requirement for the extended data set.

For additional information on JES3 allocation, see “JES3 Resource Allocation,” “Specifying Volume Affinity When Using Multivolume Data Sets,” and “JES3 Handling of Unit and Volume References.”

### **Disposition Processing of Passed Unreceived Data Sets**

A job step can pass a data set that is never received by a later step. If a job step abnormally terminates, unreceived data sets that specified a conditional disposition when passed are processed as specified in their conditional disposition, with four exceptions, as follows:

If the conditional disposition requires an update to a user catalog and:

- CATLG is specified for a data set that has a first-level qualifier of a catalog name or alias, the system does not catalog the data set.
- UNCATLG or DELETE (of a cataloged data set) is specified for a data set that has a first-level qualifier of a catalog name or alias, the system does not uncatalog the data set.
- CATLG is specified for a data set that does not have a qualifier or has a qualifier that is not a catalog name, the system catalogs the data set in the master catalog.
- UNCATLG or DELETE (of a cataloged data set) is specified for a data set that does not have a qualifier or has a qualifier that is not a catalog name, the system tries to uncatalog the data set from the master catalog.

Unreceived passed data sets that do not specify a conditional disposition, that is, those that were specified as (NEW,PASS) in this job, are deleted; all others are kept. The system deletes these data sets even if they have unexpired expiration dates or retention periods. (See “The RETPD or EXPDT Subparameters”)

If unreceived passed data sets are deleted at the end of a job, dynamic allocation is performed to allocate the unit and volume for deletion. If you have specified the second subparameter of the MSGLEVEL parameter (MSGLEVEL = (,1)) the system issues allocation messages for these data sets.

If no job step abnormally terminates before it begins execution, unreceived passed data sets that were specified as (NEW,PASS) are deleted; other data sets are kept.

If a step abnormally terminates before it actually begins execution (for example, during allocation of units and volumes or direct access space), the system ignores the disposition you code and again automatically keeps existing data sets and deletes new data sets.

For example, if you code:

```
DISP=( ,PASS,CATLG)
```

the system assumes the data set is new. If this step or any subsequent step prior to the step that receives this data set, abnormally terminates during its execution, the system tries to catalog the data set as instructed by the conditional disposition of CATLG. Any attempt by the system to catalog the data set is subject to the conditions discussed above.

## Default Disposition Processing

If you do not code the DISP parameter, or omit one of the subparameters, the system supplies default values.

If you do not specify a data set status, the system assumes NEW. If you do not code the second and third subparameters, the system determines how to handle the data set according to the status of the data set:

- Data sets that existed before the job are automatically kept (data sets for which OLD, SHR, or MOD is coded when volume information is available)
- Data sets created in the job are automatically deleted (data sets for which you coded NEW or MOD when volume information is not available, or for which you did not code a status)

## Bypassing Disposition Processing

If you define a data set as a dummy data set, the system ignores the DISP parameter (if coded), and does not perform disposition processing. For details on specifying dummy data sets, see “Defining a Dummy Data Set” on page 8-8.

## Insuring Data Set Integrity

When a job must receive control of the data sets it requests, you can request either exclusive control, allowing no other job to use the data set, or shared control, allowing the data set to be used by other jobs that also request shared control. The process of securing control of data sets for use by a job is called data set integrity processing.

Data set integrity processing avoids conflict between two or more jobs that request use of the same data set.

For example, two jobs, one named READ and another named MODIFY, both request the data set FILE.

- READ wants only to read and copy certain records
- MODIFY deletes some records and changes other records in the data set FILE

If both jobs have control of FILE concurrently, READ cannot be certain of the records contained in FILE; that is, READ cannot be certain of the integrity of the data set.

- MODIFY should have exclusive control of the data set
- READ can share control of FILE with other jobs that also want only to read the data set.

Indicate the type data set control a job requires in the DISP parameter on the DD statement defining the data set.

### Exclusive Control of a Data Set

When a job has exclusive control of a data set, no other job can use that data set until termination of the last **step** in the job that refers to the data set. A job should have exclusive control of a data set in order to modify, add, or delete records.

In some cases, you may not need exclusive control of the entire data set. You can request exclusive control of a block of records by coding the DCB, READ, WRITE, and RELEX macro instructions. (These instructions are described in *Data Management Macro Instructions*.)

To request exclusive control of a data set, you code NEW, OLD, or MOD as the first subparameter of the DISP parameter.

### Shared Control of a Data Set

Special jobs can concurrently use a data set on a direct access storage device, if these jobs request shared control of the data set; however, none of the jobs should change the data set in any way.

To request shared control, code SHR as the first subparameter in the DISP parameter. If more than one step of your job requests a data set, you must code SHR on every DD statement that defines the data set if it is to be used by concurrently executing jobs.

### How the MVS System Performs Data Set Integrity Processing

The system performs data set integrity processing once for each job, for the following types of data sets:

- Nontemporary data sets, and
- Non-VIO temporary data sets (see “Using Virtual Input/Output (VIO) for Temporary Data Sets.”)
- Data sets with alias names (created with the access method services DEFINE command; see *Access Method Services*).
- Members of generation data groups

The system **does not** perform data set integrity processing for subsystem data sets.

To secure control for all **nontemporary data sets** for the job, the system enqueues each data set, marking the data set as requested by that job and noting what kind of control was requested. A job can request either shared or exclusive control for a data set. The system assigns control of the data set until the termination of the last step in the job that refers to that data set occurs.

If you code NEW, OLD, or MOD on any reference to a data set, the system assigns **exclusive** control. A reference requesting exclusive control overrides any number of references requesting shared control.

The job **receives** control of the data set if:

- Another job is not using the data set, or
- Another job is using the data set but both the job requesting the data set and the job using the data set request shared control and there are no exclusive requests pending.

The job does **not receive** control of a data set if:

- Another job is using the data set and that job has exclusive control, or
- Another job is using the data set (with either exclusive or shared control), and the job requesting use of the data set requests exclusive control, or
- Another job is using the data set (with shared control) and there is yet another job that requested exclusive control of the data set prior to this job.

If a job requests data sets that are not available, the system issues the message “JOB jjj WAITING FOR DATA SETS” to the operator. The initiator that started the job automatically waits until the required data sets become available unless the operator cancels the job.

When the system has secured control of all nontemporary data sets, it allocates and deallocates resources for each step of the job. The job terminates after the system has deallocated all resources for the last step in the job.

Non-VIO temporary data sets, data sets with alias names, and members of generation data groups are reserved or enqueued for each step within the job. The job receives control of the data set for that step in the same manner as described for nontemporary data sets.

When a job is executing and it requires a non-VIO temporary data set, a data set with alias names or a member of a generation data group, if the job cannot secure control for the data set, the job fails. (The system cannot wait for data sets at this point: the job already owns certain resources and waiting for other resources could create a possible deadlock.)

When each step terminates, the system releases control of any data sets (except non-VIO temporary data sets) that are not used in any subsequent step of the job. The system releases control of all other data sets and terminates the job upon completion of the last step in the job.

The following table summarizes data set integrity processing.

	Data Set is currently in use as:		Data Set not in use	Data Set Has previous request for:	
	Shared	Exclusive		Shared	Exclusive
<b>Nontemporary Data Set Requesting:</b>					
Shared Control	Note: 1	Note: 2	Note: 1	Note: 1	Note: 2
Exclusive Control	Note: 2	Note: 2	Note: 1	Note: 2	Note: 2
<b>Non-VIO Temporary Data Set Requesting:</b>					
Shared Control	Note: 1	Note: 3	Note: 1	Note: 1	Note: 3
Exclusive Control	Note: 3	Note: 3	Note: 1	Note: 3	Note: 3
<b>GDG Requesting:</b>					
Shared Control	Note: 1	Note: 3	Note: 1	Note: 13	Note: 31
Exclusive Control	Note: 3	Note: 3	Note: 1	Note: 3	Note: 3
<b>Data Sets with Alias Names Requesting:</b>					
Shared Control	Note: 1	Note: 3	Note: 1	Note: 1	Note: 3
Exclusive Control	Note: 3	Note: 3	Note: 1	Note: 3	Note: 3
<p>Note 1: The requested control will be granted.</p> <p>Note 2: The requested control will not be granted until the data set is released by the job that has control or the data set is released by the job that has previously requested control.</p> <p>Note 3: The requested control will not be granted and the job is terminated at the step requesting that data set.</p>					



## Examples of Disposition Processing of Non-VSAM Data Sets

```
//DISP JOB MSGLEVEL=1
//S1 EXEC PGM=IEFBR14
//D1 DD DSN=ABC,DISP=(SHR,KEEP)
//D2 DD DSN=SYSA,DISP=(OLD,DELETE,UNCATLG)
//D3 DD DSN=SYSB,UNIT=3350,VOL=SER=335001,
// SPACE=(CYL,(4,2,1)),DISP=(NEW,KEEP,CATLG)
//D4 DD DSN=&&SYS1,DISP=(MOD,PASS),UNIT=3350,
// VOL=SER=335004,SPACE=(TRK,(15,5,1))
//S2 EXEC PGM=IEFBR14
//D1 DD DSN=&&SYS1,DISP=(MOD,DELETE),UNIT=3350,
// VOL=SER=335004,SPACE=(TRK,(15,5,1))
```

1. The JOB statement requests that all JCL statements and system messages be printed.
2. D1 in S1 defines a data set that already exists and can be shared with other data sets. It is to be kept on the volume after this job step.
3. D2 in S1 defines a data set that already exists, cannot be shared with other data sets, is to be deleted at the end of the job step, and is to be uncataloged if the program abnormally terminates.
4. D3 in S1 defines a new data set that is to be assigned a specific volume (335001) on a 3350 device. The data set is to be kept on the volume at the end of this job step for normal processing but is to be cataloged if the program abnormally terminates.
5. D4 in S1 defines a temporary data set that is to be created in this job step. It is to be assigned to volume 335004 on a 3350 device with the space request of 15 primary tracks, five secondary, and a directory. This data set is to be passed for subsequent use by a job step in this job.
6. D1 in S2 defines the same temporary data set that was defined in D4 of S1. When this step is completed, the data set is to be deleted.

---

```
//PASS JOB MSGLEVEL=1
//S1 EXEC PGM=IEFBR14
//DD1 DD DSN=A,DISP=(NEW,PASS),VOL=SER=335000,
// UNIT=3350,SPACE=(TRK,1)
//DD2 DD DSN=A,DISP=(OLD,PASS),VOL=REF=*.DD1
//DD3 DD DSN=B,DISP=(OLD,PASS),VOL=SER=335000,UNIT=3350
//DD4 DD DSN=B,DISP=(OLD,PASS),VOL=SER=335001,UNIT=3350
//S2 EXEC PGM=IEFBR14
//DD5 DD DSN=A,DISP=OLD
//DD6 DD DSN=A,DISP=OLD
//DD7 DD DSN=B,DISP=OLD
//DD8 DD DSN=B,DISP=(OLD,PASS)
//S3 EXEC PGM=IEFBR14
//DD9 DD DSN=B,DISP=OLD
```

1. DD1 and DD2 pass the same data set. DD5 and DD6 receive that same data set.
2. DD3 and DD4 pass different data sets of the same name, DD7 receives the data set passed by DD3 and DD8 receives the one passed by DD4. DD8 also continues to pass the data set originally passed by DD4.
3. DD9 receives the data set passed by DD4 and DD8.

## Requesting Units and Volumes

On the DD statement defining a data set, indicate the device and volume on which the data set can be found or will be written by specifying unit and volume information. Input/output devices are grouped according to class; a device class refers to a kind of device: direct access, magnetic tape, unit record, graphic, and communications equipment. A **unit** is a particular device: a 3350 direct access device, a 1403 printer, etc.; a **volume** is a section of auxiliary storage that is serviced by a single read/write mechanism — for example, a reel of magnetic tape, a drum, or a disk pack.

Device status can affect the device eligibility for allocation. Figure 7-1 shows the various devices and the possible status each may have.

Status	Device Type				
	Direct Access	Tape	Unit Record	Graphic	Teleprocessing
Online	Eligible for allocation				
Offline	Eligible for allocation when the operator brings device online			Eligible for allocation	
Pending Unload	Eligible for allocation when volume is specifically requested		Not applicable		
Pending Offline	Eligible for allocation when the operator brings the device online and when the volume is specifically requested		Eligible for allocation when the operator brings the device online		Not applicable

Figure 7-1. How Device Status Affects Eligibility for Allocation

## Specifying Volume Information

Data sets exist on direct access and magnetic tape volumes that must be mounted on devices before they can be used. To tell the system on which volume an existing data set can be found, make a specific volume request; to create a new data set, make a specific or nonspecific volume request. If you request multiple disk volumes to be mounted in JES3, they must all be either mountable or permanently resident; a mixture of both is not allowed.

### Specific Volume Requests

A **specific volume request** informs the system of the volume serial number of the volume required. A request for an existing data set implies request for a specific volume. Make either a **specific** or **nonspecific** volume request when creating a data set.

You are making a specific request when:

- You specify the serial numbers in the SER subparameter of the VOLUME parameter; for example, VOL = SER = (948762,945231).
- You refer the system to an earlier specific volume request to copy the volume serial numbers.

You do this by coding the name of a passed or cataloged data set or a previous DD statement in the REF subparameter of the VOLUME parameter.

- To refer the system to a passed or cataloged data set, code VOL = REF = dsname
  - To refer to a DD statement in the same step, code VOL = REF = \*.ddname
  - To refer to a DD statement in a preceding step, code VOL = REF = \*.stepname.ddname
  - To refer to a DD statement in a procedure step that is in a procedure called by a preceding step, code VOL = REF = \*.stepname.procstepname.ddname. (If you refer to a multi-device type VSAM data set, the system uses only the volume serial number of the first device type listed.)
- You pass the data set from an earlier step or you reference the data set from the catalog.

The system obtains the volume serial numbers from the passed data set information or from the catalog; you need not code the VOLUME parameter unless requesting a private volume, coding a volume sequence number, or requesting additional volumes. If a cataloged data set is cataloged in, or is to be cataloged or uncataloged from, a private catalog other than JOBCAT and STEPCAT, then the system automatically allocates that private catalog to the job step. (The private catalog must be on a permanently resident volume for JES3). If this allocation is not successful, the job fails.

### How the System Satisfies Specific Volume Requests

A specific volume request informs the system of the volume serial number of the volume required. In the following cases the system can satisfy a request for a specific volume that is already mounted:

- The volume is permanently resident or reserved. (The volume is assigned regardless of the requested use attribute, and the use attribute is not changed by the allocation.)
- The direct access volume is a removable volume that can be shared and is being used by a concurrently executing step. (If your request would make the volume unable to be shared, the system will assign you that volume only when all other job steps using the volume have terminated.)
- The direct access volume is removable but not allocated. The use attribute (private or public) assigned to the volume when it is allocated is determined by the presence or absence of the PRIVATE subparameter.
- The tape volume is a scratch volume and is not in use. The use attribute of private is assigned to the volume if the request is for a permanent data set or if the PRIVATE subparameter is coded.

## Nonspecific Volume Requests

**Nonspecific volume requests** can be made only for new data sets. When you make a nonspecific volume request, do not specify volume serial numbers. You need not code the **VOLUME** parameter unless you are requesting a private volume or a volume count.

You can make four types of nonspecific volume requests:

- A private volume for a temporary data set
- A private volume for a nontemporary data set
- A nonprivate volume for a temporary data set
- A nonprivate volume for a nontemporary data set

How the system satisfies these different types of requests is described below. Since the system satisfies the first two types of requests in the same way, these two requests are described together.

- When you make a nonspecific volume request for a private **direct access or tape volume**, the system always requests the operator to mount a volume. The operator should mount a volume whose space is unused. This allows you to have control over all space on the volume. Once mounted, the volume is assigned the use attribute of private.
- When you make a nonspecific volume request for a nonprivate **direct access volume** that is to contain a temporary data set, the system assigns a public or storage volume that is already mounted, or if no space is available, it requests the operator to mount a removable volume. If the system selects a mounted volume, its use attribute is not affected. If a removable volume is mounted, it is assigned the use attribute of public. For the definition of the **MOUNT** and **USE** attributes, see *SPL: Job Management*.
- When you make a nonspecific request for a nonprivate tape volume, data management (**OPEN**) will satisfy this request by using any available, physically mounted, tape volume. This could result in the loss of user data. However, if you use labels on your tape volumes and specify the types of labels in the **LABEL** parameter, loss of data can usually be prevented.
- When you specify labels in the **LABEL** parameter, data management (**OPEN**) checks the first record of the tape. There are various error conditions that can occur during verification of the first record. These error conditions are described in *Tape Labels*.
- When you make a nonspecific volume request for a nonprivate **tape volume** that is to contain a temporary data set, the system assigns a public or scratch volume that is already mounted, or it requests the operator to mount a tape volume. Once mounted, the volume is assigned the use attribute of public.
- When you make a nonspecific volume request for a nonprivate **direct access volume** that is to contain a nontemporary data set, the system assigns a storage volume if one is mounted. Otherwise, the system treats the request as a nonspecific volume request for a private volume.
- When you make a nonspecific volume request for a nonprivate **tape volume** that is to contain a nontemporary data set, the system treats the request as a nonspecific volume request for a private volume.

*Note:* If your nonspecific volume request requires more than one unit from a group that contains both single and dual density tape drives, the system assigns the devices so that the single density drive is the first one used. The default density is the density of the single density drive. The operator may be requested to mount the volumes in a different order than assigned by the system.

## Using Private Volumes

A **private volume** is one that can be used only by those who know the serial number. Code **PRIVATE** as the first subparameter in the **VOLUME** parameter for both specific and nonspecific volume requests. When you make a specific volume request for a direct access volume, code **PRIVATE** if you want a private volume; tape volumes for which you make a specific volume request are automatically made private, so you need not code the **PRIVATE** subparameter.

A volume already made private cannot be allocated to satisfy other nonspecific volume requests. Therefore, if you request a private volume, you will be the only user using that volume, unless another job makes a specific volume request for that volume.

If **PRIVATE** is coded or implied for a **direct access volume**, the operating system requests that the operator demount the volume at job termination.

If **PRIVATE** is coded or implied for a **tape volume**, the operating system automatically requests that the operator demount the volume after its last use in the job step unless **RETAIN** is coded or the data set is passed. If you expect to use a data set in a subsequent step for which you requested a private volume, code **RETAIN** in the **VOLUME** parameter to ensure that the volume is not demounted at the end of the step. Even if you specify **RETAIN** or a disposition of **PASS**, the operator can still unload the volume or another step in the same job or another job can allocate and demount it.

## Sharing Volumes Between Data Sets

To use fewer volumes, request that data sets be assigned the same volume. Data sets on the same volume have **volume affinity**.

You can request volume affinity either:

- Implicitly, through catalog references or by specifying the same volume serial numbers for different data sets in the **SER** subparameter of the **VOLUME** parameter.
- Explicitly, by using the **REF** subparameter of the **VOLUME** parameter to indicate that volumes identified in the catalog or on an earlier **DD** statement in the job are to be assigned to the data set being defined.

Volume affinity influences the allocation of devices. The system can modify a request for a specific number of units if a data set has volume affinity with at least one other data set. For examples of volume affinity, see “Example of **UNIT** and **VOLUME** Affinities” at the end of this section.

## Multivolume Data Sets

If you are creating or extending a data set that can require more than one volume, request the maximum number of volumes required in the **volume count** subparameter of the **VOLUME** parameter. The maximum number of volumes you can request is 255. For some jobs, each volume requested must be mounted on a unit before it can be used. For these jobs, be sure to request as many units as volumes. When you make a specific volume request for more volumes than units, the system automatically indicates that the volumes on the same unit cannot be shared.

By coding the volume sequence number subparameter when reading or lengthening an existing multivolume data set, you can instruct the system to begin processing other than the first volume. Usually a volume sequence number is coded when you are defining an existing cataloged or passed data set.

## Specifying Unit Information

Use the **UNIT** parameter to provide the system with the information it needs to assign a device to a data set. To indicate what unit or type of unit you want, code one of the following:

- Unit address
- Device type (generic name)
- User-assigned group name (esoteric name)

The **unit address** is a 3-character address made up of the channel, control unit, and unit number. For example, **UNIT=180** indicates channel 1, control unit 8, and unit number 0. Specifying a unit address, however, limits unit assignment: the system can assign only that specific unit, and, if the unit is being used, the job must be delayed or canceled. Only specify unit addresses when necessary since these specifications restrict the system.

A **device type** corresponds to a particular set of features of input/output devices. When coding a device type, you allow the system to assign any available device of that device type. For example, **UNIT=3350** indicates that you want the system to assign any available 3350 disk storage facility.

For additional information on specifying device types, see *System Generation Reference*.

During system generation, each installation can also define **user-assigned group names** to signify a group of devices that may or may not all be of the same type. By coding a user-assigned group name, you allow the system to assign any available devices included in the group. For example, if the group named **DISK** includes all 3350 and 3330 disk storage facilities and you code **UNIT=DISK**, the system assigns an available 3350 or 3330 device. If the group named **3350A** includes particular 3350's and you code **UNIT=3350A**, the system could assign one of several 3350 devices.

If the group consists of more than one device type, and more than one unit is requested, the units are allocated from the same device type. For example, if the group named **TAPE** includes both 3400-5 and 3400-6 devices, and you request two units by specifying **UNIT=(TAPE,2)**, the system assigns either two 3400-5s or two 3400-6s. If there is an insufficient number of units of any single type to satisfy the request, the job is flushed.

If a group contains more than one device type or class (for example, **SYSSQ** can refer to all tape and direct access devices), you should not code the group name when defining an existing

data set or requesting a specific volume. The volume on which the data set resides may require a device different from the one assigned to it. For example, if the data set resides on a tape volume, it must be assigned to a tape device.

The same is true if the data set resides on a 3348 Model 70F Data Module and the group name includes 3340 drives with and without the Fixed Head Feature. The 3348 Model 70F must be assigned to a 3340 with the feature. For more information on the Fixed Head Feature, see the *IBM 3340 Fixed Head Feature User's Guide*.

Only direct access devices can be simultaneously allocated for two or more jobs. Teleprocessing equipment is not allowed to be allocated more than once in the same job step. If a unit record, teleprocessing equipment, or graphics device is designated as a console, it is not eligible for allocation by a job.

### Relationship of a UNIT Specification to System Generation

Installation programmers use the IODEVICE system generation macro instruction to describe the characteristics and requirements of a device to the system. Each I/O device at your installation is described in an IODEVICE macro instruction. The UNIT parameter of the IODEVICE macro instruction describes the *device type* (generic name) that you code in the UNIT JCL parameter.

For example, when you code UNIT = 3350 in your JCL, you are coding the same specification that your installation programmer coded in the UNIT parameter of the IODEVICE macro instruction at system generation.

Installation programmers use the UNITNAME system generation macro instruction to specify a name for a group of devices. The NAME parameter of the UNITNAME macro instruction defines the *user-assigned group name* (esoteric name) that you code in the UNIT JCL parameter.

For example, when you code UNIT = DISK, you are coding the same specification that your installation programmer coded in the NAME parameter of the UNITNAME macro instruction at system generation.

For additional information on the IODEVICE and UNITNAME system generation macro instructions, see *System Generation Reference*.

### Requesting More than One Unit

To increase operating efficiency, request multiple units for a multivolume data set or for a data set that may require additional volumes. When each required volume is mounted on a separate device, execution of the job step is not interrupted to allow the operator to demount and mount volumes. You should always request multiple units when the data set can be extended to a new volume when the data set resides on a permanently resident or reserved volume. Permanently resident and reserved volumes cannot be demounted in order to mount a new volume.

Request multiple units by:

- Coding the unit count subparameter in the UNIT parameter
- Requesting parallel mounting

To request **parallel mounting**, code P in place of the unit count subparameter when you make a specific or nonspecific volume request. The system counts the number of volumes requested (by

counting the volume serial numbers specified on the DD statement or counting the volume serial numbers in cataloged or passed data sets).

It compares this sum to the volume count, if specified, and the system assigns the larger of the specified number of devices.

### Deferred Mounting of Volumes

A job step may include a data set that your program might not use. Use the DEFER subparameter to request that the system not mount the volume containing the data set until the data set is opened. This can save the operator the time it takes to mount volumes on direct access devices.

*Note:* No other job step can use such a volume until the job step specifying DEFER ends. If you code DEFER for a new data set that could be placed on a direct access device, then the system ignores DEFER.

### Relationship of the UNIT Parameter to a Volume Reference

The system can obtain unit information from sources other than the UNIT parameter. In many such cases, you do not have to code the UNIT parameter. However, if coding VOL = SER = serial-number or VOL = REF = reference, you should know where the system obtains the unit information.

Normally, you do not have to code unit information when the data set is cataloged. For cataloged data sets, the system obtains unit and volume information from the catalog. This is true **except** when you code VOL = SER = serial-number on a DD statement that defines a pre-existing data set. When you code VOL = SER = serial-number, the system does not look in the catalog — you **must** code the UNIT parameter.

You can override the data set name on a procedure DD statement with the data set name of a cataloged data set. If you do so but do not override the UNIT parameter on the DD statement in the procedure, the system will not search the catalog for unit information. Instead, the system will obtain unit information from the overridden DD statement. Therefore, when you override the data set name on a procedure DD statement, nullify the unit parameter on the procedure DD statement and the system will search the catalog for unit information.

Normally, when the data set is passed from a previous job step, you do not have to code unit information.

For passed data sets, the system obtains unit and volume information from passed data set information. This is true **except** when you code VOL = SER = serial-number on a DD statement that defines a pre-existing data set. When you code VOL = SER = serial-number on a DD statement that defines a pre-existing data set, the system does not look in the passed data set information — you **must** code the UNIT parameter.

You do not have to code unit information when the data set is to use the same volumes assigned to an earlier data set. You can code VOLUME = REF = ddname. In this case, the system obtains unit and volume information from:

- the earlier DD statement that specified the volume serial number, or
- the catalog.



When you want additional devices assigned or when you want to influence device allocation code the UNIT parameter. The system uses the coded UNIT parameter if it is a subset of the unit type referenced. Otherwise, the system ignores it.

Do not code the UNIT parameter when defining a data set included in the input stream. If UNIT is coded on a DD \* or DD DATA statement, MVS terminates the job.

### JES3 Handling of UNIT and VOLUME references

When JES3 looks in the catalog, it cannot determine whether or not a given device type is a subset of another device type. Errors might result if you request a device to be mounted on a conflicting device type (for example, a 3330 mounted on a 3350). To avoid this error, use the JES3 HWSNAME initialization statement to define to JES3 which device names are subsets of other device names.

If, in a multiple-step job, a data set is extended to additional volumes, and if a subsequent step allocates that data set, MVS allocates the additional units the job requires. JES3 is unaware of the additional unit requirement; therefore code UNIT=AFF=ddname in the last step to allocate the extended data set.

For additional information on JES3 allocation, see “Allocating Data Resources in a JES3 System” on page 6-4 and “Specifying Volume Affinity When Using Multivolume Data Sets” on page 7-35.

### Example of Requesting Units and Volumes

This job shows the unit and volume parameters.

```
//TEST      JOB  WIBORG,CLASS=C
//STEP1     EXEC PGM=TESTSYSO
//DD11      DD   DSN=A01DD1,UNIT=3330,DISP=(,PASS),
//          SPACE=(TRK,1),VOL=SER=333001
//STEP2     EXEC PGM=TESTSYSO
//DD21      DD   DSN=SYSLIB,UNIT=3350,VOL=(PRIVATE,SER=123456),
//          DISP=OLD
//DD22      DD   DSN=SYSABC,UNIT=AFF=DD21,VOL=SER=777777,
//          DISP=(OLD,KEEP)
//DD23      DD   DSN=SYSTAPE,UNIT=(3400-5,P,DEFER),DISP=OLD,
//          VOL=SER=(342001,342002,342003,342004,342005)
//DD24      DD   DSN=SYSDISK,DISP=(SHR,KEEP),UNIT=(,P),
//          VOL=SER=(333005,333008,333010)
//DD25      DD   UNIT=3350,VOL=REF=*.DD21,SPACE=(TRK,(5,2))
//DD26      DD   UNIT=3330,VOL=REF=SYSDISK,SPACE=(TRK,(10,5))
```

1. The job is assigned to class C.
2. DD11 defines a new data set named A01DD1. It is to be on volume 333001 which is mounted on a 3330 device.
3. DD21 defines an old data set named SYSLIB that exists on a private volume, 123456. The volume is mounted on a 3350 device.
4. DD22 defines an old data set named SYSABC that is to be kept after this job step is complete. SYSABC is on volume 777777. This volume is to be mounted on the same 3350 device as the volume defined on DD21.

5. DD23 defines an old data set named SYSTAPE. There are five volumes that are to be mounted only after the data set is opened (caused by the DEFER subparameter). The P requests parallel mounting; that is, all five volumes are to be mounted at the same time on five different 3400-6 devices.
6. DD24 defines an old data named SYSDISK that can be shared by another job since it will only be read. It is to be kept after this job step. The number of units used is determined by the number of volumes requested.
7. DD25 is a temporary data set (no DSNAME specified) and, therefore, assumes a disposition of NEW,DELETE. The volume to be used is the same one used in STEP2 DD21; that is, volume 123456.
8. DD26 is also a temporary data set. The backward reference for volume information is to STEP2 DD24 where the data set named SYSDISK is located.

### Sharing a Unit Between Data Sets on Different Volumes

You can conserve the number of devices used in a job step. To do so request that an existing data set be assigned to the same device or devices as assigned to a data set defined earlier in the job step. When two or more volumes are assigned the same device, the volumes are said to have **unit affinity**. Unit affinity implies deferred mounting for all except one of the volumes, since all volumes cannot be mounted on the same device at the same time.

Request explicit unit affinity by coding `UNIT=AFF=ddname` on a DD statement.

The `ddname` is the name of an earlier DD statement in the same job step. The data set defined on the DD statement that requests unit affinity is assigned the same device or devices as the data set defined on the named DD statement; the data set must reside on the same device type. If the `ddname` refers to a DD statement that defines a dummy data set, the data set defined on the DD statement requesting unit affinity is assigned a dummy status.

Unit affinity also exists on one DD statement when there are more volumes than units. This is implied unit affinity. See examples of unit affinity.

If all of the following conditions are present, the data set defined on the DD statement requesting unit affinity might be written over by the named data set:

- The named DD statement requests a scratch tape.
- The data set defined on the DD statement requesting unit affinity is opened prior to that on the named DD statement.
- The tape is not unloaded prior to the OPEN of the data set defined on the named DD statement and tape label positioning is not specified using the LABEL parameter. (Note that a tape unit that is allocated to more than one request is not unloaded (1) as a result of dynamic deallocation, or (2) when it is closed if `FREE=CLOSE` is specified.)

*Note:* You cannot request unit affinity for a new data set if the original request is for a direct access device.

**Unit and Volume Affinities:** It is possible to have unit affinity, volume affinity, and/or unit and volume affinity occurring in the same step. You can also have unit and volume affinity occurring on the same DD statement (see example 2, below). However, not all combinations are possible.

- **Explicit unit affinity** is requested when UNIT = AFF = ddname is specified.
- **Implied unit affinity** is requested when more volumes than units are specified on the same DD statement.
- **Volume affinity** is requested when two or more DD statements reference the same volume.

### Examples of Unit and Volume Affinity

The examples below illustrate the three kinds of relationships possible between unit and volume affinity.

1. All volume affinity requests are unrelated to any of the unit affinity requests. For example,

```
//DD1 DD VOL=SER=A,UNIT=3330
//DD2 DD UNIT=AFF=DD1,VOL=SER=B
//DD3 DD VOL=SER=(C,D),UNIT=3330
//DD4 DD VOL=SER=C,UNIT=3330
```

- Unit affinity is explicitly requested between DD1 and DD2.
- Volume affinity is implicitly requested between DD3 and DD4.
- Therefore, the volume affinity request is not related to the unit affinity request.

2. All volume affinity requests are contained in the unit affinity requests. For example,

```
//DD1 DD VOL=SER=(A,D),UNIT=3330
//DD2 DD UNIT=AFF=DD1,VOL=SER=(A,B)
//DD3 DD VOL=SER=X,UNIT=3330
```

- Unit affinity is explicitly requested between DD1 and DD2.
- DD1 illustrates implied unit affinity within a DD statement because the unit count defaults to one.
- Volume affinity is also implicitly requested between DD1 and DD2.
- Therefore, the volume affinity request is contained within the unit affinity request.

3. Some volume affinity requests are contained in the unit affinity requests, but not all. For example,

```
//DD1 DD VOL=SER=A,UNIT=3330
//DD2 DD UNIT=AFF=DD1,VOL=SER=B
//DD3 DD VOL=SER=B,UNIT=3330
```

- Unit affinity is explicitly requested between DD1 and DD2.
- Volume affinity is implicitly requested between DD2 and DD3.
- Therefore, some volume affinity requests are contained within the unit affinity requests, but not all.

If both unit and volume affinity do exist in the same step, sometimes only one requested affinity can be honored at a time. Figure 7-2 on page 7-34 indicates what will happen when you code unit and volume affinity for either tape or direct access devices.

Relationship of unit and volume affinity requests	Tape	Direct Access
Unit and volume affinity requests unrelated	Because there is no conflict, both unit and volume affinity requests are honored.	Because there is no conflict, both unit and volume affinity requests are honored.
All volume affinity requests contained in unit affinity requests.	All volumes will use the same unit; that is, volume affinity is ignored and unit affinity is honored.	For those volumes having volume affinity that are contained in the unit affinity requests, unit affinity is ignored. That is, they will share the same unit while the remaining requests in the unit affinity will use a different unit.
Some volume affinity requests contained in unit affinity requests.	For those volumes having volume affinity that are contained in the unit affinity requests, unit affinity is ignored. That is, they will share the same unit while the remaining requests in the unit affinity will use a different unit.	For those volumes having volume affinity that are contained in the unit affinity requests, unit affinity is ignored. That is, they will share the same unit while the remaining requests in the unit affinity will use a different unit.

Figure 7-2. Unit and Volume Affinity

*Note:* If a requested volume is mounted on an eligible, permanently-resident or reserved unit, it must be allocated to that unit regardless of any relationships to other requests. This is done because that particular volume **cannot** be dismounted.

### Positioning the Unit Affinity Request

If unit affinity (UNIT=AFF=ddname) to a DD statement is requested before the ddname is defined within the job stream, the system treats the requesting DD statement as a DUMMY DD.

For example;

```
//STEP EXEC PGM=TKM
//DD1 DD DDNAME=DD5
//DD2 DD DSN=A,DISP=OLD
//DD3 DD UNIT=AFF=DD1
```

The system treats DD3 as a DUMMY DD.

## Specifying Volume Affinity When Using Multivolume Data Sets

Do not request volume affinity for multivolume data sets without also requesting unit affinity.

When you specify volume affinity for multivolume data sets using `VOL=REF=*.ddname`, without specifying unit affinity, the system allocates a unit that is shared between all the DD statements involved in the volume affinity request. The system then initiates a mount request for the first volume serial number associated with the referenced DD statement.

Because volume affinity requests are not related to unit affinity requests, `OPEN/CLOSE/END-OF-VOLUME (O/C/EOV)` look ahead mounting or end of volume processing may cause this volume to be demounted and remounted on a unit other than the one that was originally allocated for the volume affinity request. Your job now goes into a wait because the system has requested the same volume on two different units.

To enable `O/C/EOV` to find the volume when such a remount occurs, always specify volume affinity *and* unit affinity for multivolume data sets.

## Example of UNIT and VOLUME Affinities

The purpose of this job is to show several job steps that use either unit or volume affinity for their processing.

```
//AFFIN JOB (8526,831),WOON,CLASS=J,PERFORM=50
//STEP1 EXEC PGM=TESTAFF
//DD1 DD UNIT=3400-5,VOL=SER=111111
//DD2 DD UNIT=AFF=DD1,VOL=SER=222222
//STEP2 EXEC PGM=TESTAFF
//DD11 DD UNIT=(3330,2),VOL=SER=(A,B)
//DD12 DD UNIT=AFF=DD11,VOL=SER=(C,D)
//STEP3 EXEC PGM=TESTAFF
//DD21 DD UNIT=(3330,2),VOL=SER=(A,B)
//DD22 DD UNIT=AFF=DD21,VOL=SER=(C,D)
//DD23 DD UNIT=3330,VOL=SER=B
//STEP4 EXEC PGM=TESTAFF
//DD31 DD UNIT=(3330,2),VOL=SER=(E,F)
//DD32 DD UNIT=AFF=DD31,VOL=SER=(G,H)
//STEP5 EXEC PGM=TESTAFF
//DD41 DD UNIT=3400-5,VOL=SER=(111111,222222)
//DD42 DD UNIT=AFF=DD41,VOL=SER=(222222)
//STEP6 EXEC PGM=TESTAFF
//DD51 DD UNIT=3330,VOL=SER=(ABCDEF,GHIJKL)
//DD52 DD UNIT=AFF=DD51,VOL=SER=(ABCDEF)
//STEP7 EXEC PGM=TESTAFF
//DD61 DD UNIT=3400-5,VOL=SER=111111
//DD62 DD UNIT=3400-5,VOL=SER=111111
//DD63 DD UNIT=AFF=DD61,VOL=SER=222222
```

1. The `JOB` statement assigns jobs to class J in performance group 50.
2. `STEP1` assigns one unit for both volumes. Volume 111111 will be mounted first, then 222222 will be mounted when DD2 is opened. (This processing is true for both tape and direct access.)
3. `STEP2` allocates two units to DD11 and volumes A and B are mounted. DD12 gets allocated to the same two units but volumes C and D will be mounted when DD12 is opened.

4. STEP3 is a direct access example of volume affinity for volume B. The actual allocation of units will cause volumes A and C to share one unit and volumes B and D to have their own units.
5. STEP4 is a direct access example. Assume that volume E is currently mounted and has been assigned the permanently resident or reserved attribute. In this case, since volume E cannot be dismounted, a separate unit will be allocated for it. Volume G will have its own unit and volumes F and H will share one unit. Therefore, three volumes will be allocated for these requests, instead of two, because of the permanently resident or reserved mount attributes.
6. STEP5 is a tape example. Volume affinity is ignored between the DD statements because only one tape data set for each tape volume can be open at a time.
7. STEP6 is a direct access example where unit affinity is ignored for the common volume. Volume ABCDEF of both DD statements will share the same unit while the remaining request (GHIJKL) will use a different unit.
8. In STEP7, unit affinity is requested between DD61 and DD63. Volume affinity is requested between DD61 and DD62. Because there is a volume affinity request (DD62) that is not contained in the chain of unit affinity requests, the UNIT=AFF=DD61 specification is ignored for DD63. STEP7 allocates two units; one for volume 111111, and another for volume 222222.

### Volume Attributes

MVS assigns attributes to every mounted volume. This discussion describes the relationship of JCL parameters and the disposition of removable tape volumes.

Note that 3330V or MSS volumes, while ultimately recorded on tape-like media, are treated as DASD volumes by MVS.

This discussion is not applicable to JES3-managed units.

The JCL disposition parameter (DISP) refers to a **data set's** disposition, whereas the disposition of a **tape volume** is influenced by the volume's **mount and use attributes**. For further information see *Job Management*. This discussion centers on tape volumes with the **removable mount attribute**. That is, those volumes that are not mounted in response to the operator MOUNT command. The use attributes for tape volumes are **PRIVATE** and **PUBLIC**.

MVS assigns the use attribute of **PRIVATE** if **any** one of the following is true.

- You specify the **PRIVATE** subparameter of the **VOLUME** parameter, or
- You make a specific volume request, or
- The data set is a nontemporary data set; that is, it does not have a system-generated data set name, or it has a data set disposition other than **DELETE**.

MVS assigns the use attribute of **PUBLIC** when **all** of the following are true.

- You do **not** specify the **PRIVATE** subparameter of the **VOLUME** parameter, and
- You do not request a specific volume, and

- The data set is a temporary data set; that is, you have allowed the system to generate the data set name or you have specified a disposition of DELETE.

Ultimately a tape volume has one of two dispositions; Keep (K) or Scratch (D). However, situations arise in which a tape volume must be demounted before a job is “finished” with that volume so that the drive may be used to mount a different volume.

For example, this can occur when a volume is used by job steps 1 and 3 but not by job step 2 in a three-step job.

In this instance the system assigns the volume the *RETAINED* (R) designation, in effect instructing the operator to place the tape nearby for possible later use.

In a multiple step job, if there is a period when a volume is not in use, you can request the system to attempt to keep the volume mounted by coding the RETAIN subparameter on the VOLUME parameter.

The designation R causes the system to “remember” that a unit contains such a passed or retained volume.

When a tape unit is deallocated, for example, at the end of a job step, PRIVATE volumes that are not passed or retained are demounted with a volume disposition of keep (K).

We define a private volume as one that is usable only by those who know its volume serial number. The system cannot return the volume to the scratch pool because it has no way of “knowing” that the users have relinquished the volume serial number.

By definition PUBLIC volumes are available for use by any user and therefore should remain mounted on units that are no longer in use by a particular job. This enables any subsequent job to use them. When these volumes cannot remain on the unit they are currently mounted on (because the unit is being allocated), the volume is demounted with a D (scratch) designation. This would occur when the system requests that the operator mount a specific volume on that particular unit.

There are situations where the use attribute can change from private to public and from public to private, in these cases a demount may not take place. For a discussion of when the use attributes can change, see *SPL: Job Management*.

## Specifying Data Sets for Mass Storage Systems (MSS)

Mass storage volumes are accessed on virtual direct access devices. All previous descriptions of direct access device resource requests apply, with several additional functions also available. The mass storage volume device type is 3330V.

### Mass Storage Volume Groups

The mass storage system (3850) can contain up to 4,720 mass storage volumes (3330V). To assist the installation in managing the volumes, the mass storage system utilities are used to assign the volumes to groups. When creating a new data set with a nonspecific request, the desired group can be specified using MSVGP=id. The system then selects the best volume for the requirements from the specified group.

The installation can define as many groups as necessary; one group and its name are standard in all systems (SYSGROUP). The installation then assigns each mass storage volume to a user group, to SYSGROUP, or to no group.

## Nonspecific Volume Requests for Mass Storage Volumes

Previous descriptions of nonspecific DASD volume requests apply to mass storage volumes. The type of request can be modified by the MSVGP parameter that specifies an installation defined subset of all mass storage volumes to be used by the system to satisfy the request. MSVGP implies a private volume. The system will select a volume from the defined group that has sufficient space to satisfy the space requirements of the DD statement. (See the section on mass storage volume control in *Mass Storage System (MSS) Services: General Information* or the selection of MSVGP volumes to satisfy space requirements.) If you code the MSVGP parameter, the VOLUME parameter can be used to specify a volume count, but must not be used for volume serial numbers. VOLUME=PRIVATE is redundant when MSVGP is used.

You can specify that data sets be allocated to different volumes by coding the ddname operand on the MSVGP parameter. It may be desirable to specify different volumes for two data sets, for example, when an existing data set containing a critical master file is used for input and a new data set is created for the output master file.

If MSVGP is not specified when you make a nonspecific request for a:

- Private mass storage volume, the system always causes a default group of volumes to be used (MSVGP=SYSGROUP).
- Nonprivate mass storage volume that is to contain a temporary data set, the system assigns a public or storage mass storage volume that is already mounted (if one is available). Otherwise, the request is treated as a nonspecific volume request for a private volume.
- Nonprivate mass storage volume that is to contain a nontemporary data set, the system assigns a mass storage volume if one is mounted. Otherwise, the request is treated as a nonspecific volume request for a private volume.

## Specific Volume Requests for Mass Storage Volumes

Previously defined descriptions of specific DASD volume requests (direct access storage volumes) also apply to mass storage volumes.

Because there is no operator involvement or decision making in mounting mass storage volumes, we recommend (for data integrity purposes) that you catalog all permanent data sets on mass storage volumes. All specific requests for these data sets should always reference the volumes using by the catalog, not the VOLUME parameter. Reference to the catalog is required when extending an existing multivolume data set to one or more volumes. The reason is that the system must know all volumes on which the data set currently resides before it selects the new volume. Parallel mounting must also be specified, to ensure proper multivolume extensions.



## Requesting Space for Non-VSAM Data Sets on Mass Storage Volumes

When an installation defines mass storage volume groups, each group is given a default for space. Specific volume requests for new data sets require the SPACE parameter. Nonspecific volume requests with the MSVGP parameter can optionally specify the SPACE parameter. Nonspecific volume requests without the MSVGP parameter can optionally specify the SPACE parameter if the request will default to MSVGP=SYSGROUP. If other types of space attributes are desired, the SPACE parameter can be coded to override the specified default. Neither directory nor index quantities can be provided in the default; therefore, you must code the SPACE parameter for new BPAM or ISAM data sets on mass storage volumes.

## Retrieving Generation Data Groups Residing on DASD Volumes

For generation data groups residing on DASD (including MSS) volumes, when you specify the generation group name without a generation number (GDG ALL request), and request parallel mounting in the UNIT parameter, the system mounts all volumes of **all** generations.

Before using mass storage volumes, refer to *Mass Storage System (MSS) Services: General Information* and *Mass Storage System (MSS) Services: Reference Information*.

## Requesting Space for Non-VSAM Data Sets

You must request space for every non-VSAM data set created on a direct access volume. To request space, code the SPACE parameter on the DD statement that defines the data set. The SPACE parameter provides two ways to request space:

- Tell the system how much space you want and let the system assign specific tracks.
- Tell the system the specific tracks on which you want the data set written.

Letting the system assign specific tracks is the easiest and most frequently used method of requesting space. You need only specify the unit of measurement to be used to compute the space requirement and how many of the units of measurement the data set requires. In addition, this form of the SPACE parameter offers several options:

- A secondary quantity, to be used if the data set runs out of space
- Space for a directory or index
- Release of unused space
- Contiguous space
- Whole cylinders

OS/MVT and OS/VS2 Release 1 (SVS) included the SPLIT and SUBALLOC parameters to request space for a group of data sets on a single direct access volume. These two parameters are now internally converted to SPACE requests. SUBALLOC requests are not eligible for virtual input/output (VIO).

## The Basic Space Request: Unit of Measurement and Primary Quantity

To have the system assign specific tracks, specify only the **unit of measurement** the system should use to allocate space and the **primary quantity** of space needed. As the unit of measurement, you can specify:

- Average block length of the data, for blocks
- TRK, for tracks
- CYL, for cylinders

As the primary quantity, code an integer that indicates how many blocks, tracks, or cylinders are required.

It is easiest to specify an average block length: the system will allocate the least number of tracks required to contain the number of blocks specified. Specifying block length also maintains device independence; you can change the device type in the UNIT parameter without altering the space request or you can code in the UNIT parameter a group name that includes different direct access devices.

When specifying TRK or CYL, compute the number of tracks or cylinders required; consider such variables as the device type, track capacity, tracks per cylinder, cylinders per volume, data length (blocksize), key length, and device overhead. These variables, and examples of estimating space requirements for partitioned and indexed sequential data sets, are described in *Data Management Services Guide*.

Cylinder allocation allows faster input/output of sequential data sets than does track allocation. When you request space in terms of average block length, the system will allocate tracks to contain the request unless you code ROUND as the last subparameter in the SPACE parameter. The system will then allocate the smallest number of cylinders needed to contain the request.

### How the System Satisfies Your Primary and Secondary Request

Enough available space must exist on one volume to satisfy the primary request. If there is not enough space available on a single volume, the system will terminate the job or search another volume, depending on the type of volume request made:

**Specific volume request** (for example, you code volume serial numbers): If sufficient space is not available on the first volume specified, the job is terminated. When extending a multivolume data set, if sufficient space is not available to satisfy secondary allocation on the next specified volume, the job is terminated.

**Nonspecific volume request** (for example, you do not code volume serial numbers): If space is not available on the first volume chosen, the system will choose another volume and continue the search, causing volumes to be mounted if necessary. The system continues to search until a volume with sufficient space is found or the operator cancels the job.

*Note:* If the first volume selected by allocation to satisfy a request for a new ISAM data set does not contain sufficient storage to satisfy the request, allocation does not attempt to find another volume with sufficient space if the request is of the following types.

- A request for multiple volumes or units.
- A request uses the second, third, or subsequent DD statement you used to define the dataset.

The system attempts to allocate the primary and secondary quantity in contiguous tracks or cylinders. If contiguous space is not available, the system satisfies the request with up to five noncontiguous **extents** (blocks) of space. (If user labels are specified — that is, you code **SUL** in the **LABEL** parameter — the system allocates up to four noncontiguous extents of space. The system allocates a track for user labels separate from the primary quantity; this one track is considered an extent, and therefore, up to four additional extents can be allocated to satisfy the primary quantity.)

### A Secondary Request for Space

In the primary quantity, you need not anticipate all future demands for space for a data set. Code a **secondary request** for space to be used only if the data set exceeds its allocated space. Do this by coding an integer, following the primary quantity, that indicates how much additional space should be allocated.

For data sets whose disposition is **NEW** or **MOD**, the system allocates this space on the same volume as the primary quantity until:

1. There is not enough space available on the volume to allocate the secondary quantity, or
2. A total of 16 extents, less the number of extents for the primary quantity and user label space, have been allocated to the data set. (BDAM data sets cannot be extended.)

If either of these conditions is satisfied, the system must allocate the secondary quantity on another volume. However, the system will allocate your secondary request on another volume only if you request more than one volume in the **VOLUME** parameter or for a specific volume request, you request more volumes than devices.

If you are making a nonspecific volume request and there exists the possibility that the system will allocate a permanently mounted volume, code **PRIVATE** in the **VOLUME** parameter.

When allocating a secondary quantity for a data set whose disposition is **OLD** (in other words, a data set that is preallocated or is being written over), the system will go to the next volume, if one is specified, and see if there is already a secondary quantity allocated there.

If you did specify another volume and there is already a secondary quantity, the system will

- Use that space instead of making another allocation, or
- Allocate space if no space is already allocated there for the data set.

If you didn't specify another volume, the secondary space will be allocated on the current volume.

A secondary quantity can be requested when creating a data set or when retrieving an existing data set, whether or not you coded a secondary quantity in the original request. A secondary request for an existing data set is in effect only for the duration of the job step and overrides an original request if one was made.

If you specify **SPACE** in terms of average block length, code the maximum block length of the data in either the **DCB** macro instruction or the **BLKSIZE** subparameter of the **DCB** parameter on the **DD** statement: the system uses the maximum block length to compute how many additional tracks to allocate.

## Requesting Directory Space for a Partitioned Data Set

To create a partitioned data set, request a primary quantity large enough to include space for a directory. A directory is an index used by the system to locate members in a partitioned data set. It consists of 256-byte records, and you must specify, as the third quantity in the SPACE parameter, how many records the directory is to contain. The directory is included at the beginning of the primary space, which must be large enough to contain the directory. Request enough directory space to allow for growth of the data set: you cannot lengthen the directory as you can lengthen the data set itself, that is, by requesting a secondary quantity. If the directory runs out of space, recreate the data set.

For a complete description of the directory, including details on member entries that will enable you to compute how many records to request, see *Data Management Services Guide*.

## Requesting Index Space for an Indexed Sequential Data Set

If you are creating an indexed sequential data set that occupies more than one cylinder, and you are not defining the index on a separate DD statement, you can request index space in addition to a primary quantity. Request index space as the third quantity in the SPACE parameter. The space request for an indexed sequential data set must be in terms of cylinders or absolute track allocation. The system determines whether the request is for a directory or an index by examining the DSORG subparameter of the DCB parameter on the DD statement. DCB=DSORG=IS or DCB=DSORG=ISU must be included on any DD statement defining an indexed sequential data set.

The index quantity is added to the primary quantity when considering the space requirements.

## Assigning Specific Tracks

You can request that specified tracks on a volume be allocated to a data set. This is the most stringent request for space: if any of the tracks requested are occupied, the space cannot be allocated and the job is terminated. An example of where specific track allocation is required is a data set that is to reside under the fixed heads of a 3348 Model 70F Data Module (cylinders 1-5).

To request specific tracks, you must code:

- ABSTR as the first subparameter, indicating absolute tracks
- A primary quantity, specifying the number of tracks to be allocated
- The relative track number of the first track to be allocated

When using the ABSTR subparameter, count the first track of the first cylinder on the volume as 0. Count through the tracks on each cylinder until you reach the track on which you want your data set to start. Do not request track 0.

For example, to allocate one track for a data set and specifically the second track on a volume, code:

```
//DDEX DD SPACE=(ABSTR,(1,1))
```

For a partitioned data set, specify how many records you want allocated for a directory. If requesting a user-label track, this track will be the first of the space requested.

If you are defining an indexed sequential data set using absolute track allocation, the number of tracks for the index, primary, or overflow areas must be equal to an integral number of cylinders and on a cylinder boundary. All of the DD statements defining the indexed sequential data sets must request specific tracks.

## Example of Requesting Space

One purpose of this job is to request space for two temporary data sets. The following steps refer to these data sets for volume information.

```
//ALLOC      JOB   (3416,354),STONER,MSGLEVEL=1,MSGCLASS=C
//STEP1     EXEC  PGM=TESTSYSO
//DD11      DD   UNIT=3350,DISP=(,PASS),SPACE=(TRK,(10,5))
//DD12      DD   UNIT=3330,DISP=(,PASS),SPACE=(TRK,(10,5))
//SYSABEND  DD   SYSOUT=L
//STEP2     EXEC  PGM=TESTSYSO
//DD1       DD   DSN=*.STEP1.DD11,DISP=(OLD,DELETE,DELETE)
//DD2       DD   VOL=REF=*.STEP1.DD12,SPACE=(TRK,(3,1)),UNIT=3330
//SYSABEND  DD   SYSOUT=L
```

1. The JOB statement specifies that all job related output is to be printed and that system messages for the job are to be written to output class C.
2. STEP1 defines two temporary data sets. Step 2 refers to these data sets for volume information.
3. The space requirements for these requests indicate that for DD11 and DD12 in STEP1 you want 10 primary and 5 secondary tracks; and for DD2 in STEP2 you want 3 primary and 1 secondary track.

## Specifying Data Set Processing Options

By coding JCL statements, you can request output data sets, listings of JCL statements, system messages, and abnormal termination dumps. By coding the OUTPUT JCL statement, you can request special forms processing, routing of output, grouping of output data sets, and multiple printing of data sets.

The following topics discuss the functions the system provides to process your job's output. The specific JCL statement, JES2 control statement, or JES3 control statement that performs the function is discussed under the function it provides.

This section includes the following topics:

- Processing Output Data Sets for the JOB
- Processing System Output Data Sets Using the OUTPUT JCL Statement
- Assigning System Output Data Sets to Output Classes
- Specifying the Device
- JES Output Class Processing
- Delaying the Writing of an Output Data Set
- Suppressing the Writing of an Output Data Set

- Limiting Output Records
- Specifying Forms Overflow Processing and Printer Spacing
- Interpretation of Punched Output
- Requesting Multiple Copies of an Output Data Set
- Requesting Copy Modification
- Requesting Printer Form and Character Control
- Requesting Forms Overlay

## Processing Output Data Sets for the JOB

The two ways to process output data sets are:

- Assign processing options to the data set and allow the job entry subsystem to manage the output devices.
- Specify the device on which the output should be written.

When you assign processing options to a data set, it is handled by the job entry subsystem in use at your installation. The data set is first written to the job entry subsystem spool device and then written or transmitted to the final output device by either the job entry subsystem or an external writer.

For either JES2 or JES3, when you specify the output device on the UNIT parameter, the device, if available, is exclusively assigned to your job and under the control of your program.

Output data sets to be written to a 3540 diskette must be assigned to an output class that is processed by the diskette writer (an external writer), as described in *OS/VS2 IBM 3540 Programmer's Reference*. For the diskette writer to receive data sets, the job entry subsystem initialization deck must specify the SYSOUT classes to be reserved for diskette output. To write data sets on a diskette, the operator must start the diskette writer to a 3540 device.

## Processing System Output Data Sets Using the OUTPUT JCL Statement

The OUTPUT JCL statement governs the processing of system output data sets. It allows you to specify:

- The processing options for a system output data set.
- Default processing options for system output data sets.

The general format of the OUTPUT JCL statement is:

```
//name OUTPUT parameter[,parameter]...
```

The OUTPUT JCL statement can be referenced explicitly or implicitly by a sysout DD statement, as described below.

A sysout DD statement can reference more than one OUTPUT JCL statement. For each reference to an OUTPUT JCL statement, the system processes the data set defined by the DD statement according to the output processing options that apply from the DD and the OUTPUT JCL statements.

## Explicit Reference

You code an explicit reference to an OUTPUT JCL statement by specifying the name of the OUTPUT JCL statement on the OUTPUT parameter of the sysout DD statement. You can have up to 128 explicit references to OUTPUT JCL statements from a single DD statement. Each reference causes a separate processing of the data set. For example:

```
//STEP1 EXEC PGM=MFK
//OUT1 OUTPUT COPIES=6,DEST=NY,FORMS=BILLS
//OUT2 OUTPUT COPIES=2,DEST=KY,FORMS=LOG
//REF1 DD SYSOUT=A,OUTPUT=(*.OUT1,*.OUT2)
```

In the example, two sets of output are created from DD statement REF1. One of the sets will go to NY and have six copies printed on the form defined as BILLS. The other set will go to KY and have two copies printed on the form defined as LOG.

## Implicit Reference

To code an implicit (default) reference to an OUTPUT JCL statement, code `DEFAULT=YES` on the OUTPUT JCL statement and do not code an OUTPUT parameter on the sysout DD statement.

*Note:* You can implicitly reference any number of output JCL statements.

You can place default OUTPUT JCL statements at the job level or the step level. Where you place the default OUTPUT JCL statement determines the scope of control that the OUTPUT JCL statement has on the sysout data sets in the job.

A job-level OUTPUT JCL statement is one that appears **before** the first EXEC statement in the job. Any sysout DD statement within the job can implicitly reference a job-level OUTPUT JCL statement, but only if the step does not contain a step-level default OUTPUT JCL statement and the DD statement does not explicitly reference an OUTPUT JCL statement.

A step-level OUTPUT JCL statement is one that appears anywhere **after** the first EXEC statement in the job. Only sysout DD statements within the step can implicitly reference a step-level OUTPUT JCL statement. You may code more than one job- or step-level **default** OUTPUT JCL statement per job or step.

When you explicitly reference an OUTPUT JCL statement, the system ignores all step- and job-level default OUTPUT JCL statements. For example,

```
//JOB1 JOB options.....
//OUT1 OUTPUT COPIES=8,DEST=FRANCE
//OUT2 OUTPUT COPIES=2,FORMS=A,DEFAULT=YES
//STEP1 EXEC PGM=DEMENT
//OUT3 OUTPUT DEFAULT=YES,COPIES=5,DEST=REMULAC
//INPUT DD DSN=RHINO
//MFK1 DD SYSOUT=A
//MFK2 DD SYSOUT=B,OUTPUT=*.OUT1
```

In the preceding example,

- The system processes the output from DD statement MFK1 using the options on the OUTPUT statement OUT3 (1) because MFK1 does not contain an OUTPUT parameter and (2) because OUT3 contains `DEFAULT=YES` and is in the same step as MFK1. MFK1 cannot implicitly reference the job-level default statement OUT2 because of

step-level default statement OUT3. If STEP1 had not contained OUT3, MFK1 would have referenced statement OUT2.

- The system processes the output from DD statement MFK2 according to the processing options on the job-level OUTPUT JCL statement OUT1 because DD statement MFK2 explicitly references OUT1 using the OUTPUT parameter. Note that the system ignores the processing options on all default OUTPUT JCL statements (OUT2 and OUT3).

### JES2 /\*OUTPUT References

If you explicitly or implicitly reference an OUTPUT JCL statement from a DD statement, and you also reference a JES2 /\*OUTPUT control statement(s) from the same DD statement, the system ignores the JES2 /\*OUTPUT control statement(s). If you do not reference an OUTPUT JCL statement, but you do reference a JES2 /\*OUTPUT control statement, the system uses the output processing options coded on the JES2 /\*OUTPUT control statement.

Be careful when you modify a job that includes DD statements that previously referenced JES2 /\*OUTPUT control statements. The DD statements referenced the JES2 /\*OUTPUT control statements using the code field in the SYSOUT parameter on the DD statement. If this DD statement now references an OUTPUT JCL statement, the references to the JES2 /\*OUTPUT statement(s) are ignored and the code field is no longer recognized as a reference to a JES2 /\*OUTPUT statement. The system now interprets the code field as a forms name to be used when processing the data set.

### JES3 /\*\*FORMAT Statement with OUTPUT JCL Statement

When a JES3 /\*\*FORMAT statement explicitly references a sysout DD statement that, in turn, explicitly references an OUTPUT JCL statement, the processing options from both the OUTPUT JCL and JES3 /\*\*FORMAT statements apply. For example:

```
//PUT1    OUTPUT options...
/**FORMAT PR,DDNAME=DD9,options...
//DD9    DD  SYSOUT=A,OUTPUT=* .PUT1
```

**Two separate sets of system output** are created from the data set defined by DD statement DD9. One set of output is created according to the processing options on OUTPUT JCL statement PUT1. The other is created according to the processing options on the JES3 /\*\*FORMAT statement.

### Using the OUTPUT JCL Statement to Tailor the Job Stream

When a sysout DD statement references an OUTPUT JCL statement, either explicitly or implicitly, the system selects the processing options to be used as follows:

- If the same options are on the OUTPUT JCL statement and the sysout DD statement, the values on the **sysout DD statement override** the values coded on the OUTPUT JCL statement.
- If an option is on only one of the statements, the system uses it.

The system combines the options from the two statements to determine how the sysout data set is processed.



**Processing for Explicit References:** When you explicitly reference an OUTPUT JCL statement using the OUTPUT parameter on the sysout DD statement, consider the following:

- The OUTPUT JCL statement must appear earlier in the input stream, before any sysout DD statement that references it.
- When you code the OUTPUT parameter on a DD statement, the system ignores OUTPUT JCL statements containing DEFAULT = YES. However, the the OUTPUT parameter can refer to an OUTPUT JCL statement that contains DEFAULT = YES, and the system will accept this reference.

**Processing for Implicit (Default) References:** You can use the OUTPUT JCL statement to provide output processing options for all or part of the sysout data sets in your job. Then, if you wish slightly different processing options for a specific data set in the job, specify the different options on the sysout DD statement. The DD options override those on the OUTPUT JCL statement for that specific set of output, without affecting other output from the job.

**An Example Using Explicit and Implicit References:** This example illustrates the use of the OUTPUT JCL statement and shows how the statement's position affects the processing of the output data sets.

```
//EXAMP      JOB      MSGCLASS=A
//OUT1       OUTPUT   DEFAULT=YES , DEST=COMPLEX7 , FORMS=BILLING ,
//           CHARS=( AOA , AOB ) , COPIES=2
//OUT2       OUTPUT   DEFAULT=YES , DEST=COMPLEX1
//STEP1      EXEC     PGM=ORDERS
//R1         DD       SYSOUT=A
//R2         DD       SYSOUT=A
//STEP2      EXEC     PGM=BILLING
//OUT3       OUTPUT   DEFAULT=YES , DEST=COMPLEX3
//B1         DD       SYSOUT=A
//B2         DD       SYSOUT=A , OUTPUT=( * .OUT3 , * .OUT2 )
//STEP3      EXEC     PGM=REPORTS
//OUT4       OUTPUT   FORMS=SHORT , DEST=COMPLEX1
//RP1        DD       SYSOUT=A
//RP2        DD       SYSOUT=A , OUTPUT=( * .STEP2 .OUT3 , * .OUT1 )
//
```

In STEP1, the system processes DD statements R1 and R2 using the processing options specified on job-level OUTPUT JCL statements OUT1 and OUT2 because

- DEFAULT = YES is specified on OUTPUT JCL statements OUT1 and OUT2, and
- there is no OUTPUT JCL statement with DEFAULT = YES within STEP1.
- The OUTPUT parameter is not specified on DD statements R1 and R2.

In STEP2, the system processes DD statement B1 using the processing options specified on OUTPUT JCL statement OUT3 because:

- DEFAULT = YES is specified on OUTPUT JCL statement OUT3 and OUTPUT JCL statement OUT3 is within the job step STEP2.
- The OUTPUT parameter is not specified on DD statement B1.
- OUTPUT JCL statement OUT3 is within STEP2; therefore, the system ignores the DEFAULT = YES specification on job-level OUTPUT JCL statements OUT1 and OUT2 when processing DD statement B1.

In STEP2, the system processes DD statement B2 using the processing options specified on OUTPUT JCL statements OUT3 and OUT2 because:

- Both of the OUTPUT JCL statements are explicitly referenced from the SYSOUT statement. Explicitly-referenced OUTPUT JCL statements can be in any previous procedure or step, before the DD statement in the current step, or at the job-level.
- Note that default OUTPUT JCL statement OUT1 is ignored when processing the data set defined by DD statement B2 because B2 explicitly references OUTPUT JCL statements OUT3 and OUT2.

In STEP3, the system processes DD statement RP1 using the output processing options specified on the job-level OUTPUT JCL statements OUT1 and OUT2 because:

- DEFAULT= YES is specified on OUTPUT JCL statements OUT1 and OUT2, and
- no OUTPUT JCL statement with DEFAULT= YES is coded within STEP3.
- The OUTPUT parameter is not specified on DD statement RP1.

*Note:* In STEP3, OUTPUT JCL statement OUT4 is not used at all because it does not have DEFAULT= YES coded, and no DD statement explicitly references OUT4.

In STEP3, DD statement RP2 is processed using OUTPUT statements OUT3 and OUT1. You can explicitly reference an OUTPUT JCL statement in another step if you use a fully qualified reference, such as the reference to OUTPUT statement OUT3 used on DD statement RP2.

You may explicitly reference an OUTPUT JCL statement with DEFAULT= YES coded, such as the reference to OUT1 from DD statement RP2. The system ignores the DEFAULT parameter and uses the remaining processing options according to the normal rules that apply when coding explicit references.

### Specifying Sublist Using the OUTPUT JCL Statement

You must be careful when you are coding the COPIES, MODIFY or FLASH parameter on an OUTPUT JCL statement. If you code the COPIES, FLASH or MODIFY parameter on a sysout DD statement, you get a JCL error if you code it with a null first subparameter. For example, you **cannot** code MODIFY=(,3) on a DD statement, even though you **can** code MODIFY=(,3) on an OUTPUT JCL statement. For example:

```
//EXAMP2 OUTPUT FLASH=( , 3 ),DEFAULT=YES  
//FVZ2 DD FLASH=( INV)
```

In the example, INV is used because the FLASH parameter on DD statement FVZ2 overrides the entire FLASH parameter on OUTPUT JCL statement EXAMP2. This means that the FLASH count will not be set to 3; instead all copies of the entire data set will be flashed because the FLASH COUNT subparameter was not specified on the DD statement FVZ2. In the case of overrides, even though a part may be left off an overriding parameter, the system replaces the whole overridden parameter with the whole overriding parameter. The system uses the FLASH parameter as coded on DD statement FVZ2 and ignores whatever is coded on the FLASH parameter of OUTPUT JCL statement EXAMP2.

## Specifying a Destination for the Data Set

You can specify a destination for an output data set. You may want a set of reports printed in Chicago, New York, Paris, and Los Angeles. To do this, you must code and reference four different OUTPUT JCL statements with a destination specified on each because you can code only one destination on each OUTPUT JCL statement. However, by referencing OUTPUT JCL statements, you can specify up to 128 different destinations from a single DD statement. In addition, you can use the OUTPUT JCL statement to specify the output processing options you may want to use at any or all of these destinations.

## Grouping Data Sets Using the OUTPUT JCL Statement

In JES2 systems, you can group system output data sets together using the GROUPID parameter on the OUTPUT JCL statement. Output data sets with the same group identifier are processed together by the system. You can use grouping to keep related information from different data sets closer together in both the location and time when they are printed.

You may always group system output data sets with similar processing characteristics. But, you cannot group output data sets with differing SYSOUT classes, destinations, processing modes, writer names, or groupids. If you attempt to do so, your output group specified via GROUPID is further broken down into smaller output groups in which all data sets have identical class, destinations, processing modes (PRMODE), writer name, and groupid.

Your installation controls whether or not you can group output data sets with different printer setup requirements, such as forms. Such output groups are called demand setup groups. If you are attempting to create a demand setup group and your installation does not permit demand setup groups, the group is further broken down into smaller groups in which all data sets have identical setup characteristics. Consult with your installation to determine if demand setup grouping is allowed.

*An Example Using the OUTPUT JCL Statement to Group Output:* The following example illustrates grouping of data sets using the OUTPUT JCL statement.

```
//TEST1 JOB      MSGCLASS=B
//OUT1  OUTPUT   GROUPID=GRP10,UCS=PN,DEST=RT6,DEFAULT=YES
//STEP1 EXEC     PGM=REPORT
//RP1   DD       SYSOUT=A
//RP2   DD       SYSOUT=B
//RP3   DD       SYSOUT=A
```

In this example, two groups are created for the three different system output data sets.

## Managing the System-Managed Data Sets: The JESDS Parameter

Sysout DD statements are not coded for the system-generated and -maintained output data sets, which are the job log, the JCL statements and messages, and the system messages. The JESDS parameter of the OUTPUT JCL statement indicates that the processing characteristics for these data sets are coded on the OUTPUT JCL statement. You can code the JESDS parameter **only** on job-level OUTPUT JCL statements. Use:

```
JESDS=LOG to control the job log data set;
JESDS=JCL to control the JCL images data set;
JESDS=MSG to control the system messages data set;
JESDS=ALL to control all of the system-managed data sets.
```

The following example requests that the three system-managed data sets be printed normally, but also requests that a copy of each be routed to an external writer named JCLOGGER.

```
// JOB
// OUTPUT JESDS=ALL
// OUTPUT JESDS=ALL,WRITER=JCLOGGER
// EXEC PGM=REPORT
```

In the next example, four different output groups are created. Group SYSPROG will contain a copy of all three of the system-managed data sets. Group OPER will also contain a copy of all three system managed data sets. Group USER will contain a copy of the system-managed data sets as well as a copy of the data set defined by DD statement SYSPRINT (this group is processed locally). The system creates a fourth group without a user-specified group name. The system generates a group name and that group contains a copy of the three system-managed data sets and a copy of the data set defined by DD statement SYSPRINT (this group is processed remotely at destination REMOTE):

```
// JOB MSGCLASS=A
//SYSPROG OUTPUT JESDS=ALL,GROUPID=SYSPROG
//OPER OUTPUT JESDS=ALL,GROUPID=OPER
//USER OUTPUT JESDS=ALL,DEFAULT=YES,GROUPID=USER
//REMOTE OUTPUT JESDS=ALL,DEFAULT=YES,DEST=REMOTE
// EXEC PGM=REPORT
//SYSPRINT DD SYSOUT=A
```

Be careful when combining sysout data sets and system-managed output data sets within an output group. The values you specify on the sysout DD statements might override those specified on the OUTPUT JCL statement for the DD-defined data sets; but the values you specify on the OUTPUT JCL statement **always** apply to the system-managed data sets. Therefore, the output characteristics given to the system-managed output data sets and sysout data sets can vary greatly, even if the data sets all reference the same OUTPUT JCL statement. An example of how problems can arise when you try to group the system-managed output data sets and sysout data sets is shown below.

```
//SYSDS JOB MSGCLASS=A
//OUT1 OUTPUT JESDS=ALL,CLASS=F,GROUPID=JOINT,DEFAULT=YES
//STEP1 EXEC PGM=REPORT
//REQPRT DD SYSOUT=A
```

In this example, two groups are produced:

1. The system messages are put in a subgroup of the group called JOINT and are printed in system output class F, specified in the OUTPUT JCL statement.
2. The REQPRT output data set is put in a different subgroup of the group called JOINT and is printed in system output class A, specified in the DD statement.

Even though the sysout DD statement REQPRT references the OUTPUT JCL statement with the JESDS parameter, the output processing characteristics of the data sets are different. The data set defined by sysout DD statement REQPRT takes its processing characteristics from a combination of the values coded on the sysout DD statement and the OUTPUT JCL statement. The system-managed data sets take their output processing characteristics only from the OUTPUT JCL statement.

## Specifying a Priority for Sysout Data Sets.

You can specify a priority for an output data set using the PRIORITY parameter on the OUTPUT JCL statement. You can use this function to increase an output data set's priority so it will be printed much sooner than it otherwise might have been.

However, your installation may instruct the system to ignore the priority specified on an OUTPUT JCL statement. Consult your systems programmer to determine whether you can specify an output data set's processing priority using the OUTPUT JCL statement.

## Assigning System Output Data Sets to Output Classes

Output classes generally include system output data sets that have similar characteristics and that are written to the same device. There are 36 possible output classes; each is defined by an alphabetic (A-Z) or numeric (0-9) character. The output class is indicated on the DD statement SYSOUT parameter, the CLASS parameter of the OUTPUT JCL statement, or the MSGCLASS parameter on the JOB statement.

The letter and number names have no inherent meaning; each installation defines its own output classes and can assign special processing characteristics for each class. For example, output class W might contain low-priority output; class X might contain output to be printed on a special form (eliminating the need to request the form directly); class J might be reserved for high-volume output.

If you want the output data set and messages from the job to be printed on the same output listing, specify one of the following:

- SYSOUT = \* on the DD statement.
- CLASS = \* on the OUTPUT JCL statement.
- The same output class in the DD SYSOUT parameter or OUTPUT JCL CLASS parameter as specified in the JOB MSGCLASS parameter.

## Held System Output Classes

The installation can designate certain system output classes as held, that is, not able to be selected by an output device. If the output class specified for the MSGCLASS parameter is not designated as a held class, the system generated data sets will not be held and none of the job's data sets assigned to held classes will be held. Data sets can be explicitly held by coding the HOLD = YES parameter or by coding TSO commands. (Refer to *TSO Command Language Reference* for information on the TSO commands.) Jobs can be released from the hold state by the operator or by the time-sharing user with the TSO OUTPUT command. Control of holding or not holding of all desired print data sets is done using held classes on the MSGCLASS parameter on the JOB statement. If MSGCLASS is set to a held class and SYSOUT class is not set to a held class, the system data sets will be held but the job's data sets will not be held.

For more information on holding data sets, see "Delaying Initiation of Your Job in JES2" on page 3-23.

If your installation allows demand setup, output for all data sets could be printed on the same listing, even if parameters such as FORMS, FCB, and UCS are different.

## Specifying the Device

To process an output data set without using the job entry subsystem output service, code the UNIT parameter on the DD statement defining the device on which the data set is to be written. The system will allocate the device exclusively to the job if the device is available: no other job can write output to that device until it is released. Jobs cannot share an output device as they can when output is assigned to output classes.

Data management routines write the output from the program to the device specified in the UNIT parameter. Specifying a particular output device in the UNIT parameter generally is not the most efficient method for obtaining system output.

## Specifying the Internal Reader

You might wish to make the output of a job or job step the input to another job or step. Instead of directing that output to a card punch or a tape drive and then resubmitting the output as input to the later job, you can direct the output to an internal reader.

The input to the internal reader must consist of JCL statements to run the later job.

To direct the output of a job or job step to the internal reader, code:

```
//IROUT DD SYSOUT=(A,INTRDR)
```

- “INTRDR” is an IBM-reserved name identifying the internal reader as the program to process this output data set.
- The SYSOUT class of this DD statement becomes the default message class for the job going into the internal reader unless you code the MSGCLASS parameter on the JOB statement. See “Job Log” on page 3-14.

The system places the output records for the internal reader into a buffer in your address space. When this buffer is full, the contents are copied into the JES address space; JES3 then spools the data. The JES input service can now process the contents of this buffer as input to the job you specify.

Instead of waiting for the buffer in your address space to fill up, you can send the contents of the internal reader buffer directly to the JES input service by coding:

- /\*EOF as the last record of the job.
  - For JES2, this control statement delimits the current job and makes it eligible for immediate processing by JES2 input service.
  - For JES3, this control statement is a request for special end-of-record processing. The internal reader facility closes the data set and sends the data set to the JES3 input service. The internal reader facility closes the data set without deallocating it so it is available for more records.

- **/\*DEL** as the last record of the job.
  - For **JES2**, this control statement cancels the current job and schedules it for immediate output processing. The output will consist of any JCL submitted followed by a message indicating that the job was deleted before execution.
  - For **JES3**, this control statement is treated like **/\*EOF**.
- **/\*PURGE** as the last record in the job.
 

For **JES2 only**, this control statement cancels the current job and schedules it for purge processing; no output is produced for the job.
- **/\*SCAN** as the last record in the job.
 

For **JES2 only**, this control statement requests that JES2 scan the current job for JCL errors only. The job is not to be executed.

### Example of Using the Internal Reader

In the following example, different groups of data are directed to the internal reader. Each group is started with a **JOB** statement.

```
//JOBA      JOB      D58JTH,HIGGIE
//GENER     EXEC     PGM=IEBGENER
//SYSIN     DD       DUMMY
//SYSPRINT  DD       SYSOUT=A,DEST=NODE1
//SYSUT2    DD       SYSOUT=(M,INTRDR)
//SYSUT1    DD       DATA

//JOB      JOB      D58JTH,HIGGIE,MSGLEVEL=(1,1)
//REPORTA  EXEC     PGM=SUMMARY
//OUTDD1   DD       SYSOUT=*
//INPUT    DD       DSN=REPRTSUM,DISP=OLD

//JOB      JOB      D58JTH,HIGGIE,MSGLEVEL=(1,1)
//REPORTB  EXEC     PGM=SUMMARY
//OUTDD2   DD       SYSOUT=A,DEST=NODE2
//INPUT    DD       DSN=REPRTDAT,DISP=OLD

/*EOF
```

- **JOBA** executes program **IEBGENER**.
- Program **IEBGENER** reads **JOB** and **JOBC** from **SYSUT1** and submits them to the internal reader.
- The message class for **JOB** and **JOBC** is **M**, the **SYSOUT** class specified on the internal reader **DD** statement (**SYSUT2**).
- The message class for the output data set from **JOB** (**OUTDD1**) is **M** because **SYSOUT = \*** is coded.
- The **/\*EOF** statement specifies that the preceding jobs are to be sent immediately to the job entry subsystem for input processing.

For more information on the internal reader, see *SPL: Job Management*, *SPL: JES2 Installation, Initialization, and Tuning*, or *JES3 SPL: Initialization and Tuning*.

## **JES Output Class Processing**

Using JES2 or JES3 is an efficient way to write output. The job entry subsystems support the use of local and remote printers and punches as devices on which data sets are written. An external writer supports tape and direct access devices and user-written writer routines.

Output for all data sets will generally be printed on the same listing if such parameters as CLASS, FORMS, FCB, UCS, and DEST have similar characteristics and a user-written writer is not specified. The installation may choose to print all data sets that specify the same output class as the MSGCLASS parameter on the same listing, even though FORMS, UCS, FCB, and DEST are different.

For an external writer, the operator will determine which data sets will be selected. This can cause certain output to print out on the same listing even though all of the FORMS, DEST, UCS, and FCB parameters do not indicate the same characteristics.

When an external writer is specified, either an IBM-supplied external writer or a user-written writer can process the output. The operator must start the external writer to have the data written. If you want to know more about how to write an external writer routine, refer to *SPL: Job Management*.

## **JES2 Output Class Processing**

Job-related output is output that is not held, spun off or processed by a user-written writer. (A spun off data set is made available for output processing before job termination.) Job-related output will be retained until the end of the job and printed by JES2. If you release a held data set in time for it to be printed with other non-held or no longer held output, JES2 will print them together if the following criteria are satisfied:

- The released data set has not been spun-off; spun-off data sets are always printed separately.
- Printing of the job-related data sets has not begun, or else released data sets are printed independent of job related data sets.
- The job-related data set does not have multiple copies; released data sets will not be printed with job-related data sets for which there are multiple copies.
- The released data sets would have been part of the job-related data set had they not been held.

Dynamically deallocated SYSOUT data sets are spun off and are not considered part of the job-related output.



## Delaying the Writing of an Output Data Set

A data set can be made available for inspection from a time-sharing terminal and its printing delayed by specifying held classes and/or by coding the HOLD parameter. For example, the installation can direct the delayed printing of a very large data set to prevent monopolizing an output device until smaller data sets are written. If a data set requires special forms that are not immediately available, it can be held until the operator supplies those forms. When HOLD= YES is specified on the DD statement, the data set is placed on a hold queue until the operator releases it. Notify the operator (using the NOTIFY parameter for TSO, the MESSAGE statement for JES2, or the OPERATOR statement for JES3) when that data set is ready for processing because no message will be sent to the operator. The operator or time-sharing user can release the data set for output processing.

## Suppressing the Writing of an Output Data Set

Whether writing an output data set by coding the SYSOUT parameter or the UNIT parameter, you can suppress the writing of the data set by defining it as a dummy data set. This is useful when you are testing a program and you do not want data sets printed until you are sure they will contain meaningful output. Suppressing the writing of a data set saves processing time.

If you are creating an output data set by coding the SYSOUT parameter, code the DUMMY parameter to define the data set as a dummy data set. When DUMMY is coded, the system ignores the SYSOUT parameter and does not print the data set.

You can also suppress the writing of an output data set by specifying a particular installation-defined sysout class defined to delete all output data sets before they are printed. Use this technique to suppress the output of started tasks such as START and MOUNT commands.

**JES3 users** can suppress the writing of an output data set by coding COPIES=0 on the `//*FORMAT PR` or `//*FORMAT PU` (print or punch) control statements.

If the device on which the data set will be written is specified in the UNIT parameter, you can assign the data set a dummy status by coding DUMMY or by assigning the data set name NULLFILE. All parameters other than DUMMY or DSNAME=NULLFILE and DCB are ignored; no units are assigned to the data set. When the program requests that the data set be written, the request is recognized but no data is transmitted. Requests to write a dummy data set are supported by the basic sequential access method (BSAM), virtual storage access method (VSAM), and queued sequential access method (QSAM). If any other access method is used, the job is terminated.

## Limiting Output Records

To limit the number of logical records in the **output data set**, specify a maximum number of records on the OUTLIM parameter on a DD statement. For example, a program is printing and goes into an endless loop. You can anticipate this problem and only have a maximum number of records printed before having the system abnormally terminate the job.

## JES2 Output Limiting

In a JES2 system, you can specify the estimated number of logical records, bytes, or pages for the **job's** output using the **LINES**, **BYTES** or **PAGES** parameters on a **/\*JOBPARM** control statement (for punch data sets, use **CARDS** instead of **LINES**). For further information about JES2 output limiting, see "**/\*JOBPARM Statement**" on page 16-4.

## JES3 Output Limiting

To limit the printed or punched output of a **job**, specify the estimated number of bytes, lines, pages or cards of output associated with your job by coding the **LINES**, **BYTES**, **PAGES** and/or **CARDS** parameters on the **JES3 /\*MAIN** statement. JES3 uses this information to monitor output and take whatever action is specified if you exceed the estimates. These actions request that the operator receive a warning (the **WARNING** subparameter), that the job be canceled (the **CANCEL** subparameter), or that the job be canceled with a storage dump (the **DUMP** subparameter). JES3 initialization parameter values are used if you omit the estimates.

The **LINES** parameter will not limit the size of an internal reader data set because JES3 does not consider an internal reader data set to be part of the printed output of a job. To restrict this type of data set, you must use the **OUTLIM** parameter on the **DD** statement.

## Specifying JES2 Page Overflow Processing

JES2 will automatically limit the number of lines it prints per page, thus preventing printing over the edge of the form. Overflow processing is specified either by the installation during JES2 initialization or by the programmer coding **JCL** or **JES2** control statements.

You can code the "linect" field in the Accounting Information parameter on the **JOB** statement, or the **LINECT** parameter on the **OUTPUT JCL** statement, the **JES2 JOBPARM** statement, or the **JES2 /\*OUTPUT** statement. You can override the installation-specified number of lines per page through the **LINECT** parameter on the **OUTPUT JCL** statement, **JOBPARM** statement, **JES2 /\*OUTPUT** statement, or on the "linect" field in the accounting information parameter on the **JOB** statement. You can turn off line limiting by coding **LINECT=0**.

## Specifying JES3 Forms Overflow Processing and Printer Spacing

Use the overflow parameter (**OVFL**) on the **JES3 /\*FORMAT PR** control statement to prevent printing across page folds. Specifying **OVFL=ON** on the **/\*FORMAT PR** statement or the **JES3 SYSOUT** initialization statement causes the printer to eject a page when it senses the end-of-forms indicator (channel 12) on the printer's carriage control or in the printer's **FCB**.

You can also control page ejection by specifying the **CONTROL=PROGRAM** parameter on the **/\*FORMAT PR** statement. This causes the format specified in the **DCB=(...,RECFM=format,...)** parameter on the **DD** statement to be used for printer carriage control. Do not use this method if the printer control (tape or **RCB**) contains channel 12 indicators and **OVFL=ON** (the default).

JES3 defaults to **OVFL=ON** and **CONTROL=PROGRAM** on the **/\*FORMAT PR** statement. Therefore, you must specify **OVFL=OFF** on the **/\*FORMAT PR** statements for data sets that are program-controlled. You can turn overflow off by specifying **OVFL=OFF** on each data set's **/\*FORMAT PR** statement.

If the number of data sets requiring `OVFL=ON` is small, the installation can turn off the overflow by specifying `OVFL=OFF` on the `SYSOUT` initialization statement. For those data sets requiring overflow, code a `/*FORMAT PR` statement with `OVFL=ON` to override the `SYSOUT` initialization statement.

## Interpretation of Punched Output

Cards punched on a 3525 card punch from output spooled by either job entry subsystem will be interpreted if:

- You code `FUNC=I` as a DCB subparameter on the `SYSOUT` card, and
- The spooled output is processed by the JES writer rather than an external writer.

If the JES writer processes the spooled output on to a card punch other than the 3525, JES ignores the `FUNC=I` subparameter. Check with your installation to determine if a special output class has been set aside for 3525 output. Card interpretation by the external writer is an operator-specified function. Output to be interpreted should be placed in a class designated by the installation as a punch-with-interpretation class.

### JES3 Punch Output Interpretation on a 3525

Punched output may or may not be interpreted depending on the installation-defined standard for the `SYSOUT` class. You can specify that punched output is to be interpreted by coding the `INT=YES` parameter on the `JES3 /*FORMAT PU` statement. If you omit the device name that specifies a 3525I, JES3 attempts to find one for the output. If you specify a non-interpreting punch device, output is punched on it but not interpreted.

## JES2 Support of the 3211 Indexing Feature

You can request that output printed by JES2 on a 3211 printer be indexed to the right or the left by coding the `INDEX` or `LINDEX` parameters, respectively, on the `OUTPUT JCL` statement or the `JES2 /*OUTPUT` statement. JES2 ignores these parameters if the output is processed by an external writer or is processed to a device other than a 3211. Ask your installation's system programming staff whether an output class has been set aside for output to be processed on a 3211 printer.

## Requesting Multiple Copies of an Output Data Set Using JES2

You can control the number of hard copies of output data sets that JES2 produces. As many as **255** copies of an output data set can be obtained by coding the `COPIES` parameter on a `sysout DD` statement that defines the data set, on the `OUTPUT JCL` statement, or on the `JES2 /*OUTPUT` statement. As many as **255** copies of the entire job-related output are obtained by coding the `COPIES` parameter on the `JES2 JOBPARM` control statement.

The number of `JOB` copies can be limited by each installation.

If you request multiple copies of a data set by coding the `COPIES` parameter on a `JES2 /*OUTPUT`, `OUTPUT JCL`, or `sysout DD` statement **and** the `JOBPARM` control statement, JES2 output processing gives the **product** of the requested amount for each `SYSOUT` data set. For example, if you request two copies of the entire job output (`COPIES=2` on the `JOBPARM` statement) and three copies of a certain output data set (`COPIES=3` on a `sysout DD` statement or `OUTPUT JCL` statement), you will receive two copies of the entire job output but will

receive a total of six copies of the output data set. However, if the data set has been written directly to an output device, held, spun off, or processed by an external writer, it is no longer a job-related data set and is not affected by the COPIES parameter on the JOBPARM statement. In this case, you would receive three copies of the requested output data set.

For the 3800 printer, you can also specify on the sysout DD statement, the OUTPUT JCL statement, and the JES2 /\*OUTPUT statement how the copies of the output data set are to be grouped. Each group value of the COPIES parameter specifies the number of copies of each individual page that is to be printed before copies of the next page are printed. The total number of copies printed equals the sum total of the group values. The system allows a maximum of eight group values.

## Requesting Multiple Copies of an Output Data Set Using JES3

You can control the number of hard copies of the system output data sets that JES3 produces. You can request as many as **254** copies of an output data set by coding the COPIES parameter on the sysout DD statement defining the data set, or up to **255** copies by coding the COPIES parameter on the JES3 // \*FORMAT PR control statement or on the OUTPUT JCL statement.

For the 3800 printer, you can also specify on the sysout DD statement, on an OUTPUT JCL statement, or on the JES3 // \*FORMAT PR statement how the copies of the output data set are to be grouped. Each group value of the COPIES parameter specifies the number of copies of each individual page that is to be printed before copies of the next page are printed. The total number of copies printed equals the sum total of the group values.

## Requesting Copy Modification

When using the 3800 printer, you can modify selected copies of output by specifying a copy modification module name in the MODIFY parameter on the sysout DD statement, on the OUTPUT JCL statement, on the JES3 // \*FORMAT PR control statement, or on the JES2 /\*OUTPUT control statement. This allows the printing of predefined data on all pages of a specific copy or copies of a data set. For example, you may want to vary column headings or explanatory remarks on different copies of the same printed page of data. Copies might also be personalized with the recipient's name, address, and other desired information. Blanks or printable characters, such as asterisks, might also be used to suppress the printing of variable data on particular copies of a page. (This is a function done in other printers by using short or spot carbon in the forms set.)

The predefined data is created as a copy modification module and stored on SYS1.IMAGELIB using the IEBIMAGE utility program. For information on using IEBIMAGE, see the *IBM 3800 Printing Subsystem Programmer's Guide*.

## Requesting Printer Form and Character Control

When requesting that an output data set be printed, you can give the job entry subsystem special instructions on how to print the data set. You can request:

- A special output form.
- A special character set or arrangement, when output is being printed by a 3211, 3203 Model 5, or 1403 printer with the universal character set feature or by a 3800 printer.

- A specific FCB (forms control buffer) module, which controls how many lines per inch are printed and the length of the form, when the data set is written to a 3211, 3203 Model 5, or 3800 printer.
- A specific FCB (forms control buffer) module, which controls how many lines per inch are printed and the length of the form, when the data is written to a remote job processing (RJP) printer supported by systems network architecture (SNA) or to a 3211, 3203 Model 5 printer or a 3800 printer.
- A specific carriage control tape, when the data set is written to a 1403 printer.
- A test for printer overflow and spacing.
- Interpretation of punch output on the 3525.

*Note:* **JES2** treats the 3203 Model 5 printer the same as a 3211 printer with the following exceptions:

- The UCSs (universal character sets) for the 3203 Model 5 are the same as for the 1403 printer.
- The 3203 Model 5 printer does not support indexing; therefore JES2 indexing commands are ignored.
- You cannot explicitly identify the 3203 Model 5 printer to JES2 via JES2 initialization parameters. MVS passes the 3203 Model 5 identification to JES2 via the UCB.

For further information on UCSs and UCBs, see *SPL: Data Management*.

### Requesting a Special Output Form

To request special forms for output data set printing, include the form name in the SYSOUT parameter on the DD statement defining the data set; or code the FORMS parameter on the OUTPUT JCL statement, on the JES2 /\*OUTPUT control statement, or on the JES3 /\*FORMAT PR control statement. For example, assign a data set to an output class to be routed to a printer and specify the data set be printed on a special form. (Code SYSOUT=(A,,FMS2) on the DD statement.) The job entry subsystem and the external writer insure that the proper form is mounted.

For JES2, the entire job can be printed on a special form if you code the FORMS parameter on the JOBPARM statement. If you code a forms name on either the DD statement with the SYSOUT parameter, the FORMS parameter on the OUTPUT JCL statement, or the JES2 /\*OUTPUT statement, it overrides the forms name in the JOBPARM statement.

### Requesting a Special Character Set Using the UCS Feature

The universal character set (UCS) feature is requested by coding the UCS parameter on a DD statement defining an output or SYSOUT data set, or:

- For JES2, by coding the UCS parameter on the OUTPUT JCL statement or on the JES2 /\*OUTPUT control statement for SYSOUT data sets.
- For JES3, by coding the TRAIN parameter on the /\*FORMAT PR control statement.

You can request the UCS feature for different sets of characters to be printed for various applications.

To request a special character set for a 3211, 3203 Model 5, or 1403 printer, specify the code identifying the character set in the UCS parameter on a DD statement, on an OUTPUT JCL statement, or on a JES2 /\*OUTPUT statement. The codes for the IBM standard special character sets are in Figure 12-1 on page 12-128

### Requesting Character Arrangements with a 3800 Printer

Specify character-arrangement tables to be used when printing with the 3800 on the CHARS parameter of the sysout DD statement, the OUTPUT JCL statement, or:

- For JES2, on the JES2 /\*OUTPUT statement.
- For JES3, on the JES3 //FORMAT PR statement.

For the table names supplied for the 3800, see the *IBM 3800 Printing Subsystem Programmer's Guide*. See your system programmer for the selection of table names available at your installation.

When more than one character arrangement table is specified, you can code OPTCD=J as a DCB subparameter to indicate that your data line contains a table reference character for dynamically selecting the table you want. (See the description of the OPTCD subparameter for BSAM and QSAM in the topic, "The DCB Parameter.") Using the IEBIMAGE utility program, you can modify or construct character arrangement tables and graphic character modification modules to allow substitution of existing or user-designed characters. For details on using the OPTCD subparameter, see the *IBM 3800 Printing Subsystem Programmer's Guide*.

You can specify the UCS (universal character set) parameter on the same output DD statement as the CHARS parameter; do this to permit output to go to either the 3800 or to other printers. If a printer other than the 3800 is selected for output, the system ignores the CHARS parameter.

If the UCS value is supplied and the CHARS parameter is not, and you requested that the data set be printed on a 3800 printer, the UCS value is used as the default value for the missing CHARS parameter.

### Requesting Forms Control

**For a 1403 printer and printers supported by systems network architecture (SNA) remote job entry (RJE):**

Request forms control by specifying a specific carriage control tape in the FCB parameter on a sysout or output DD statement, or

- For JES2, with the FCB parameter on the OUTPUT JCL statement or the JES2 /\*OUTPUT control statement.
- For JES3, with the FCB parameter on the OUTPUT JCL statement or the CARRIAGE parameter on the //FORMAT PR control statement.

Carriage specifications are used for JES output processing only; they are ignored by the external writer.

**For a 3211 or 3203 Model 5 printer and printers supported by systems network architecture (SNA) remote job entry (RJE):**

Request specific forms control images (for example, the number of lines per page or number of characters per line) by coding an image identifier in the FCB parameter on a sysout or output DD statement, the OUTPUT JCL statement, or:

- For JES2, the JES2 /\*OUTPUT control statement.
- For JES3, the JES3 /\*FORMAT PR control statement.

You can also specify a carriage control tape for JES3 output processing in the CARRIAGE parameter on the /\*FORMAT PR control statement.

The FCB image is stored in SYS1.IMAGELIB. IBM provides two standard FCB images: STD1 and STD2. STD1 specifies that 6 lines per inch are to be printed on an 8.5-inch form. STD2 specifies that 6 lines per inch are to be printed on an 11-inch form.

*Note:* Do not specify STD1 or STD2 for JES2 or JES3 processing unless instructed to do so by your installation.

Additional FCB images can be specified by the installation. For information on IBM- and user-supplied FCB images, see *SPL: Data Management*.

**Programming notes for JES2:** JES2 treats the 3203 Model 5 printer the same as a 3211 printer with the following exceptions:

- The UCSs for the 3203 Model 5 are the same as for the 1403 printer.
- The 3203 Model 5 printer does not support indexing, so JES2 indexing commands are ignored.
- You cannot explicitly identify the 3203 Model 5 printer to JES2 via JES2 initialization parameters. MVS passes the 3203 Model 5 identification to JES2 via the UCB.

For further information on UCSs and UCBs, see *SPL: Data Management*.

**For a 3800 printer:** Request forms control by specifying an FCB module name in the FCB parameter on a sysout DD statement, the OUTPUT JCL statement, or:

- For JES2, the JES2 /\*OUTPUT control statement.
- For JES3, the JES3 /\*FORMAT PR control statement.

The FCB module is stored in SYS1.IMAGELIB. IBM provides a standard FCB module, STD3, which specifies output of 80 lines per page at 8 lines per inch on 11-inch long paper. (For a 3800 using ISO paper sizes, STD3 can be redefined by the installation.) Additional FCB modules can be specified by the installation. For information on IBM- and user-supplied FCB modules, see the *IBM 3800 Printing Subsystem Programmer's Guide*.

## Requesting Forms Overlay

The forms overlay feature of the 3800 printer allows printing of the image from a forms overlay negative together with the data being printed. This reduces the need for pre-printed forms, and for changing of forms.

Specify a forms overlay using one of the following:

- The FLASH parameter on a sysout DD statement.
- The FLASH parameter on an OUTPUT JCL statement.
- For JES2, the FLASH parameters on the JES2 /\*OUTPUT control statement.
- For JES3, the FLASH parameter on the /\*FORMAT PR control statement.

Identify the overlay to be used and the number of copies on which that overlay is to be printed. When you do not specify the FLASH parameter on either a DD statement or a JES control statement, the 3800 printer uses the default specified at JES initialization.

For information on designing and making or obtaining forms overlay negatives, see the *Forms Design Reference Guide for the IBM 3800 Printing Subsystem*.

## Bursting of Output

The optional Burster-Trimmed-Stacker of the 3800 printer separates continuous form paper into individual sheets. To specify to the operator whether the output is to go to the Burster-Trimmed-Stacker or to the continuous forms stacker, specify one of the following:

- The BURST parameter on an output or sysout DD statement.
- The BURST parameter on an OUTPUT JCL statement.
- For JES2, the BURST parameter on the JES2 /\*OUTPUT control statement.
- For JES3, the BURST parameter on the /\*FORMAT PR control statement.



## Chapter 8. Guide to Special Data Sets

You can define data sets to satisfy a special purpose. Such data sets are usually defined with a special ddname, a specific data set name, or a specific parameter.

This section includes eight topics:

- Creating and using private and temporary libraries.
- Requesting an abnormal termination dump.
- Defining a dummy data set.
- Using virtual input/output (VIO) for temporary data sets.
- Entering data through the input stream.
- Virtual storage access method (VSAM) data sets.
- Creating and retrieving indexed sequential (ISAM) data sets.
- Creating and retrieving generation data sets.
- Creating and using a subsystem data set.

### Creating and Using Private and Temporary Libraries

A library is simply a partitioned data set — a data set in direct access storage that is divided into partitions, called members, each of which can contain a program or part of a program. Each partitioned data set contains a directory (or index) that the control program can use to locate a program in the library. All programs that can be executed must exist in a library; that is, they must be members of a partitioned data set.

A private library is a partitioned data set that contains user-written programs. To inform the system that a program exists in a private library code a DD statement defining that library. You can define a private library to be used throughout the job by coding a DD statement with the ddname JOBLIB, or you can define a library to be used in a specific step by coding a DD statement with the ddname STEPLIB. The library defined by a JOBLIB or STEPLIB DD statement is searched prior to the system libraries (such as SYS1.LINKLIB) for the program to be executed (that is, the program named in the PGM= field of the EXEC statement).

A temporary library is a partitioned data set created during the job to store a program, as a member of the partitioned data set, until it is executed in a following step. For example, if in the job you want to assemble, link edit, and then execute a program, make the output of the linkage editor a member of a library. Any library that is created and deleted in the same job is a temporary library.

Code the PGM parameter as the first parameter on the EXEC statement to execute a program contained in a library.

If the program exists in a private library, code PGM=program name and either a JOBLIB or STEPLIB DD statement. If the program exists in a temporary library, code either

PGM = \*.stepname.ddname or PGM = \*.stepname.procstepname.ddname. Ddname is a temporary library created in and pointed to by stepname and procstepname. They identify the job step or job step and procedure step defining the library. If you define a private library, the system looks in that library for the program you want executed.

This chapter describes how to code JCL statements to create or retrieve private and temporary libraries. Complete information on creating a partitioned data set, and on adding members to and deleting members from a partitioned data set appears in *Data Management Services Guide*.

## Creating a Private Library

Use the JOBLIB DD statement to create a private library. The JOBLIB DD statement must appear immediately after the JOB statement and any JES statements. Do not use the ddname JOBLIB unless you are defining a private library. The library defined with a JOBLIB DD statement is automatically available to every step in the job. (The STEPLIB DD statement is included among the DD statements in a step and is available only to that step unless you pass the library or redefine it in subsequent steps; since the library on a JOBLIB DD statement is available to every step, it is easier to create a library with the JOBLIB DD statement.)

When creating the library on the JOBLIB DD statement, you are creating a partitioned data set. Steps in the job must add members to the library before those members (programs) can be used by subsequent steps.

On the JOBLIB DD statement, assign the library a name in the DSNNAME parameter, give unit and volume information in the UNIT and VOLUME parameters (a partitioned data set must be contained on one direct access volume; if, however, you make a nonspecific volume request, you need not code the VOLUME parameter), request space for the entire library in the SPACE parameter, and assign a data set status and disposition in the DISP parameter. Code NEW as the data set status and either CATLG or PASS as the data set disposition. When you specify CATLG, the library is cataloged, available throughout the job, and kept at the end of the job. When specifying PASS, the library is available throughout the job, but is deleted at job termination. (If you do not code a disposition, or code a disposition other than CATLG or PASS, the system assumes DELETE. This means that the library will be deleted at the end of the first step and will not be available to any later job steps.) You must also code the DCB parameter if complete data control block (DCB) information is not included in the processing program.

## Adding Members to a Private Library

For a job step to add members to the library code a DD statement that defines the library and names the member to be added to the library.

In the DSNNAME parameter, follow the library name with the name of the program being added to the library, for example, DSNNAME = LIBRARY(PROGRAM).

Do not code the SPACE parameter; request space for the entire library on the JOBLIB DD statement.

In the DISP parameter, specify MOD as the data set status; the partitioned data set already exists since you created it in the JOBLIB statement, and you are lengthening it with a new member. If you cataloged the library in the JOBLIB DD statement, that is, coded DISP = (NEW,CATLG), do not specify CATLG again when adding a member: you need not code a second disposition at all. For a cataloged library, you do not have to specify unit and

volume information, except in one instance: if you are adding a member to the library in the first step of the job, supply unit and volume information; the library is not cataloged until the first step completes execution. Refer to the JOBLIB DD statement for unit and volume information by coding VOL = REF = \*.JOBLIB.

In the following example, the JOBLIB DD statement creates a library named GROUPLIB; STEP1 adds the program RATE to the library; STEP2 calls the program RATE:

```
//EG      JOB  MSGLEVEL=1
//JOBLIB DD  DSN=GROUPLIB,DISP=(NEW,CATLG),
//          UNIT=3350,VOL=SER=727104,
//          SPACE=(CYL,(50,3,4))
//STEP1   EXEC PGM=FINDD
//ADDPGMD DD  DSN=GROUPLIB(RATE),DISP=MOD,
//          VOL=REF=*.JOBLIB
//STEP2   EXEC PGM=RATE
```

In STEP1, the system looks for the program named FIND in SYS1.LINKLIB — the private library created on the JOBLIB DD statement does not actually exist until a member is added to it. In STEP2, the system looks for the program named RATE first in the private library.

## Retrieving an Existing Private Library

If you are retrieving several programs from one library (several steps in the job will be using the library), use the JOBLIB DD statement to define the library: the library will be available in every step of the job for which you do not code a STEPLIB DD statement. The JOBLIB DD statement must appear immediately after the JOB statement. To make a library available in a single step, define the library on a STEPLIB DD statement. The STEPLIB DD statement is included with the DD statements for a step (in no specific order) and is available only to that step, unless you pass the library and retrieve it in a subsequent step. Use the ddnames JOBLIB and STEPLIB only when defining private libraries.

The system will search for a program in the private library you define. If both JOBLIB and STEPLIB DD statements appear in a job, the STEPLIB definition supersedes the JOBLIB definition; that is, the private library defined by the JOBLIB DD statement is not searched for any step that contains the STEPLIB definition. If you want the JOBLIB definition ignored but the step does not require use of another private library, define a system library on the STEPLIB DD statement:

```
//STEPLIB DD DSN=SYS1.LINKLIB,DISP=SHR
```

Retrieve a private library as you would any partitioned data set: if the library is cataloged, or in the case of a STEPLIB definition, passed from a previous step, you need not specify unit and volume information; otherwise, you must code the UNIT and VOLUME parameters.

For both cataloged and uncataloged libraries, code:

- the DSN parameter, specifying the name of the library
- the DCB parameter, if complete data control block information is not included in the data set label
- the DISP parameter, specifying data set status and disposition.

Normally, you will want to specify SHR as the data set status: SHR indicates that the data set is old, but also allows other jobs to simultaneously use the library. All references to the library in the job must specify SHR if the data set is to be shared; do not code SHR, however, if you will be adding members to the library in the job. (A more thorough discussion of sharing a data set is included in the topic "Insuring Data Set Integrity.") Code PASS as the data set disposition for a library defined on the JOBLIB DD statement: PASS makes the library available throughout the job. (If you do not code a disposition, the system assumes PASS.) For a library defined on a STEPLIB DD statement, code any valid disposition, depending on how you want the data set treated after its use in the job step: for example, if the library is not cataloged, and you want it to be cataloged, code CATLG; if you want the library deleted, code DELETE.

The following job includes both JOBLIB DD and STEPLIB DD statements:

```
//CAMILLE JOB   MSGLEVEL=1
//JOBLIB  DD   DSNNAME=LIB5.GRP4,DISP=SHR
//STEP1   EXEC PGM=FINN
//STEP2   EXEC PGM=GATHER
//STEPLIB DD   DSNNAME=ACCOUNTS,DISP=(SHR,KEEP),
//          UNIT=3350,VOL=SER=727104
```

- In STEP1, the system searches the library named LIB5.GRP4, defined on the JOBLIB DD statement, for the program named FINN.
- In STEP2, the system searches the library named ACCOUNTS, defined on the STEPLIB DD statement, for the program named GATHER.

Add a program to an existing library by coding a DD statement in a job step that defines the library and names the program to be added — see "Adding Members to a Private Library" for details on coding this DD statement. The new member must be added to the library before it can be executed (the step that adds the program to the library must precede the step that calls the program). Do not code SHR as the data set's status when modifying the library.

### Concatenating Private Libraries

If the job uses programs contained in several libraries, you can concatenate these libraries on one JOBLIB DD statement or one STEPLIB DD statement; all the libraries concatenated must be existing libraries. Omit the ddname from all the DD statements defining the libraries, except the first:

```
//JOBLIB DD   DSNNAME=D58.LIB12,DISP=(SHR,PASS)
//          DD   DSNNAME=D90.BROWN,DISP=(SHR,PASS),
//          UNIT=3330,VOL=SER=411731
//          DD   DSNNAME=A03.EDUC,DISP=(SHR,PASS)
```

This entire group must appear immediately after the JOB statement. When concatenating libraries using STEPLIB as the ddname, the entire group appears as part of the DD statements for the step.

The system will search the libraries for the program in the order in which the DD statements defining the libraries are coded.

## Using Private Catalogs

Use access method services to define private user catalogs, as explained in *VSAM Programmer's Guide*. A JOBCAT or STEPCAT is a private catalog that is searched prior to the system catalog whenever a DD statement does not specify unit and volume serial information for a data set. JOBCAT applies to each step of a job in which a STEPCAT has not been specified.

To locate a data set, VSAM searches catalogs in the following order:

1. User catalogs specified in the current job step (STEPCAT), or if no user catalogs are specified for the job step user catalogs specified in the current job (JOBCAT).
2. A CVOL or user catalog indicated by the first qualifier of the data set name, if any.
3. The master catalog.

## Temporary Libraries

Temporary libraries are libraries that are created and deleted within the job. It is not necessary to define a temporary library on a JOBLIB DD or STEPLIB DD statement: simply code a DD statement creating a partitioned data set and adding the program to it in the step that produces the program. You can then retrieve this program in a later step. (You can also use the VIO facilities to define temporary data sets. For more information, refer to "Defining a VIO Temporary Data Set" later in this section.)

For example, STEP2 illustrated below calls the program IEWL, which link edits object modules to form a load module that can be executed. Place the results of the linkage edit step in a library so that a subsequent step can use those results. Since the results are not a program other jobs will call, it is logical to place the program in a temporary library:

```
//STEP2    EXEC PGM=IEWL
//SYSLMOD  DD  DSNAME=&&PARTDS(PROG),UNIT=3350,
//          DISP=(NEW,PASS),SPACE=(1024,(50,20,1))
//STEP3    EXEC PGM=*.STEP2.SYSLMOD
```

Call the program in STEP3 by naming the step in which the library was created and the name of the DD statement that defines the program as a member of a library. If STEP2 had called a procedure, and the DD statement named SYSLMOD was included in PROCSTEP3 of the procedure, you would code PGM=\*.STEP2.PROCSTEP3.SYSLMOD.

## Requesting an Abnormal Termination Dump

To obtain a dump in the event of abnormal termination of a job step, code a DD statement defining a dump data set. The name of the DD statement must be `SYSABEND`, `SYSMDUMP`, or `SYSUDUMP`. If more than one of these DD statements is present, the system uses the last one (provided each has a different ddname).

When MVS encounters dump requests with duplicate ddnames, processing is as follows:

- Under JES2 the job fails with message IEC912I.
- Under JES3
  - If both DD statements request JES3- or jointly-managed devices, the job is cancelled during the JES3 interpreter phase.
  - If only one or neither statement requests JES3- or jointly-managed devices, the job fails with message IEC912I.

To change the type of dump request DD statement in a cataloged procedure, add a DD statement with a ddname that is not the same as the ddname of the dump request in the procedure.

`SYSABEND`, `SYSMDUMP`, and `SYSUDUMP` DD statements can each provide a dump containing the processing program's virtual storage areas, the system nucleus, the entire system queue area, all local system queue areas, and any active link pack area (LPA) modules for the failing task. If either the Generalized Trace Facility (GTF) or the System Trace is active, the dump will contain their records. In addition, if your installation permits dumping of the common storage area, a `SYSMDUMP` DD statement can provide a dump containing those parts of the CSA that the failing program is authorized to reference.

You can take subsequent `SYSMDUMPs` to the same data set if the data set name is `SYS1.SYSMDPxx` (where xx can be 00 - FF) and the data set disposition is SHR. Before attempting to take subsequent `SYSUDUMPs` see "Rules for Coding" under "The `SYSABEND`, `SYSMDUMP` and `SYSUDUMP` Facilities" in this publication.

If your program has issued an `ABEND` macro, or if you have written a recovery routine, you can determine what dump options you want, in addition to the installation defaults, and define them in a dump option list on the `ABEND` macro or on a `SETRP` macro issued by your recovery routine. How to do this is explained in *Supervisor Services and Macro Instructions*.

Dumps with more data per page are available with the 3800 Printing Subsystem. By specifying `CHARS=DUMP` on the `SYSABEND` or `SYSUDUMP` DD statement, the dump is formatted in a 204-character line containing 64 bytes of storage. If `FCB=STD3` is specified, the page is printed at 8 lines per inch. The dump program recognizes only `STD3` for producing 8 lines per inch.

Descriptions of dumps and information on reading dumps are included in the *Debugging Handbook* and *Diagnostic Techniques*.

To print the dump that was produced for a `SYSABEND` or `SYSUDUMP` DD statement, either assign the dump to an output class using the `SYSOUT` parameter on the DD statement, or code the `UNIT` parameter and specify the printer on which you want the dump printed. To store the

dump, define the data set as you would any other data set, coding the DSNAME, DISP, UNIT, and VOLUME parameters. If the data set will go to a direct access device, code the SPACE parameter.

The dump taken for a SYSMDUMP DD statement is machine-readable (unformatted) and must be stored on either a magnetic tape unit or a direct access device. If the job or step is running with nonpageable virtual storage (ADDRSPC=REAL) on the JOB or EXEC statements, the SYSMDUMP output must be directed to a VIO data set.

To format and print a dump taken for a SYSMDUMP DD statement, use the AMDPRDMP service aid, which is documented in *SPL: Service Aids*, or IPCS, which is documented in *Interactive Problem Control System (IPCS) System Information*. Do not print the dump by sending it to SYSOUT=A because the output will be unformatted and difficult to read.

If you are using IPCS to format and print a dump taken for a SYSMDUMP DD statement, the data set disposition specified will produce the following results:

- DISP=MOD on a SYSMDUMP DD statement permits the collection of dump information pertaining to each of multiple events that occur during one job step.

Use of DISP=MOD produces a data set that only IPCS can process (although IPCS's COPYDUMP subcommand can generate AMDPRDMP input from any dump collected in such a data set).

- DISP=NEW or DISP=OLD on a SYSMDUMP DD statement permits the collection of only the dump information pertaining to the last of multiple events that occur during one job step.

Use of either DISP=NEW or DISP=OLD produces a data set that either IPCS or AMDPRDMP can process.

If a private data set is specified for SYSABEND or SYSUDUMP and more than one dump is possible, specify the data set with a disposition of MOD because it will be closed after each dump.

## Defining a Dummy Data Set

To save processing time, you might not want a data set to be processed every time the job is executed. For example, while testing a program, you might want to suppress the writing of an output data set until you are sure it will contain meaningful output; you might want to skip the reading of a data set to be used only once a week. When you define a dummy data set, input/output operations are bypassed, disposition processing is not performed, and devices and storage are not allocated to the data set.

Define a dummy data set by:

- Coding the **DUMMY** parameter on the **DD** statement or
- Assigning the data set name **NULLFILE** in the **DSNAME** parameter on the **DD** statement

### Coding the **DUMMY** Parameter

Code **DUMMY** as the first parameter on the **DD** statement. **DUMMY** is a positional parameter: it must precede all keyword parameters on the **DD** statement.

When you code the **DUMMY** parameter, the system ignores all other parameters on the **DD** statement, except the **DCB** parameter. (The parameters are checked for syntax, however; if a parameter is coded incorrectly, a JCL error message is issued.) Therefore, although you can code **UNIT**, **VOLUME**, and **DISP**, no device or external storage is allocated to the data set and no disposition processing is performed. The **DCB** parameter must be coded if you would code it for normal I/O operations. For example, when an **OPEN** routine requires a **BLKSIZE** specification to obtain buffers and **BLKSIZE** is not specified in the **DCB** macro instruction, you should supply this information in the **DCB** parameter on the **DD** statement.

When a **DD** statement that overrides a procedure **DD** statement contains the **DUMMY** parameter, all of the parameters coded on the procedure **DD** statement are nullified, except for the **DCB** parameter.

When you want the data set to be processed, replace the **DD** statement containing the **DUMMY** parameter with a **DD** statement containing the parameters required to define the data set. When a procedure **DD** statement contains the **DUMMY** parameter, nullify it by coding the **DSNAME** parameter on the overriding **DD** statement and assigning a data set name other than **NULLFILE**.

If you request unit or volume affinity with a dummy data set, the data set requesting affinity is assigned a dummy status. (Unit and volume affinity are described in the topic "Requesting Units and Volumes.")

If unit affinity (**UNIT=AFF=ddname**) to a **DD** statement is requested before the **ddname** is defined within the job stream, the system treats the requesting **DD** statement as a **DUMMY DD**.

For example:

```
//STEP EXEC PGM=TKM
//DD1 DD DDNAME=DD5
//DD2 DD DSN=A,DISP=OLD
//DD3 DD DSN=C,DISP=SHR,UNIT=AFF=DD1
//DD5 DD DSN=B,DISP=SHR
```



1. The ddname for step DD1 will be defined at DD5.
2. Step DD3 requests unit affinity with step DD1. Because the ddname in step DD1 has yet to be defined, the system treats DD3 as a DUMMY DD.

### **Coding DSNAME = NULLFILE**

Assigning the name NULLFILE in the DSNAME parameter has the same effect as coding DUMMY. The data set is assigned a dummy status; no device or storage is allocated and no disposition processing is performed. All parameters except DSNAME and DCB are ignored. (The parameters are checked for syntax, however; if a parameter is coded incorrectly, a JCL error message is issued.) Code the DCB parameter when defining a dummy data set if you would code it for normal I/O operations.

When you want the data set to be processed, replace the name NULLFILE with another data set name. (Assigning names to data sets is described under “Specifying the DSNAME Parameter.”)

### **Requests to Read or Write a Dummy Data Set**

When the program asks to read a dummy data set, an end-of-data-set exit is taken immediately. When the program requests that the data set be written, the request is recognized but no data is transmitted. VSAM supports dummy data sets for both read and write processing. Otherwise, use the basic sequential access method (BSAM) or queued sequential access method (QSAM) when requesting to write a dummy data set; if any other access method is used, the job is terminated.

If you define a data set as a dummy data set, the DISP parameter, if coded, is ignored and disposition processing is not performed.

If you define a data set using the DUMMY parameter, you should not concatenate other data sets to it. When the processing program asks to read a dummy data set, the system takes an end-of-data-set exit immediately and ignores any concatenated data set.

## Using Virtual Input/Output (VIO) for Temporary Data Sets

Temporary data sets can be handled by a facility called virtual I/O (VIO). (VIO processing does not apply to nontemporary data sets.) Data sets for which VIO is specified reside within the paging space; however, to a problem program and the access method, the data sets appear to reside on some other real direct access storage device.

During system generation, the installation can define new and/or existing unit names as eligible for VIO. You can code these unit names on a DD statement that defines a data set to specify VIO processing for any system-named temporary data set.

### Defining a VIO Temporary Data Set

The DD statement for a VIO data set is similar to the DD statement for a conventional temporary data set, with the following exceptions:

- Volume serial numbers cannot be specified for VIO.
- The UNIT keyword in the VIO DD statement must specify a name that has been defined as eligible for VIO.
- If the SPACE parameter is not coded for virtual I/O data sets, the default value is 10 primary and 50 secondary blocks with an average block length of 1000. Up to a one-volume limit, you will always obtain the full amount of space requested (that is, the primary quantity plus fifteen secondary requests). If the primary quantity for space is larger than the simulated volume, the job will fail. If the primary request is met, but the secondary request is greater than one volume, you will get up to one volume. When allocating by average block length for a VIO data set, the secondary request is determined by the average block length specified in the SPACE parameter.
- VIO does not support ISAM or VSAM, so you cannot specify ISAM or VSAM indicators in the DSORG parameter of a DD statement for a VIO data set. The “area” of an ISAM data set cannot be specified in the DSNAME parameter.
- The DISP parameter must be specified as NEW or PASS when creating a data set. Do not specify KEEP or CATLG in the DISP parameter for a temporary VIO data set.
- The DSNAME parameter need not be coded, but if it is, it must only be specified in && name form.
- A VIO data set will be allocated to non-VIO if any of the above exceptions are violated, except the SPACE parameter request.
- The unit count subparameter of the UNIT parameter is ignored.

*Note:* Empty input data sets and SUBALLOC requests are not eligible for VIO.

## Backward References to VIO Data Sets

If the referring DD statement (`VOL=REF=`) defines a temporary data set and refers to a DD statement that defines a VIO data set, the data set is assigned to external page storage as a VIO data set.

If the referring DD statement requests unit affinity but does not define a temporary data set, the referring statement takes on the unit specification of the DD statement to which reference is made, but not the VIO status.

The following examples assume that you defined the user-assigned group name `SYSDA` and the device type name `3330` at system generation (with the `UNITNAME` macro instruction) as group names eligible for VIO processing.

The data sets defined by the following DD statements are assigned to external page storage for VIO processing:

```
//DD1 DD UNIT=SYSDA
```

---

```
//DD2 DD UNIT=3330
```

---

```
//DD3 DD DSN=&&A,DISP=(NEW),SPACE=(CYL,(30,10)),UNIT=SYSDA
```

---

```
//DD1 DD UNIT=SYSDA  
//DD2 DD VOL=REF=*.DD1
```

---

```
//DDA DD UNIT=SYSDA  
//ddb DD VOL=REF=*.DDA,UNIT=3330
```

In each of the following examples, the data set defined on the first DD statement is assigned to external page storage for VIO processing. The second DD statement does not request VIO because it defines a nontemporary data set.

```
//DD1 DD UNIT=SYSDA  
//DD2 DD DSN=NONTEMP,DISP=(,KEEP),  
// VOL=REF=*.DD1,SPACE=(CYL,10)
```

---

```
//DD1 DD UNIT=SYSDA  
//DD2 DD DSN=TEMP,DISP=(,KEEP),VOL=SER=665431,  
// SPACE=(CYL,10),UNIT=AFF=DD1
```

## Using Virtual Input/Output (VIO) to Pass Temporary Data Sets Among Job Steps

VIO data sets are passed in the same way as conventional data sets. For example, the following JCL statements show the DD statements required by VIO for a job with compilation, linkage editor, and execution steps. The VIO data sets in the various job steps are defined as system-named temporary data sets. The unit name PAGEDEV has been defined as eligible for VIO (via the UNITNAME macro instruction during system generation).

```
(1) //ASM          EXEC PGM=IFOX00
    ---
    ---
    //ASM.SYSGO DD   DSN=&&OBJ,UNIT=PAGEDEV,DISP=(NEW,PASS)
(2) //LKED          EXEC PGM=IEWL
    //SYSLIN        DD   DSN=&&OBJ,DISP=(OLD,DELETE)
    //              DD   DDNAME=SYSIN
    //SYSLMOD        DD   DSN=&&LOAD(A),DISP=(NEW,PASS),UNIT=PAGEDEV,
    //              DCB=DSORG=PO,SPACE=(TRK,(5,5,1))
    ---
    ---
(3) //GO            EXEC PGM=*.LKED.SYSLMOD
```

*Note:*

You must code the SPACE parameter on the //SYSLMOD DD card to ensure that directory space is allocated.

## Entering Data Through the Input Stream

You can enter data through the input stream by coding either the `*` or `DATA` parameters on the `DD` statement. The `DD *` statement precedes data in an input stream; the `DD DATA` statement precedes data in an input stream when the data contains JCL statements. The `DLM` parameter allows the use of a delimiter other than `/*` to terminate data defined in the input stream. Code this parameter on either the `DD *` or `DD DATA` parameters.

You can include several distinct groups of data in the input stream. Two types of data are for job steps specifying a program name or for job steps that call a cataloged or in-stream procedure. However, cataloged and in-stream procedures cannot contain `DD` statements defining data in the input stream.

## VSAM Data Sets

Virtual Storage Access Method (VSAM) is an access method for use with direct-access storage. It is different from all other access methods and you need to take certain precautions when coding VSAM data sets. You can use JCL parameters to identify cataloged VSAM data sets and to specify options for them. To process a VSAM data set, specify a DD statement in the form:

```
//ddname DD DSNAME=dsname,DISP={OLD|SHR}
```

The **DSNAME** parameter specifies the name of the VSAM cluster to which the data set you are processing belongs.

The **DISP** parameter must specify either OLD or SHR because the data set is cataloged.

You cannot use JCL to create VSAM data sets; you must use access method services commands. VSAM data sets cannot be passed within a job.

Some DD parameters and subparameters have different meanings for VSAM data sets. For example, VSAM data sets are described by the access-method control block (ACB), not the DCB. Therefore, the DCB parameter is not applicable to VSAM. Parameters that can be used without modification are explained in Figure 8-1 on page 8-15. Parameters that either should not be used or should be used only with caution are explained in Figure 8-2 on page 8-16. The STEPCAT and JOBCAT facilities identify user catalogs. These parameters are similarly used for all data sets and are discussed in this section under "Creating and Using Private Libraries."

VSAM has one JCL parameter of its own: AMP. The AMP parameter takes effect when the data set defined by the DD statement is opened. It has subparameters for:

- Overriding operands specified with the ACB, EXLST, or the GENCB macro instructions
- Supplying operands missing from the ACB or GENCB macro instruction
- Indicating checkpoint/restart options
- Indicating options when using ISAM macro instructions to process a key-sequenced data set
- Indicating that the data set is a VSAM data set when you specify unit and volume information or DUMMY in a DD statement
- Indicating that you want VSAM to supply storage dumps of the access-method control block(s) that identify this DD statement

Parameter	Subparameter	Comment
<b>DDNAME</b>	<i>ddname</i>	No special considerations for VSAM.
<b>DISP</b>	<b>SHR</b>	Indicates that you are willing to share the data set with other jobs. This subparameter alone, however, does not guarantee that sharing will take place. See <i>VSAM User's Guide</i> for a full description of data-set sharing.
	<b>OLD</b>	No special considerations for VSAM.
<b>DSNAME</b>	<i>dsname</i>	No special considerations for VSAM.
<b>DUMMY</b>		No special considerations for VSAM, except that an attempt to read results in an end-of-data condition, and an attempt to write results in a return code that indicates the write was successful. If specified, AMP=AMORG must also be specified.
<b>DYNAM</b>		No special considerations for VSAM.
<b>FREE</b>		No special considerations for VSAM.
<b>PROTECT</b>		No special considerations for VSAM.
<b>UNIT</b>	<i>address</i>	Must be the address of a valid device for VSAM (2305, 3330V, 3330, 3340, 3344, 3350, 3375, or 3380). If not, OPEN will fail
	<i>type</i>	Must be a type supported by VSAM (2305, 3330, 3330V, 3340, 3350, 3375, or 3380). If not, OPEN will fail.
	<i>group</i>	Must be a group supported by VSAM. If not, OPEN will fail.
	<i>p</i>	There must be enough units to mount all of the volumes specified. If sufficient units are available, UNIT=p can improve performance by avoiding the mounting and demounting of volumes.
	<i>unit count</i>	If the number of devices requested is greater than the number of volumes on which the data set resides, the extra devices are allocated anyway. If a key-sequenced data set and its index reside on unlike devices, the extra devices are allocated evenly between the unlike device types. If the number of devices requested is less than the number of volumes on which the data set resides but greater than the minimum number required to gain access to the data set, the devices over the minimum are allocated evenly between unlike device types. If devices beyond the count specified are in use by another task but can be shared and have mounted on them volumes containing parts of the data set to be processed, they will also be allocated to this data set.
<b>VOLUME</b>	<b>DEFER</b>	No special considerations for VSAM.
	<b>PRIVATE</b>	No special considerations for VSAM.
	<b>SER</b>	The volume serial number(s) used in the access method services DEFINE command for the data set must match the volume serial numbers in the VOLUME=SER specification when the data set is defined. After a VSAM data set is defined, the volume serial number(s) need not be specified on a DD statement to retrieve or process the data set. If, however, VOLUME=SER and UNIT=type are specified, only those volumes specifically named are initially mounted. Other volumes may be mounted when they're needed if at least one of the units allocated to the data set is not shareable or the unit count is equal to the total number of volumes allocated to the data set. A unit is unshareable when unit count is less than the number of volume serial numbers specified or when DEFER is specified. If VOLUME=SER is specified and the data set is cataloged in a user catalog, the user catalog should be defined as a JOBCAT or a STEPCAT for the current step.

**Figure 8-1. DD parameters used with VSAM**

Parameter	Subparameter	Comment
BURST		Because this parameter applies only to unit-record devices, it does not apply to VSAM.
CHARS		Because this parameter applies only to unit-record devices, it does not apply to VSAM.
CHKPT		VSAM ignores CHKPT
COPIES		Because this parameter applies only to unit-record devices, it does not apply to VSAM.
DATA		Because there is no way to get VSAM data into the input stream, this parameter is not applicable to VSAM.
DCB	All	The access-method control block, not the DCB, describes VSAM data sets; therefore, the DCB parameter is not applicable to VSAM. An access-method control block is generated by an ACB or GENCB macro, and can be modified by a MODCB macro.
DEST		Specify DEST only with the SYSOUT parameter.
DISP	CATLG	VSAM data sets are cataloged and uncataloged as a result of an access method services command; if CATLG is coded, a message is issued, but the data set is not cataloged.
	DELETE	VSAM data sets are deleted as a result of an access method services command; if DELETE is coded, a message issued, but the data set is not deleted.
	MOD	For VSAM data sets, MOD is treated as if OLD were specified, except for processing with an ISAM program, in which case MOD indicates resume load.
	KEEP	Because KEEP is implied for VSAM data sets, it need not be coded.
	NEW	VSAM data spaces are initially allocated as a result of the access method services DEFINE command. If NEW is specified, the control program also allocates space, and it is never used by VSAM. Moreover, an access method services request for space may fail if the DISP=NEW acquisition of space causes too little space to remain available.
	UNCATLG	VSAM data sets are cataloged and uncataloged as a result of access method services commands; if UNCATLG is coded, a message is issued, but the data set is not uncataloged.
	PASS	The PASS parameter is not applicable to VSAM. However, because there is no error checking, coding PASS for a key-sequenced data set whose index resides on a like device does not result in an error. If a VSAM data set and its index reside on unlike devices, the results are unpredictable. In either case, the data set is not passed.
DLM		Because there is no way to get VSAM data into the input stream, this parameter is not applicable to VSAM.
DSNAME	<i>dsname(area-name)</i>	The name is used; area-name is ignored.
	<i>dsname(generation)</i>	The name is used; generation is ignored.
	<i>dsname(member)</i>	The name is used; member is ignored.
	All temporary <i>dsnames</i>	Because VSAM data sets are built by access method services, which used the data-set name supplied in the DEFINE command, temporary names cannot be used with VSAM.
	All backward DD references of the form <i>*.ddname</i>	If the object referred to is a cluster and the data set and index reside on unlike devices, the results of a backward DD reference are unpredictable.

Figure 8-2 (Part 1 of 2). DD parameters you should avoid with VSAM



Parameter	Subparameter	Comment
<b>FCB</b>		Because this parameter applies only to unit-record devices, it does not apply to VSAM.
<b>FLASH</b>		Because this parameter applies only to unit-record devices, it does not apply to VSAM.
<b>LABEL</b>	<b>BLP, NL, NSL</b>	Because these subparameters have no meaning for direct-access devices, they do not apply for VSAM data sets, which all reside on direct-access storage.
	<b>IN</b>	Because IN is used to override DCB subparameters and the DCB parameter does not apply to VSAM data sets, IN does not apply.
	<b>OUT</b>	Because OUT is used to override DCB subparameters and the DCB parameter does not apply to VSAM data sets, OUT does not apply.
	<b>NOPWREAD</b>	The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NOPWREAD specification in the LABEL parameter.
	<b>PASSWORD</b>	The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NOPWREAD specification in the LABEL parameter.
	<b>SL, SUL</b>	Although these parameters apply to direct-access storage devices, SL is always used for VSAM, whether you specify SL, SUL, or neither.
<b>MODIFY</b>		Because this parameter applies only to unit-record devices, it does not apply to VSAM.
<b>MSVGP</b>		You must explicitly specify the volume serial number.
<b>SPACE</b>		VSAM data spaces are initially allocated as a result of the access method services DEFINE command. If SPACE is specified, therefore, an extent is allocated that is never used by VSAM. Moreover, an access method services request for space may fail as a result of the SPACE acquisition of space.
<b>SYSOUT</b>		If SYSOUT is coded with a mutually exclusive parameter (for example, DISP), the job step is terminated with an error message.
<b>UCS</b>	All	Because this parameter applies only to unit-record devices, it does not apply to VSAM.
<b>UNIT</b>	<b>AFF</b>	Use this subparameter carefully. If the cluster components, the data and its index, reside on unlike devices, the results of UNIT = AFF are unpredictable.
<b>VOLUME</b>	<b>REF</b>	Use this subparameter carefully. If the referenced volumes are not a subset of those contained in the catalog record for the data set, the results are unpredictable.
	<i>vol-seq-number</i>	Results are unpredictable.
	<i>volume-count</i>	This subparameter is used to request some number of nonspecific volumes. Because all VSAM volumes must be specifically defined before processing, volume count is not applicable to VSAM data sets.
	*	Because there is no way to get VSAM data into the input stream, this parameter has no application with VSAM.

**Figure 8-2 (Part 2 of 2). DD parameters you should avoid with VSAM**

## Creating and Retrieving Indexed Sequential Data Sets

Indexed sequential (ISAM) data sets are created and retrieved using special subsets of DD statement parameters and subparameters. Each data set can occupy up to three different areas of space:

1. **Prime area** — This area contains data and related track indexes. It exists for all indexed sequential data sets.
2. **Overflow area** — This area contains overflow from the prime area when new data is added. It is optional.
3. **Index area** — This area contains master and cylinder indexes associated with the data set. It exists for any indexed sequential data set that has a prime area occupying more than one cylinder.

Indexed sequential data sets must reside on direct access volumes. The data set can reside on more than one volume and the device types of the volumes may in some cases differ.

### Creating an Indexed Sequential Data Set

One to three DD statements can be used to define a new indexed sequential data set. When using three DD statements to define the data set, each DD statement defines a different area and the areas must be defined in the following order:

1. Index area
2. Prime area
3. Overflow area

When using two DD statements to define the data set, the areas must be defined in the following order:

1. Index area
2. Prime area

or

1. Prime area and, optionally, index area
2. Overflow area

When using one DD statement to define the data set, you are defining the prime area and, optionally, the index area.

When more than one DD statement is used to define the data set, assign a ddname only to the first DD statement; the name field of the other statements must be blank.

The only DD statement parameters that can be coded when defining a new indexed sequential data set are the DSNAME, UNIT, VOLUME, LABEL, DCB, DISP, and SPACE parameters. When to code each of these parameters and what restrictions apply are described in the following paragraphs.

## The DSNNAME Parameter

The DSNNAME parameter is required on any DD statement that defines a new temporary or nontemporary indexed sequential data set. To identify the area you are defining, you follow the DSNNAME parameter with the area.

For example,

```
DSNAME=name ( INDEX )
DSNAME=name ( PRIME )
or
DSNAME=name ( OVFLOW ) .
```

If you are using only one DD statement to define the data set, code

```
DSNAME=name ( PRIME )
or
DSNAME=name
```

When you reuse previously allocated space to create an ISAM data set, the DSNNAME parameter must contain the name of the old data set to be overlaid.

## The UNIT Parameter

The UNIT parameter is required on any DD statement that defines a new indexed sequential data set unless VOLUME = REF = reference is coded. You must request a direct access device in the UNIT parameter and must not request DEFER.

If there are separate DD statements defining the prime and index areas, request the same number of direct access devices for the prime area as there are volumes specified in the VOLUME parameter. You request only one direct access volume for an index area and one for an overflow area.

A DD statement for the index area or overflow area can request a device type different than the type requested on the other statements.

Another way to request a device is to code UNIT = AFF = ddname (except for new data sets), where the named DD statement requests the direct access device or device type you want.

## The VOLUME Parameter

The VOLUME parameter is required if you want an area of the data set written on a specific volume or the prime area requires the use of more than one volume. (If the prime area and index area are defined on the same statement, you cannot request more than one volume on the DD statement.) Either supply the volume serial number or numbers in the VOLUME parameter or code VOLUME = REF = reference. In all cases, you can use the VOLUME parameter to request a private volume (VOL = PRIVATE).

*Notes:*

1. *If a new ISAM data set is being created with a nonspecific volume request and its DSNNAME already exists on a volume eligible for allocation, the job might fail due to duplicate names on the volume. If the old data set that has a duplicate name resides on another volume than the one selected for the new data set, however, the new data set is not affected and will be added to the volume. You can correct job failures because of duplicate data set names by scratching the old data set or by renaming the new data set before resubmitting the job.*

2. Allocation fails a nonspecific volume request for any new ISAM data set when there is not sufficient space on any of the volumes eligible for allocation.
3. If the first volume selected by allocation to satisfy a request for a new ISAM data set does not contain sufficient storage to satisfy the request, allocation does not attempt to find another volume with sufficient space if the request is of the following types.
  - A request for multiple volumes or units.
  - A request uses the second, third, or subsequent DD statement you used to define the dataset.

### The LABEL Parameter

The LABEL parameter need only be coded to specify a retention period (EXPDT or RETPD) or password protection (PASSWORD).

### The DCB Parameter

You must code the DCB parameter on every DD statement that defines an indexed sequential data set. At minimum, the DCB parameter must contain DSORG = IS or DSORG = ISU. Other DCB subparameters can be coded to complete the data control block if the processing program does not complete it.

When more than one DD statement is used to define the data set, code all the DCB subparameters on the first DD statement. Code,

```
DCB=* . ddname
```

on the remaining statement or statements; ddname is the name of the DD statement that contains the DCB subparameters.

When reusing previously allocated space and recreating an ISAM data set, desired changes in the DCB parameter must be coded on the DD statement. Although you are creating a new data set, some DCB subparameters cannot be changed if you want to use the space the old data set used. The DCB subparameters you can change are:

BFALN	DSORG	NCP	RECFM
BLKSIZE	KEYLEN	NTM	RKP
CYLOFL	LRECL	OPTCD	

### The DISP Parameter

If you are creating a new data set and not reusing preallocated space, the DISP parameter need be coded only if you want to:

Keep the data set: code DISP=(,KEEP)  
 Catalog the data set: DISP=(,CATLG)  
 Pass the data set: DISP=(,PASS)

If you are reusing previously allocated space and recreating an ISAM data set, code DISP=OLD. The newly created data set will overlay the old one.

To update an existing ISAM data set, code DISP=OLD. If you code SHR, the data set will not open correctly.

In order to catalog the data set by coding `DISP=(,CATLG)` or to pass the data set by coding `DISP=(,PASS)`, you must define the data set on only one DD statement. If you define the data set on more than one DD statement and the volumes containing the data set correspond to the same device type, use the access method services DEFINE command to catalog the data set. For details, refer to *Access Method Services*.

## The SPACE Parameter

The SPACE parameter is required on any DD statement that defines a new indexed sequential data set. Use either the recommended nonspecific allocation technique or the more restricted absolute track (ABSTR) technique. If you use more than one DD statement to define the data set, each DD statement must request space using the same technique.

### Nonspecific Allocation Technique

You must request the primary quantity in cylinders (CYL). When the DD statement that defines the prime area requests more than one volume, each volume is assigned the number of cylinders requested in the SPACE parameter.

One of the subparameters of the SPACE parameter, the “index” subparameter, is used to indicate how many cylinders are required for an index. When you use one DD statement to define the prime and index areas and you want to explicitly state the size of the index, code the “index” subparameter.

You can code the CONTIG subparameter in the SPACE parameter. However, if you code CONTIG on one of the statements, you must code it on all of them.

You cannot request a secondary quantity for an indexed sequential data set. Also, you cannot code the subparameters RLSE, MXIG, ALX, and ROUND.

### Absolute Track Technique

The number of tracks requested must be equal to one or more whole cylinders. The address of the beginning track must correspond with the first track of a cylinder other than the first cylinder on the volume. When the DD statement that defines the prime area requests more than one volume, space is allocated for the prime area beginning at the specified address and continuing through the volume and onto the next volume until the request is satisfied. (This can only be done if the volume table of contents of the second and all succeeding volumes is contained within the first cylinder of each volume.)

Use the “index” subparameter of the SPACE parameter to indicate how many tracks an index requires. The number of tracks specified must be equal to one or more cylinders. When you use one DD statement to define the prime and index areas and you want to explicitly state the size of the index, code the “index” subparameter.

*Note:* If the indexed sequential data set is to reside on more than one volume and an error is encountered as the volumes are being allocated to the data set, follow this procedure before resubmitting the job: Use the IEHPROGM utility program to scratch the data set labels on any of the volumes to which the data set was successfully allocated. This utility program is described in *Utilities*.

## Area Arrangement of an Indexed Sequential Data Set

When creating an indexed sequential data set, the arrangement of the areas is based on two criteria:

1. The number of DD statements used to define the data set
2. What area each DD statement defines

An additional criterion is used when you do not include a DD statement that defines the index area: Is an index size coded in the SPACE parameter on the DD statement that defines the prime area?

Figure 18-5 on page 18-7 illustrates the different arrangements that can result based on the criteria listed above. In addition, it indicates what restrictions apply on the number and types of devices that can be requested.

## Retrieving an Indexed Sequential Data Set

If all areas of an existing indexed sequential data set reside on volumes of the same device type, you can retrieve the entire data set with one DD statement. If the index or overflow resides on a volume of a different device type, use two DD statements. If the index and overflow reside on volumes of different device types, use three DD statements to retrieve the data set. The DD statements are coded in the following order:

1. First DD statement - defines the index area
2. Second DD statement - defines the prime area
3. Third DD statement - defines the overflow area

The **only** DD statement parameters that you may code when retrieving an indexed sequential data set are:

DSNAME	VOLUME	DISP
UNIT	DCB	

When to code each of these parameters and what restrictions apply are described in the following paragraphs.

### The DSNAME Parameter

The DSNAME parameter is always required. Identify the data set by its name; however, it is not necessary to include the terms INDEX, PRIME, or OVFLOW when retrieving an indexed sequential data set. If the data set was passed from a previous step, identify it by a backward reference.

### The UNIT Parameter

The UNIT parameter must be coded unless the data set resides on one volume and was passed. You identify in the UNIT parameter the device type and how many of these devices are required.

If the data set resides on more than one volume and the volumes correspond to the same device type, you need only one DD statement to retrieve the data set. Request one device per volume in the UNIT parameter. If the index or overflow area of the data set resides on a different type of volume than the other areas, you must use two DD statements to retrieve the data set. On

one DD statement, request the device type required to retrieve the index or overflow area. On the other DD statement, request the device type and the number of devices required to retrieve the prime area and the overflow area if the overflow area resides on the same device type. If the index and the overflow areas reside on device types different from the prime area, you need a third DD statement.

### **The VOLUME Parameter**

The VOLUME parameter must be coded unless the data set resides on one volume and was passed from a previous step. Identify in the VOLUME parameter the serial numbers of the volumes on which the data set resides. Code the serial numbers in the same order as they were coded on the DD statements used to create the data set.

### **The DCB Parameter**

The DCB parameter must always contain DSORG=IS or DSORG=ISU. You do not have to code other DCB subparameters if the data set is passed from a previous step or is cataloged. However, you can code other DCB subparameters to complete the data control block if it has not been completed by the processing program.

### **The DISP Parameter**

The DISP parameter must always be coded. The first subparameter of the DISP parameter must be SHR or OLD. When you are updating an existing ISAM data set, code DISP=OLD. If you specify DISP=SHR, the data set will not open correctly. You can, optionally, assign a disposition as the second subparameter.

## Examples of Creating and Retrieving an Indexed Sequential Data Set

The following job creates an indexed sequential data set on one 3330 volume.

```
//ISAMJOB JOB    , ,MSGLEVEL=(1,1),PERFORM=25
//STEP1  EXEC   PGM=INCLUDE
//DD1    DD     DSN=DATASET1(INDEX),DISP=(NEW,KEEP),UNIT=3330,
//        VOL=SER=777777,SPACE=(CYL,(10),,CONTIG),
//        DCB=(DSORG=IS,RECFM=F,LRECL=80,RKP=1,KEYLEN=8)
//        DD     DSN=DATASET1(PRIME),DISP=(NEW,KEEP),UNIT=3330,
//        VOL=REF=*.DD1,SPACE=(CYL,(25),,CONTIG),DCB=*.DD1
//        DD     DSN=DATASET1(OVFLOW),DISP=(NEW,KEEP),UNIT=3330,
//        VOL=REF=*.DD1,SPACE=(CYL,(25),,CONTIG),DCB=*.DD1
```

---

The following job includes the DD statements required to retrieve the indexed sequential data set created above.

```
//RETRISAM JOB    , ,MSGLEVEL=(1,1),PERFORM=25
//STEP1  EXEC   PGM=RETRIEVE
//DDISAM DD     DSN=DATASET1,DCB=DSORG=IS,UNIT=3330,DISP=OLD,
//        VOL=SER=777777
```

---

The following job creates an indexed sequential data set on one 3330 and two 3350 volumes.

```
//ISAMJOB JOB    , ,MSGLEVEL=(1,1),PERFORM=25
//STEP1  EXEC   PGM=IEFISAM
//DDISAM DD     DSN=DATASET2(INDEX),DISP=(NEW,KEEP),UNIT=3330,
//        VOL=SER=888888,SPACE=(CYL,10,,CONTIG),DCB=(DSORG=IS,
//        RECFM=F,LRECL=80,RKP=1,KEYLEN=8)
//        DD     DSN=DATASET2(PRIME),DISP=(,KEEP),UNIT=3350,
//        VOL=SER=999999,SPACE=(CYL,10,,CONTIG),DCB=*.DDISAM
//        DD     DSN=DATASET2(OVFLOW),DISP=(,KEEP),UNIT=3350,
//        VOL=SER=AAAAAA,SPACE=(CYL,10,,CONTIG),DCB=*.DDISAM
```

---

The following job includes the DD statements required to retrieve the indexed sequential data set created above.

```
//RERISAM JOB    , ,MSGLEVEL=(1,1),PERFORM=25
//STEP1  EXEC   PGM=IEFISAM
//DDISAM DD     DSN=DATASET2,DCB=DSORG=IS,DISP=OLD,UNIT=3330,
//        VOL=SER=888888
//        DD     DSN=DATASET2,DCB=DSORG=IS,DISP=OLD,UNIT=(3350,2),
//        VOL=SER=(999999,AAAAAA)
```



## Creating and Retrieving Generation Data Sets

A generation data set is one of a collection of successive, historically related, cataloged data sets known as a generation data group. The system keeps track of each data set in a generation data group as it is created so that new data sets can be chronologically ordered and old ones easily retrieved.

To create or retrieve a generation data set, identify the generation data group name in the DSNNAME parameter and follow the group name with a relative generation number. When you create a generation data set, the relative generation number tells the system whether this is the first data set being added during the job, the second, the third, etc. When retrieving a generation data set, the relative generation number tells the system how many data sets have been added to the group since this data set was added.

Relative generation numbers are obtained from the catalog as it existed:

- For JES2, at the beginning of the first step that specifies the generation data set by relative generation number.

*Note:* In a shared DASD environment, if two or more jobs running on different systems simultaneously create new generations of the same data set, one of the jobs could fail with a JCL error.

- For JES3, when the job is set up, and again by the operating system at the beginning of the first step that specifies the generation data set by relative generation number. If the most recent data set is not the same at both times, the results are unpredictable.

A generation data group can consist of cataloged sequential and direct data sets residing on tape volumes, direct access volumes, or both. Generation data sets can have like or unlike DCB attributes and data set organizations. If the attributes and organizations of all generations in a group are identical, the generations can be retrieved together as a single data set (up to 255 data sets can be retrieved in this way). The retrieval order is last in-first out. If the generation data group resides on more than one device type, all generations cannot be retrieved together.

### Building a Generation Data Group Base Entry

Before defining the first generation data set, you must build a generation data group base entry in a VSAM, OS CVOL, or ICF catalog. This provides for as many generation data sets (up to 255) as you would like to have in the generation data group. The system uses the base to keep track of the chronological order of the generation data sets. Use the access method services DEFINE command to build generation data group bases in a VSAM or ICF catalog. This command is described in *Access Method Services*.

Another requirement of generation data groups is that a data set label list exist. The system uses this label to refer to DCB attributes when you define a new generation data set. There are two ways to satisfy this requirement: (1) create a model data set label on the same volume as the catalog before defining the first generation data set; or (2) use the DCB parameter to refer the system to an existing cataloged data set each time you define a new generation data set.

## Creating a Model Data Set Label

To create a model data set label, define a data set and request that it be placed on the same volume as the generation data group base. This ensures that there is always a data set label on the same volume as the catalog to which the system can refer.

The name assigned to the data set can be the same or different than the name assigned to the generation data group. (If you assign the same name for both, the data set associated with the model data set label cannot be cataloged.) Request a space allocation of zero tracks or cylinders. The DCB attributes that can be supplied are DSORG, OPTCD, BLKSIZE, LRECL, KEYLEN, and RKP.

You need not create a model data set label for every generation data group whose indexes reside on the same volume. Instead, create one model data set label to be used by any number of generation data groups. When creating a generation data set, specify the name of the model in the DCB parameter; follow the name with a list of all the DCB subparameters required for the new generation data set that are different than specified in the model; that is, DCB=(dsname,list of attributes).

## Referring the System to a Cataloged Data Set

If there is a cataloged data set residing on the same volume as the generation data group index and you are sure that data set will exist as long as you are adding data sets to the generation data group, you need not create a model data set label. When creating a generation data group, specify the name of the cataloged data set in the DCB parameter by coding DCB=dsname. If all the DCB attributes are not contained in the label of the cataloged data set, or if you want to override certain attributes, follow the data set name with these attributes; that is, DCB=(dsname,list-of-attributes).

## Creating a Generation Data Set

When defining a new generation data set, always code the DSNAME, DISP, and UNIT parameters. Other parameters you might code are the VOLUME, SPACE, LABEL, and DCB parameters.

### The DSNAME Parameter

In the DSNAME parameter, code the name of the generation data group followed by a number enclosed in parentheses. This number must be 1 or greater. If this is the first data set you are adding to a particular generation data group during the job, code +1 in parentheses. Each time during the job you add a data set to the same generation data group, increase the number by one. When the first character is a plus (+), the remaining digits (three digits or less) must not exceed 255.

Any time you refer to this data set later in the job, use the same relative generation number as was used earlier. At the end of the job, the system updates the relative generation numbers of all generations in the group to reflect the additions.

*Note:* Unpredictable results can occur if you use a relative generation number that causes the actual generation number to exceed G9999.

### **The DISP Parameter**

Assign new generations a status of **NEW** and a disposition of **CATLG** in the **DISP** parameter; that is, **DISP=(NEW,CATLG)**. If the **DISP** parameter is not specified, the system assumes **DISP=(NEW,DELETE)** and the new generation will be deleted at the end of the step.

### **The UNIT Parameter**

The **UNIT** parameter is required on any **DD** statement that defines a new generation data set unless **VOLUME=REF=reference** is coded. In the **UNIT** parameter, identify the type of devices you want (tape or direct access).

### **The VOLUME Parameter**

You can assign a volume in the **VOLUME** parameter or let the system assign one for you. The **VOLUME** parameter can also be used to request a private volume (**PRIVATE**) and to indicate that more volumes may be required (volume count).

### **The SPACE Parameter**

Only code the **SPACE** parameter when the generation data set is to reside on a direct access volume.

### **The LABEL Parameter**

You can specify label type, password protection (**PASSWORD**), and a retention period (**EXPDT** or **RETPD**) in the **LABEL** parameter. If the data set will reside on a tape volume and is not the first data set on the volume, specify a data set sequence number.

### **The DCB Parameter**

A model data set label that has the same name as the group name may exist. If this is so, and if the label contains all the attributes required to define this generation, you need not code the **DCB** parameter. If all the attributes are not contained in the label, or if you want to override certain attributes, code **DCB=(list of attributes)**.

If a model data set label has a different name than the group name and if the label contains all the attributes required to define this generation data set, you need code only the name of the data set associated with the model data set label. Code the name in the **DCB** parameter; that is, **DCB=dsname**. If all the attributes are not contained in the label, or if you want to override certain attributes, follow the data set name with these attributes; that is, **DCB=(dsname,list of attributes)**.

If a model data set label does not exist, you must code the name of a cataloged data set that resides on the same volume as the generation data group index. If all the attributes are not contained in the label for this data set, or if you want to override certain attributes, follow the data set name with these attributes.

## Retrieving a Generation Data Set

To retrieve a generation data set, always code the DSNNAME and DISP parameters. Other parameters you might code are the UNIT, LABEL, and DCB parameters.

### The DSNNAME Parameter

Using the DSNNAME parameter you can retrieve a single generation data set or all of the generation data sets in the generation data group.

**Retrieving a Single Generation Data Set:** If you want to retrieve a single generation data set, code in the DSNNAME parameter the name of the generation data group followed by a number enclosed in parentheses. This number can be a maximum of four characters. The number coded depends on which generation data set is to be retrieved. To retrieve the most recent data set, code a zero (0). If the first character is zero (0), the remaining characters must be zero or blanks.

To retrieve data sets created prior to the most recent data set, code a minus value (-nnn). When nnn is a minus value, the remaining digits (3 digits or less) must not exceed 255. The value of nnn is determined by the relation of the desired data set to the most current data set. Minus one (-1) refers to the data set created immediately preceding the most recent data set; minus two (-2) refers to the data set created preceding the data set identified by the minus one value.

For example:

WEEKLY.PAYROLL is the name of a generation data group.

DSN=WEEKLY.PAYROLL(0) The most recent generation data set.  
DSN=WEEKLY.PAYROLL(-1) Last week's generation data set.  
DSN=WEEKLY.PAYROLL(-2) Generation data set of two weeks ago.

Relative generation numbers are maintained by the system only when generation data sets are specified using relative generation numbers.

*Note:* When you are retrieving a generation data set within a started task, and the generation data set is cataloged in a private catalog or control volume (CVOL), coding a relative generation number causes unpredictable results.

**Retrieving All Generation Data Sets:** If you want to retrieve all generations of a generation data group as a single data set, specify the generation data group name without a generation number in the DSNNAME parameter: for example,

DSNAME=WEEKLY.PAYROLL

where WEEKLY.PAYROLL is the generation data group name.

To retrieve all generations in this manner, the DCB attributes and data set organization of all generations must be identical.

When you specify the generation data group name without a generation number, the operating system treats your request as a concatenation of all existing data sets in the generation data group, starting with the most recent data set and ending with the oldest data set. In addition, all data sets, except the most recent, will have unit affinity to the most recent data set. For further information, see "Concatenating Data Sets" on page 2-9.

For generation data groups residing on tape, when you specify the generation group name without a generation number (GDG ALL request), and request parallel mounting in the UNIT parameter, the system mounts all volumes of the **first** generation only.

For generation data groups residing on DASD (including MSS) volumes, when you specify the generation group name without a generation number (GDG ALL request), and request parallel mounting in the UNIT parameter, the system mounts all volumes of **all** generations.

The relative generation number of the most recent data set is obtained from the catalog as it existed:

- **For JES2**, at the beginning of the first step that specifies the generation data set by relative generation number.

*Note:* In a shared DASD environment, if two or more jobs running on different systems simultaneously create new generations of the same data set, one of the jobs could fail with a JCL error.

- **For JES3**, when the job is set up, and again by the operating system at the beginning of the first step that specifies the generation data set by relative generation number. If the most recent data set is not the same at both times, the results are unpredictable.

*Note:* When retrieving a generation data set within a started task, and the generation data set is cataloged in a private catalog or control volume (CVOL), coding a relative generation number causes unpredictable results.

### **The DISP Parameter**

You must always code the DISP parameter. The first subparameter of the DISP parameter must be OLD, SHR, or MOD. You can, optionally, assign a disposition as the second subparameter. The second subparameter must be specified for a generation data group. Do not code PASS as the second subparameter when you retrieve all generations of a generation data group as a single data set. In all such retrievals, the unit and volume information for each generation level is obtained from the catalog, and not from the pass mechanism. If you code the DISP subparameter MOD for a generation data set and the specified relative generation does not exist in the catalog, the operating system changes the disposition to NEW.

### **The UNIT Parameter**

Code the UNIT parameter when you want more than one device assigned to the data set. Code the number of devices you want in the unit count subparameter, or, if the data set resides on more than one volume and you want as many devices as there are volumes, code P in place of the unit count subparameter.

### **The VOLUME Parameter**

Use the VOLUME parameter to request a private volume (PRIVATE) and to indicate that more volumes might be required (volume count). A volume serial number specified for an old generation data group is ignored; the system obtains the volume serial number from the catalog except for deferred checkpoint restart (see "Submitting a Job for Restart," below).

## The LABEL Parameter

Code the LABEL parameter when the data set resides on tape and has other than standard labels. If the data set is not the first data set on the volume, specify the data set sequence number. If the data set sequence number is coded for a GDG ALL request, it is ignored; the data set sequence number will be taken from the catalog.

## The DCB Parameter

Code DCB=(list of attributes) when the data set has other than standard labels and DCB information is required to complete the data control block. Do not code DCB=dsname when retrieving a generation data set.

## Deleting and Uncataloging Generation Data Sets

In a multiple-step job, if you attempt to delete or uncatalog any generation data set **except** the oldest member of a generation data group, catalog management can lose orientation within the data group. This could cause the deletion, uncataloging, or retrieval of the wrong data set when you later refer to a specific generation. Therefore, if you delete a generation data set in a multiple step job, do not refer to any previous (older) generation in later job steps.

Also, we recommend that in a multiple-step job, you catalog or uncatalog generation data sets using JCL instead of IEHPROGM or a user program. Because allocation/deallocation routines access the catalog during job execution, they are unaware of the functions performed by IEHPROGM or a user program; you might get unpredictable results.

## Submitting a Job for Restart

Certain rules apply when you refer to generation data sets in a job submitted for restart (the RESTART parameter is coded on the JOB statement).

If your installation has installed MVS/System Product Release 3 (5740-XYN or 5740-XXS) or subsequent releases without installing Data Facility/Device Support Release 1 Enhancements (5740-AM7), do not use the checkpoint/restart facility.

If you have installed Data Facility/Device Support with either of the MVS/System Products, you can use the checkpoint/restart facility with certain restrictions.

For additional information concerning these restrictions, see *Checkpoint/Restart*.

*For step restart:* generation data sets that were created and cataloged in steps preceding the restart step must not be referred to (by means of the same relative generation numbers that were used to create them) in the restart step or in steps following the restart step. Instead, you must refer to a generation data set by means of its present relative generation number. For example, if the last generation data set created and cataloged was assigned a generation number of +2, it would be referred to as 0 in the restart step and in steps following the restart step. In this case, the generation data set assigned number of +1 would be referred to as -1.

*For checkpoint restart:* If generation data sets created in the restart step were kept instead of cataloged, that is, DISP=(NEW,CATLG,KEEP) was coded, you can, during checkpoint restart, refer to these data sets and generation data sets created and cataloged in steps preceding the restart step by means of the same relative generation numbers that were used to create them.

*For Deferred Checkpoint Restart:* the system does not use the catalog to obtain the volume serial numbers for a generation data group. Therefore, if you changed the volume serial numbers in the catalog between the original submission of the job and the restart, you **must** code volume serial information.

## Example of Creating and Retrieving Generation Data Sets

The following job step includes the DD statements that could be used to add three data sets to a generation data group.

```
//STEPA EXEC PGM=PROCESS
//DD1 DD DSNAME=A.B.C(+1),DISP=(NEW,CATLG),UNIT=3400-6,
// VOL=SER=13846,LABEL=(,SUL)
//DD2 DD DSNAME=A.B.C(+2),DISP=(NEW,CATLG),UNIT=3330,
// VOL=SER=10311,SPACE=(480,(150,20))
//DD3 DD DSNAME=A.B.C(+3),DISP=(NEW,CATLG),UNIT=3350,
// VOL=SER=28929,SPACE=(480,(150,20)),
// DCB=(LRECL=120,BLKSIZE=480)
```

The first two DD statements do not include the DCB parameter because a model data set label exists on the same volume as the generation data group index and has the same name as the generation data group (A.B.C). Since the DCB parameter is coded on the third DD statement, the attributes LRECL and BLKSIZE, along with the attributes included in the model data set label, are used.

---

The following job includes the DD statements required to retrieve the generation data sets defined above when no other data sets have been added to the generation data group.

```
//JWC JOB CLASS=B
//STEP1 EXEC PGM=REPORT9
//DDA DD DSNAME=A.B.C(-2),DISP=OLD,LABEL=(,SUL)
//DDB DD DSNAME=A.B.C(-1),DISP=OLD
//DDC DD DSNAME=A.B.C(0),DISP=OLD
```

## Creating and Using a Subsystem Data Set

Use the DD SUBSYS parameter to:

- Specify the name of the subsystem that will process the associated subsystem data set
- Specify up to 254 subsystem-defined parameters that describe the subsystem data set to the subsystem

The subsystem processes the subsystem-defined parameters according to its own rules.

When you specify the SUBSYS parameter, the subsystem may alter the significance of certain DD statement parameters. To determine if a particular subsystem alters the significance of any DD statement parameters, and if it does, to determine which statements are affected and how they are affected, refer to the documentation for the subsystem.

If you specify the DUMMY parameter, MVS invokes the specified subsystem to syntax check the subsystem-defined parameters. If the syntax is acceptable, MVS assigns a dummy status to the data set and processes the request as a dummy request.

If you request unit affinity to a subsystem data set, MVS substitutes SYSALLDA as the UNIT parameter specification.



## Chapter 9. Guide to Cataloged and In-Stream Procedures

Applications that require many control statements and that are used on a regular basis can be considerably simplified through the use of cataloged and in-stream procedures. A cataloged procedure is a set of job control statements that are placed in a partitioned data set known as the procedure library; an in-stream procedure is a set of job control statements that are placed in the input stream within a job. You can execute a procedure simply by specifying its name on an EXEC statement in your job. This section describes how to write and use cataloged and in-stream procedures.

This section includes the following topics:

- Writing Cataloged and In-Stream Procedures
- Identifying an In-Stream Procedure
- Identifying Procedure Statements on an Output Listing

### Writing Cataloged and In-Stream Procedures

Cataloged and in-stream procedures are simply the job control statements needed to perform an application. A procedure contains one or more procedure steps, each consisting of an EXEC statement that identifies the program to be executed, DD statements defining the data sets to be used or produced by the program, and, optionally, OUTPUT JCL statements defining the processing options the system is to use for output data sets. The program requested on the EXEC statement must exist in a private library or the system library. If you do request a program that is contained in a private library, the procedure step calling that program must include a DD statement with the ddname STEPLIB that defines the private library; the STEPLIB DD statement is described in the chapter, “Creating and Using Private and Temporary Libraries.”

Cataloged and in-stream procedures cannot contain:

- EXEC statements that refer to other cataloged or in-stream procedures
- JOB, delimiter, or null statements
- DD statements defining private libraries to be used throughout the job (DD statements with the ddname JOBLIB)
- DD statements defining data in the input stream (statements including the \* or DATA parameters)
- OUTPUT JCL statements prior to the first EXEC statement within the procedure.

- Any JES2 control statements; they are ignored
- Any JES3 control statements; they are ignored in cataloged procedures only

## Identifying an In-Stream Procedure

To identify an in-stream procedure, code the PROC and PEND job control statements.

On the PROC statement, which must be the first statement in an in-stream procedure, assign the procedure a name. This name is the name that a programmer codes to call the procedure. Optionally, you can also assign default values to symbolic parameters contained in the procedure and code comments. (A symbolic parameter is a symbol preceded by an ampersand that stands for a parameter, a subparameter, or a value in a procedure; including symbolic parameters in a procedure is described in detail in "Symbolic Parameters" on page 2-15. If you do not assign default values to symbolic parameters on the PROC statement, you cannot code comments. The simplest form of the PROC statement, to identify an in-stream procedure named PAYROLL, would be:

```
//PAYROLL PROC
```

The PEND statement marks the end of the in-stream procedure. You can include a name on the PEND statement and comments, but these are optional. Both of the following examples are acceptable:

```
//ENDPROC PEND end of in-stream procedure
//          PEND
```

The following example illustrates an in-stream procedure named SALES consisting of two procedure steps. Note that STEP2 includes a STEPLIB DD statement to define the private library in which the program JUGGLE can be found.

```
//SALES    PROC
//STEP1    EXEC PGM=FETCH
//DD1A     DD  DSN=RECORDS(BRANCHES),DISP=OLD
//DD1B     DD  DSN=RECORDS(MORGUE),DISP=MOD
//STEP2    EXEC PGM=JUGGLE
//STEPLIB DD  DSN=PRIV.WORK,DISP=OLD
//DD2A     DD  SYSOUT=A
//          PEND
```

## Placing a Cataloged Procedure in a Procedure Library

The major difference between cataloged and in-stream procedures is where they are placed. Cataloged procedures must be placed in a procedure library before being used. In-stream procedures are placed within the job that calls them.

A procedure library is simply a partitioned data set containing cataloged procedures. IBM supplies a procedure library named SYS1.PROCLIB, but the installation can have additional procedure libraries with different names. When a programmer calls a cataloged procedure, he receives a copy of the procedure; therefore, a cataloged procedure can be used simultaneously by more than one programmer.

To add a procedure to a procedure library, use the IEBUPDTE utility program. You can also use the IEBUPDTE utility to permanently modify an existing procedure. (Before modifying an existing cataloged procedure, however, you must notify the operator; he must delay the

execution of jobs that might use the procedure library while it is being updated.) Details on using the IEBUPDTE utility appear in *Utilities*. In JES3, you can use the procedure library update feature to modify an existing procedure. The UPDATE parameter on the JES3 MAIN statement indicates that a procedure library is being updated and causes all jobs using the library to be held until the update is complete.

Before placing or modifying a cataloged procedure in a procedure library, test it without overriding any parameters to ensure that the procedure statements are syntactically correct. Additionally, test the procedure by first running it as an in-stream procedure. In-stream testing enables you to detect any errors in overridden parameters prior to cataloging the procedure.

No special job control statements are used to identify a cataloged procedure. The PEND statement is never used and the PROC statement is optional. You need code the PROC statement as the first statement in a cataloged procedure only when you want to assign default values to symbolic parameters. The name of the PROC statement is not necessarily the name of the cataloged procedure; you assign the procedure a name when adding it to the procedure library.

## Allowing for Changes in Cataloged and In-Stream Procedures

The usefulness of cataloged and in-stream procedures is destroyed if a programmer who uses the procedure has to permanently modify the procedure every time he wants to make a change. When writing a procedure, you can define, as symbolic parameters, those parameters, subparameters and values that are likely to vary each time the procedure is used. For details on coding symbolic parameters, see "Symbolic Parameters" on page 2-15.

## Using Cataloged and In-Stream Procedures

To use a cataloged or in-stream procedure, specify the procedure name on an EXEC statement. You can modify the procedure by adding DD statements and OUTPUT JCL statements, by overriding, adding, or nullifying parameters on EXEC, DD, and OUTPUT JCL statements, and by assigning values to symbolic parameters. Calling and modifying procedures is explained in greater detail in the following paragraphs.

## How to Call Cataloged and In-Stream Procedures

To call a cataloged or in-stream procedure, identify the procedure on the EXEC statement of the step calling the procedure; do this by coding one of the following as the first operand on the EXEC statement:

- The procedure name
- PROC= the procedure name

A cataloged procedure must exist in the procedure library before you attempt to use it. JES2 or JES3 is responsible for fetching cataloged procedures. Refer to "Scheduling a Job" on page 5-1 to see how JES2 or JES3 determines what library to select. When using an in-stream procedure, include the procedure, beginning with a PROC statement and ending with a PEND statement, with the job control language for the job; the procedure must follow the JOB statement but appear before the EXEC statement that calls it. You can include as many as fifteen uniquely named in-stream procedures in one job and can use each procedure as many times as you wish in the job.

To call a cataloged procedure named PROCESSE, you would code:

```
//CALL EXEC PROCESSE or  
//CALL EXEC PROC=PROCESSE
```

On the EXEC statement, you can also code changes you would like to make for this execution of the procedure.

## Modifying Cataloged and In-Stream Procedures

You can modify a procedure by:

- Assigning values to or nullifying symbolic parameters contained in the procedure
- Overriding, adding, or nullifying parameters on EXEC, DD, and OUTPUT JCL statements in the procedure
- Adding DD statements to the procedure
- Adding OUTPUT JCL statements to the procedure

All changes you make are in effect only during the current execution of the procedure. For a discussion of symbolic parameters, see "Symbolic Parameters" on page 2-15. Other modifications are described in the following sections.

## Modifying Parameters on an EXEC Statement

To override, add, or nullify a parameter on an EXEC statement in a procedure, identify on the EXEC statement that calls the procedure the parameter you are changing, the name of the EXEC statement on which the parameter appears, and the change to be made:

```
//CALL EXEC procedurename,parameter.procstepname=value
```

When overriding a parameter, the value coded for the parameter on the EXEC statement calling the procedure replaces the value assigned in the procedure. When adding a parameter, that parameter is used in the execution of the procedure step. When nullifying a parameter, you do not follow the equal sign with a value; the value assigned to the parameter in the procedure is ignored. All changes made are in effect only for the current execution of the procedure.

You can make more than one change to each EXEC statement in the procedure, and you can change parameters on more than one EXEC statement in the procedure. You cannot, however, change the PGM parameter. When making changes to different steps in the procedure, code all changes for one procedure step before you code changes to a subsequent step.

Test all new procedures without overriding any parameters to ensure that the procedure statements are syntactically correct and contain no invalid backward references.

*Note:* You cannot override invalid backward references or syntactical errors on an EXEC statement within a procedure with valid parameters. The system scans the original text for errors, and thus an overriding parameter does not eliminate the error.

For example, the first three EXEC statements in a procedure named IRISH are:

```
//STEP1 EXEC PGM=YEATS , PARM=' *14863 '  
//STEP2 EXEC PGM=NOLAN  
//STEP3 EXEC PGM=SYNGE , TIME=( 2 , 30)
```

and you want to make the following changes:

- Nullify the PARM parameter in STEP1.
- Add the COND parameter, specifying the test (8,LT), in STEP2.
- Change the time limit in the TIME parameter in STEP3 to 4 minutes.

On the EXEC statement calling the procedure, you would code:

```
//CALL EXEC IRISH , PARM . STEP1 = ,  
// COND . STEP2 = ( 8 , LT ) , TIME . STEP3 = 4
```

In the above example, code TIME.STEP3=1440 to nullify the TIME parameter. If you code TIME.STEP3=, the default time value for the job class is assigned.

You need not name the procedure step when changing a parameter. When you omit the name, the procedure is modified as follows:

- If the PARM parameter is coded, it applies only to the first procedure step. If a PARM parameter appears in a later EXEC statement in the called procedure, it is nullified.
- If the TIME parameter is coded, it applies to the total procedure. If the TIME parameter appears on any of the EXEC statements in the called procedure, it is nullified.
- If any other parameter is coded, it applies to every step in the called procedure. Nullifying the parameter on the EXEC statement calling the procedure causes the parameter to be ignored on every EXEC statement in the procedure; if you assign a value to the parameter on the EXEC statement calling the procedure, the parameter is overridden where it appears in the procedure and added to EXEC statements in the procedure on which it does not appear.

For example, assume the EXEC statements in the procedure named COMPUTE are:

```
//STEP1 EXEC PGM=LIST , TIME=( 1 , 30 )  
//STEP2 EXEC PGM=UPDATE , RD=NC , TIME=2  
//STEP3 EXEC PGM=CHECK , RD=RNC , COND=ONLY
```

You want to make the following changes:

1. Assign a time limit of 4 minutes to the entire procedure; TIME parameters on individual EXEC statements in the procedure will be nullified.
2. Allow automatic step restart for each step of the job by coding RD=R. The RD parameter will be added to the first step of the job and will override the RD parameters in STEP2 and STEP3.

To call the procedure and make these changes, you would code:

```
//CALL EXEC COMPUTE , TIME=4 , RD=R
```

During the processing of the JCL statements for the job, the EXEC statements appear as:

```
//STEP1 EXEC PGM=LIST, RD=R  
//STEP2 EXEC PGM=UPDATE, RD=R  
//STEP3 EXEC PGM=CHECK, RD=R, COND=ONLY
```

If any parameter change affects every step of the job (by omitting the procedure step name), you must code that parameter on the EXEC statement calling the procedure before you code changes to parameters on different steps (for which, you include the procedure step name). Time will be a total of four minutes, each step using the remaining amount of time available from the total. If more than four minutes is required, the step will abnormally terminate.

### Modifying Parameters on a DD Statement

To override, add, or nullify parameters on a DD statement in a procedure, include a DD statement containing the changes you want to make following the EXEC statement that calls the procedure. The name of the DD statement containing the changes is composed of the procedure step name and the ddname of the DD statement in the procedure:

```
//procstepname.ddname DD parameter=value
```

When overriding a parameter, the value you code replaces the value assigned to the parameter in the procedure.

When adding a parameter, the parameter is added to the DD statement in the procedure for the current execution of the procedure.

When nullifying a parameter, do not follow the equal sign with a value; that parameter in the procedure is ignored. Do not nullify a parameter when you are replacing it with a mutually exclusive parameter; it will be nullified automatically. (See Figure 18-6 on page 18-8 for a table of mutually exclusive parameters on the DD statement.)

All changes you make are in effect only for the current execution of the procedure. If you are overriding a DD statement, the system does not check for mutually exclusive parameters on the DD statement. Therefore, all procedures should be executed once without any overriding statements to ensure that they do not contain any mutually exclusive parameters.

You can change more than one parameter on a DD statement and you can change parameters on more than one DD statement in the procedure. However, the DD statements containing the changes must be coded in the same order as the corresponding DD statements in the procedure. Test all new procedures without overriding any parameters to ensure that the procedure statements are syntactically correct.

For example, the first two steps of the cataloged procedure TEA are:

```
//STEP1 EXEC PGM=SUGAR  
//DD1A DD DSNAME=DRINK, DISP=(NEW,DELETE),  
// UNIT=3400-6, VOL=SER=568998  
//DD1B DD UNIT=SYSSQ  
//STEP2 EXEC PGM=LEMON  
//DD2A DD UNIT=3350, DISP=(,PASS),  
// SPACE=(TRK,(20,2))
```

You want to make the following changes for this invocation:

1. Change the disposition on the DD statement named DD1A to CATLG.
2. Change the volume serial number on the DD statement named DD1A to a nonspecific request allowing the operating system to choose the volume.
3. Change the unit on the DD statement named DD1B to TAPE.
4. Change the SPACE parameter on the DD statement named DD2A to SPACE=(CYL,(4,1)).

When calling the procedure, you would code:

```
//CALL          EXEC TEA
//STEP1.DD1A DD  DISP=(NEW,CATLG),VOL=SER=
//STEP1.DD1B DD  UNIT=TAPE
//STEP2.DD2A DD  SPACE=(CYL,(4,1))
```

When changing DCB keyword subparameters, you need code only those subparameters you are changing. The DCB keyword subparameters you do not code (and for which you do not code a mutually exclusive subparameter) remain unchanged. For example, a DD statement named DD1 in a procedure step named STEP1 contains,

```
DCB=(BUFNO=1,BLKSIZE=800,RECFM=FM,BUFL=800)
```

To change the block size to 320 and the buffer length to 320, you would code:

```
//STEP1.DD1 DD  DCB=(BLKSIZE=320,BUFL=320)
```

The subparameters BUFNO and RECFM remain unchanged.

If a DCB positional subparameter is needed, the DCB positional subparameter must be coded on the override statement regardless of whether one exists in the statement to be overridden. To nullify a DCB positional parameter, do not code the DCB positional parameter on the override statement.

For example, a DD statement named DD2 in a procedure step named STEP2 contains DCB=(DSN1,BLKSIZE=80). To change the block size to 400, and copy other DCB information from the cataloged data set named DSN1, you would code:

```
//STEP2.DD2 DD  DCB=(DSN1,BLKSIZE=400)
```

To nullify the DCB parameter, you must nullify each subparameter. For example, if a DD statement in a procedure contains DCB=(RECFM=FB,BLKSIZE=160,LRECL=80), you must code DCB=(RECFM=,BLKSIZE=,LRECL=) in order to nullify the DCB parameter.

To nullify the DUMMY parameter, code the DSNAME parameter on the overriding DD statement and assign a data set name other than NULLFILE. To nullify all the parameters on a DD statement other than DCB, code DUMMY on the overriding DD statement.

If you code DUMMY on a DD statement, the system ignores all other parameters on the DD statement, except the DCB parameter. However, the system does syntax check all parameters so they must be correct. (The DUMMY parameter is described in detail under "Defining a Dummy Data Set.")

## Modifying Parameters on DD Statements that Define Concatenated Data Sets

When a concatenation of data sets is defined in a cataloged procedure and you attempt to override the concatenation with one DD statement, only the first (named) DD statement is overridden. To override others, you must include an overriding DD statement for each DD statement; the DD statements in the input stream must be in the same order as the DD statements in the procedure. The second and subsequent overriding statements must not be named. If you do not wish to change one of the concatenated DD statements, leave the operand field blank on the corresponding DD statement in the input stream. (This is the only case where a blank operand field for a DD statement is valid.)

For example, suppose you are calling a procedure that includes the following sequence of DD statements in STEPC:

```
//DD4 DD DSNAME=A.B.C,DISP=OLD
// DD DSNAME=STRP,DISP=OLD,UNIT=3350,VOL=SER=X12182
// DD DSNAME=TYPE3,DISP=OLD,UNIT=3350,VOLUME=SER=BL1421
// DD DSNAME=A.B.D,DISP=OLD
```

To override the DD statements that define the data sets named STRP and A.B.D, you would code:

```
//STEPC.DD4 DD
// DD DSNAME=INV.CLS,DISP=OLD
// DD
// DD DSNAME=PAL8,DISP=OLD,UNIT=3350,VOL=SER=125688
```

## Adding DD Statements to a Procedure

You can add DD statements to a procedure when calling the procedure. These additional DD statements are in effect only during the current execution of the procedure.

To add a DD statement to a procedure step, place the additional DD statement after the EXEC statement that calls the procedure and after any overriding DD statements for that step. The ddname of the DD statement identifies the procedure step to which this statement is to be added; you must assign a ddname that is different from all the ddnames in the procedure step. If you do not identify the procedure step in the ddname, the DD statement is added to the step specified by the last DD statement that contains a stepname and modifies a DD statement in the procedure. If there are no DD statements that contain stepname.ddname, then the DD statement is added to the first step of the procedure.

For example, if you use the following procedure:

```
//LINKS1 PROC
//LK1 EXEC PGM=IEWL,REGION=512K
//SYSUT1 DD SPACE=(CYL,(5,2)),UNIT=SYSDA
//SYSPRINT DD SYSOUT=A
//SYSLMOD DD DISP=OLD,UNIT=3330,VOL=SER=&SER,
// DSN=SYS1.TESTLIB
//LK2 EXEC PGM=IEWL,REGION=512K
//SYSUT1 DD SPACE=(CYL,(5,2)),UNIT=SYSDA
//SYSPRINT DD SYSOUT=A
//SYSLMOD DD UNIT=3330,VOL=SER=TSTVOL,
// DSN=COPY.TESTLIB,DISP=OLD
```



and you specify these DD statements:

```
//LK2.SYSLMOD DD DSN=COPY2.TESTLIB,UNIT=3350,  
//              VOL=SER=ATEST,DISP=SHR  
//ADD          DD DSN=SYS1.LPALIB,DISP=SHR
```

then the DD statement with the ddname of ADD would be added to step LK2. If you did not code the DD statement LK2.SYSLMOD, then the DD statement ADD would be added to step LK1.

You can use symbolic parameters on DD statements that you are adding to a procedure. However, if you are adding a DD statement to the last step of a procedure, do **not** use symbolic parameters that are not used elsewhere in the procedure.

When adding DD statements to a procedure that contains concatenated DD statements, follow the rules outlined in the previous topic “Modifying Parameters on DD Statements That Define Concatenated Data Sets.”

### Modifying Parameters on an OUTPUT JCL Statement

To override, add, or nullify parameters on an OUTPUT JCL statement in a procedure, include an OUTPUT JCL statement containing the changes you want to make following the EXEC statement that calls the procedure. The name of the OUTPUT JCL statement containing the changes is composed of the procedure step name and the name of the OUTPUT JCL statement in the procedure:

```
//procstepname.name OUTPUT parameter=value
```

When overriding a parameter, the value you code replaces the value assigned to the parameter in the procedure.

When adding a parameter, the parameter is added to the OUTPUT JCL statement in the procedure for the current execution of the procedure.

When nullifying a parameter, do not follow the equal sign with a value; that parameter in the procedure is ignored.

All changes you make are in effect only for the current execution of the procedure.

You can change more than one parameter on an OUTPUT JCL statement and you can change parameters on more than one OUTPUT JCL statement in the procedure. However, the OUTPUT JCL statements containing the changes must be coded in the same order as the corresponding OUTPUT JCL statements in the procedure. Test all new procedures without overriding any parameters to ensure that the procedure statements are syntactically correct.

For example, the first two steps of the cataloged procedure MOVE are:

```
//STEP1 EXEC PGM=TRUCK  
//OUTA OUTPUT GROUPID=RPT,BURST=YES,COPIES=6,  
//       FORMS=STD,DEST=NEWYORK  
//DD1   DD SYSOUT=A,OUTPUT=*.OUTA  
//STEP2 EXEC PGM=LOAD  
//OUTB OUTPUT COPIES=2,FORMS=IMG1,FLASH=(AB,2),  
//       DEST=ARIZ
```

You want to make the following changes for this invocation:

1. Change the BURST parameter on OUTPUT JCL statement OUTA to NO.
2. Change the COPIES parameter on OUTPUT JCL statement OUTA to 12.
3. Change the COPIES parameter on OUTPUT JCL statement OUTB to 12.
4. Nullify the FORMS parameter on OUTPUT JCL statement OUTB.
5. Change the DEST parameter on the OUTPUT JCL statement OUTB to NEWYORK.

When calling the procedure, you would code:

```
//CALL          EXEC      MOVE
//STEP1.OUTA   OUTPUT    BURST=NO,COPIES=12
//STEP2.OUTB   OUTPUT    COPIES=12,FORMS=,DEST=NEWYORK
```

*Note:* The OUTPUT parameter coded on DD statement DD1 is still valid. The output data set for DD statement DD1 receives the output processing as specified on OUTPUT JCL statement OUTA **and** the changes specified on the overriding OUTPUT JCL statement identified by //STEP1.OUTA.

## Adding OUTPUT JCL Statements to a Procedure

You can add OUTPUT JCL statements to a procedure when calling the procedure. These additional OUTPUT JCL statements are in effect only during the current execution of the procedure.

To add an OUTPUT JCL statement to a procedure step, place the additional OUTPUT JCL statement after the EXEC statement that calls the procedure and after any overriding OUTPUT JCL statements for that step. The name of the OUTPUT JCL statement identifies the procedure step to which this statement is to be added; you must assign the OUTPUT JCL statement a name that is different from all the OUTPUT JCL statement names in the procedure step. If you do not identify the procedure step in the name, the OUTPUT JCL statement is added to the step specified by the last overriding OUTPUT JCL statement that contains a stepname. If there are no overriding OUTPUT JCL statements that contain stepname.name, then the OUTPUT JCL statement is added to the first step of the procedure.

For example, if you use the following procedure:

```
//LINKS1       PROC
//LK1          EXEC      PGM=IEWL,REGION=512K
//OUTRP1       OUTPUT    BURST=YES,COPIES=2,DEST=POK
//SYSPR1       DD        SYSOUT=A
//LK2          EXEC      PGM=IEWL,REGION=512K
//OUTRP2       OUTPUT    COPIES=2,FORMS=RA
//SYSPR2       DD        SYSOUT=A
```

and you specify these OUTPUT JCL statements:

```
//STEPA        EXEC      LINKS1
//LK1.OUTRP1   OUTPUT    DEFAULT=YES,FLASH=(XA,2),DEST=HQ
//ADD1         OUTPUT    DEFAULT=YES,DEST=MONT
//LK2.OUTRP2   OUTPUT    DEFAULT=YES,FORMS=STD,DEST=HQ
//LK2.OUTRP3   OUTPUT    DEFAULT=YES,DEST=FLA
//ADD2         OUTPUT    DEFAULT=YES,COPIES=2,DEST=POK
```

- The OUTPUT JCL statement with the name LK1.OUTRP1 adds parameters to OUTPUT JCL statement OUTRP1 in procedure step LK1.

- The system adds OUTPUT JCL statement ADD1 to procedure step LK1 because it does not have a name in the form *stepname.name*.
- The OUTPUT JCL statement with the name LK2.OUTRP2 adds parameters to OUTPUT JCL statement OUTRP2 in procedure step LK2.
- The system adds OUTPUT JCL statement LK2.OUTRP3 to procedure step LK2.
- The system adds OUTPUT JCL statement ADD2 to procedure step LK2 because ADD2 does not have a name in the form *stepname.name*.

*Notes:*

1. Because OUTPUT JCL statements LK1.OUTRP1 and ADD1 have DEFAULT= YES specified, the system processes the output data set defined by DD statement SYSPR1 DD statement SYSPR2 using the processing options specified on OUTPUT JCL statements OUTRP1 and ADD1.
2. Because OUTPUT JCL statements LK2.OUTRP2, LK2.OUTRP3, and ADD2 have DEFAULT= YES specified, the system processes the output data set for DD statement SYSPR2 using the processing options specified on OUTPUT JCL statements OUTRP2, OUTRP3, and ADD1.

For more information and examples of the relationship between the DEFAULT parameter on an OUTPUT JCL statement and a DD statement with the SYSOUT DD parameter and OUTPUT DD parameter, see “OUTPUT Parameter” on page 12-103.

### Adding OUTPUT JCL statements when there is an OUTPUT DD Parameter

For example, if you use the following procedure:

```
//LINKS1    PROC
//LK1      EXEC      PGM=IEWL,REGION=512K
//OUTRP1   OUTPUT    BURST=YES,COPIES=2,DEST=POK
//SYSPR1   DD        SYSOUT=A,OUTPUT=*.OUTRP1
//LK2      EXEC      PGM=IEWL,REGION=512K
//OUTRP2   OUTPUT    COPIES=2,FORMS=RA
//SYSPR2   DD        SYSOUT=A,OUTPUT=*.LK1.OUTRP1
//SYSPR3   DD        SYSOUT=A
//LK3      EXEC      PGM=IEWL,REGION=512K
//SYSPR4   DD        SYSOUT=A,OUTPUT=*.LK1.OUTRP1
```

and you specify these OUTPUT JCL statements:

```
//STEPSA   EXEC      LINKS1
//LK1.OUTRP1 OUTPUT    DEFAULT=YES,FLASH=(XA,2),DEST=HQ
//ADD1     OUTPUT    DEFAULT=YES,DEST=MONT
//LK2.OUTRP2 OUTPUT    DEFAULT=YES,FORMS=STD,DEST=HQ
```

- The OUTPUT JCL statement with the name LK1.OUTRP1 adds parameters to OUTPUT JCL statement OUTRP1 in procedure step LK1.
- The system adds OUTPUT JCL statement ADD1 to procedure step LK1 because it does not have a name in the form *stepname.name*.
- The OUTPUT JCL statement with the name LK2.OUTRP2 adds parameters to OUTPUT JCL statement OUTRP2 in procedure step LK2.

*Notes:*

1. Because output DD statement SYSPR1 makes an explicit reference to OUTPUT JCL statement OUTRP1, the system processes the data set using the combined processing options coded on OUTPUT JCL statements OUTRP1 and LK1.OUTPUTRP1.
2. Because output DD statement SYSPR2 makes an explicit reference to OUTPUT JCL statement OUTRP1, the system processes the data set using the combined processing options coded on OUTPUT JCL statements OUTRP1 and LK1.OUTPUTRP1.
3. Because output DD statement SYSPR3 does not make an explicit reference to an OUTPUT JCL statement and because OUTPUT JCL statement LK2.OUTPUTRP2 specifies DEFAULT= YES, the system processes the data set for DD statement SYSPR3 using the combined processing options specified on OUTPUT JCL statements LK2.OUTPUTRP2 and OUTRP2.
4. Because output DD statement SYSPR4 makes an explicit reference to OUTPUT JCL statement OUTRP1, the system processes the data set using the combined processing options coded on OUTPUT JCL statements OUTRP1 and LK1.OUTPUTRP1.

If you add an OUTPUT JCL statement to procedure step LK3, you cannot refer to it from DD statement SYSPR4, because the system adds the additional OUTPUT JCL statement at the end of procedure step LK3. Because you are using the backward reference feature of JCL to refer to an OUTPUT JCL statement, the OUTPUT JCL statement must precede the DD statement that makes the reference.

If you want the system to process an output data set according to processing options on an OUTPUT JCL statement that you are adding to the procedure and the DD statement makes an explicit reference to a different OUTPUT JCL statement, you can use the following method.

For example, if you use the following procedure:

```
//LINKS1 PROC
//LK1 EXEC PGM=IEWL,REGION=512K
//OUTRP1 OUTPUT BURST=YES,COPIES=2,DEST=POK
//SYSPR1 DD SYSOUT=A,OUTPUT=* .OUTRP1
//LK2 EXEC PGM=IEWL,REGION=512K
//OUTRP2 OUTPUT COPIES=2,FORMS=RA
//SYSPR2 DD SYSOUT=A,OUTPUT=* .OUTRP2
//SYSPR3 DD SYSOUT=A
//LK3 EXEC PGM=IEWL,REGION=512K
//SYSPR4 DD SYSOUT=A,OUTPUT=* .LK1.OUTPUTRP1
```

and you want the system to process the data set for DD statement SYSPR4 according to the processing options specified on an OUTPUT JCL statement different from OUTRP1, you would specify:

```
//JOB1 JOB
//ADD2 OUTPUT DEFAULT=YES,COPIES=2,DEST=POK
//STEPS EXEC LINKS1
//LK3.SYSPR4 DD OUTPUT=
```

OUTPUT JCL statement ADD2 is now a job-level OUTPUT JCL statement and the DEFAULT parameter applies to any DD statement with the SYSOUT parameter coded for which there is no step-level OUTPUT JCL statement with DEFAULT = YES coded. Then, when you nullify the OUTPUT parameter on DD statement SYSPR4, SYSPR4 is processed according to ADD2. However, because the DEFAULT = YES parameter applies to any DD

statement with the SYSOUT parameter, the system uses OUTPUT JCL statement ADD2 when processing DD statement SYSPR3 and SYSPR4. To avoid using the DEFAULT = YES specification you could code:

```
//JOB1          JOB
//ADD2          OUTPUT COPIES=2,DEST=POK
//STEP1        EXEC   LINKS1
//LK3.SYSPR4   DD     OUTPUT=* .ADD2
```

You override the OUTPUT parameter specification on DD statement SYSPR4 with a reference to a job-level OUTPUT JCL statement. Because there is no DEFAULT = YES specification, OUTPUT JCL statement ADD2 does not apply to any other output data set unless there is an explicit reference made to ADD2.

For a discussion of step-level and job-level OUTPUT JCL statements, see “Processing System Output Data Sets Using the OUTPUT JCL Statement” on page 7-44.

You can use symbolic parameters on OUTPUT JCL statements that you are adding to a procedure. The use of symbolic parameters on an OUTPUT JCL statement is the same as for a DD statement. See “Symbolic Parameters” on page 2-15. However, if you are adding an OUTPUT JCL statement to the last step of a procedure, do **not** use symbolic parameters that are not used elsewhere in the procedure.

## Identifying Procedure Statements on an Output Listing

You can request that cataloged and in-stream procedure statements be included on the output listing by coding 1 as the first subparameter in the MSGLEVEL parameter on the JOB statement. (For a description of the MSGLEVEL parameter, see “Requesting Listings of JCL Statements and System Messages.”)

Procedure statements are identified on the output listing as illustrated in Figure 9-1 and Figure 9-2 on page 9-14. The output listing will also show the symbolic parameters and the values assigned to them.

Columns 1,2,3	
XX	cataloged procedure statement you did not override
X/	cataloged procedure statement you did override
XX*	cataloged procedure statement, other than a comment statement, that the system considers to contain only comments
***	comment statement, JES2, and JES3 statements
Note: The X/ identifier applies only to DD statements.	

**Figure 9-1. Identification of Cataloged Procedure Statements on the Output Listing**

Columns 1,2,3	
++	in-stream procedure statement you did not override
+/	in-stream procedure statement you did override
++*	in-stream procedure statement, other than a comment statement, that the system considers to contain only comments
***	comment statement, JES2, and JES3 statements
Note: The +/ identifier applies only to DD statements.	

**Figure 9-2. Identification of In-stream Procedure Statements on the Output Listing**

## Part 4. Reference to Job Control Statements and Parameters

This part details the coding of each JCL, JES2, and JES3 control statement. The chapters are:

- Chapter 10. Coding the JOB Statement
- Chapter 11. Coding the EXEC Statement
- Chapter 12. Coding the DD Statement
- Chapter 13. Coding Special DD Statements
- Chapter 14. Coding the OUTPUT JCL Statement
- Chapter 15. Coding Special JCL Statements
- Chapter 16. Coding JES2 Control Statements
- Chapter 17. Coding JES3 Control Statements

In chapters 10, 11, 12, and 14, which each cover only one statement, the parameters are listed alphabetically. In chapters 13, 15, 16, and 17, the statements are listed alphabetically and, for each statement, the parameters are listed alphabetically.

For each statement and parameter, this part gives the following information, as needed:

- **Parameter type:** positional or keyword, required or optional.
- **Purpose** of the parameter.
- **References** to related information in this book or other IBM publications.
- **Syntax** and coding rules.
- **Parameter or subparameter definitions:** how to code each parameter or subparameter.
- **Defaults** if you do not code a statement, parameter, or subparameter.
- **Overrides:** statements that this statement overrides or is overridden by or parameters that this parameter overrides or is overridden by.
- **Relationship to other parameters,** including other parameters or subparameters that must not be coded with this one.

## Reference

- **Relationship to other control statements.**
- **On EXEC statement that calls a procedure, for EXEC statement parameters.**
- **Location in the JCL.**
- **Other information required to code the statement or parameter.**
- **Examples.**



## Chapter 10. Coding the JOB Statement

**Purpose:** Use the JOB statement to mark the beginning of a job and to tell the system how to process the job. Also, when jobs are stacked in the input stream, the JOB statement marks the end of the preceding job.

The parameters you can specify for job processing are arranged alphabetically in the following pages.

**References:** For more information on coding JOB-related parameters, see Chapter 3, “Guide to Job Control” on page 3-1 and Chapter 5, “Guide to Job and Step Control” on page 5-1. For information about the JES initialization parameters that provide installation defaults, see *SPL: JES2 Initialization and Tuning* and *SPL: JES3 Initialization and Tuning*.

### Syntax:

```
//jobname JOB positional-parameters[,keyword-parameter]... comments
```

The JOB statement consists of the characters // in columns 1 and 2 and four fields: name, operation (JOB), parameter, and comments.

A JOB statement is required for each job.

### Name Field

Code a jobname on every JOB statement, as follows:

- Each jobname must be unique.
- The jobname must begin in column 3.
- The jobname is 1 through 8 alphanumeric or national characters.
- The first character must be alphabetic or national.
- The jobname must be followed by at least one blank.

### Parameter Field

A JOB statement has two kinds of parameters: positional and keyword. All parameters are optional unless your installation requires the accounting information parameter and the programmer’s name parameter.

# JOB

**Positional Parameters:** A JOB statement can contain two positional parameters. They must precede all keyword parameters. You must code the two positional parameters in the following order:

1. accounting information
2. programmer's name

**Keyword Parameters:** A JOB statement can contain the following keyword parameters. You can code any of the keyword parameters in any order in the parameter field after the positional parameters.

ADDRSPC  
CLASS  
COND  
GROUP  
MSGCLASS  
MSGLEVEL  
NOTIFY  
PASSWORD  
PERFORM  
PRTY  
RD  
REGION  
RESTART  
TIME  
TYPRUN  
USER

Do not use JOB statement parameter keywords as symbolic parameters, names, or labels.

## Comments Field

The comments field follows the parameter field after at least one intervening blank. If you do not code any parameters on a JOB statement, do not code any comments.

## Location in the JCL

A JOB statement must be the first statement in each job. JOB statements never appear in cataloged or in-stream procedures.

## Examples of JOB Statements

```
//ALPHA JOB 843,LINLEE,CLASS=F,MSGCLASS=A,MSGLEVEL=(1,1)
//LOS JOB ,'J M BUSKIRK',TIME=(4,30),MSGCLASS=H,MSGLEVEL=(2,0)
//MART JOB 1863,RESTART=STEP4 THIS IS THE THIRD JOB STATEMENT.
//TRY8 JOB
//RACF1 JOB 'D83,123',USER=RAC01,GROUP=A27,PASSWORD=XYX
```

## Accounting Information Parameter

**Parameter Type:** Positional, required (according to installation procedures)

**Purpose:** Use the accounting information parameter to enter an account number and any other accounting information that your installation requires.

**References:** For more information on the accounting information parameter, see “Job Accounting Information Parameter” on page 3-2, and on how to add accounting routines, see *SPL: System Management Facilities*.

If you are to provide accounting information for an individual step within a job, code an ACCT parameter on the EXEC statement for that step. For information on coding the EXEC statement ACCT parameter, see “ACCT Parameter” on page 11-4.

**Syntax:**

```
( [account-number] [, accounting-information] ... )
```

**Location:** Code the accounting information parameter first in the parameter field.

**Omission:** If you omit the accounting information parameter but you are coding a programmer’s name parameter, code a comma to indicate the omitted parameter. If you omit both positional parameters, do not code any commas before the first keyword parameter.

**Length:** The entire accounting information parameter must not exceed 142 characters:

- Including any commas, which are considered part of the information.
- Excluding any enclosing parentheses, which are not considered part of the information.

**Multiple Subparameters:** When the accounting information parameter consists of more than one subparameter, separate the subparameters by commas and enclose the parameter in parentheses or apostrophes. For example, (5438,GROUP6) or '5438,GROUP6'. If you use apostrophes, all information inside the apostrophes is considered one field.

**Special Characters:** When a subparameter contains special characters, other than hyphens, enclose it in apostrophes and the entire parameter in parentheses or enclose all of the parameter in apostrophes. For example, (12A75,'DEPT/D58',706) or '12A75,DEPT/D58,706'.

Code each apostrophe or ampersand that is part of the accounting information as two consecutive apostrophes or ampersands. For example, code DEPT'D58 as (12A75,'DEPT''D58',706) or '12A75,DEPT''D58,706'. Code 34&251 as '34&&251'.

**Continuation onto Another Statement:** Enclose the accounting information parameter in parentheses. End each statement with a comma after a complete parameter. For example:

```
//JOB1 JOB (12A75,'DEPT/D58',
//      706)
```

# JOB: Accounting Information

## Subparameter Definition

### **account-number**

Specifies an accounting number, as defined by the installation.

### **accounting-information**

Specifies more information, as defined by the installation. For example, your department and room numbers.

## JES2 Accounting Information Format

JES2 assumes that the JOB accounting information parameter could, alternatively, appear on the JES2 /\*JOBPARM statement. If you code the accounting information parameter in the following format, JES2 can interpret and use it.

**References:** For a discussion of the JES2 scan of the accounting information parameter, see *SPL: JES2 Initialization and Tuning*.

### **Syntax:**

(pano,room,time,lines,cards,forms,copies,log,linect)
--

Code a comma in place of each omitted subparameter when other subparameters follow.
---

## **Subparameter Definition**

### **pano**

Specifies the programmer's accounting number. pano is 1 to 4 alphanumeric characters.

### **room**

Specifies the programmer's room number. room is 1 to 4 alphanumeric characters.

### **time**

Specifies the estimated execution time in minutes. time is 1 to 4 decimal numbers. For example, code **30** for 30 minutes. If you omit time, JES2 uses an installation default specified at initialization.

### **lines**

Specifies the estimated line count in thousands of lines. lines is 1 to 4 decimal numbers. For example, code **5** for 5000 lines. If you omit lines, JES2 uses an installation default specified at initialization.

### **cards**

Specifies the estimated number of cards JES2 is to punch. cards is 1 to 4 decimal numbers. If you omit cards, JES2 uses an installation default specified at initialization.

### **forms**

Specifies the forms that JES2 is to use for printing output for the entire job. forms is 1 to 4 alphanumeric characters. For example, code **5** for 5-part forms. If you omit forms, JES2 uses an installation default specified at initialization.

## **copies**

Specifies the number of times JES2 is to print and/or punch this job's output. `copies` is 1 to 3 decimal numbers not exceeding an installation-specified limit. The maximum is 255. For example, code **2** for two copies. If you omit `copies`, JES2 assumes one copy.

The `copies` subparameter is ignored and only one copy is produced if the output class for the job log, as specified in the `JOB MSGCLASS` parameter, or the output class of any of the job's system output data sets is a held class.

## **log**

Specifies whether or not JES2 is to print the job log. Code **N** to request no job log. If you code any other character or omit this subparameter, JES2 prints the job log. If your installation specified `NOLOG` for this job's class during JES2 initialization, JES2 will not print a job log.

## **linect**

Specifies the number of lines JES2 is to print per page. `linect` is 1 to 3 decimal numbers. When you send a data set across a network, `linect` cannot exceed 254. When you print the data set locally, `linect` cannot exceed 255. If you omit `linect`, JES2 uses an installation default specified at initialization. If you code a zero, JES2 does not eject to a new page when the number of lines exceeds the installation default.

**Invalid Subparameters:** Your installation can initialize JES2 to do one of the following if the accounting information contains subparameters that are invalid to JES2:

- Ignore the invalid subparameters.
- Terminate the job. In this case, JES2 requires the first two subparameters: **pano** and **room**.

**Overrides:** A parameter on any of the following statements overrides an equivalent accounting information subparameter on the `JOB` statement:

- JES2 `/*JOBPARM` statement
- JES2 `/*OUTPUT` statement
- `OUTPUT JCL` statement
- `DD` statement

## **Examples of the Accounting Information Parameter**

```
//JOB43 JOB D548-8686
```

---

```
//YOURJOB JOB ,SUE,CLASS=A
```

In this statement, the accounting information parameter is omitted, but indicated by a comma.

---

## **JOB: Accounting Information**

```
//JOB44 JOB (D548-8686,'12/8/85',ERICKSON)
```

Because this statement contains an account-number plus additional accounting-information, parentheses are required.

---

```
//JOB45 JOB (CFH1,2G14,15,,,2)
```

This statement shows a JES2 accounting information parameter: programmer's accounting number, CFH1; room number, 2G14; estimated job time, 15 minutes; and copies, 2. Parentheses are required. Standard values are assumed for the other JES2 subparameters.

## ADDRSPC Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the ADDRSPC parameter to indicate to the system that the job requires virtual storage (pageable) or real storage (nonpageable).

**References:** For more information on the ADDRSPC parameter, see “Requesting Storage for Execution” on page 5-23 and “The REGION Parameter” on page 5-25.

**Syntax:**

```
ADDRSPC= { VIRT | REAL }
```

### Subparameter Definition

#### **VIRT**

Requests virtual storage. The system **can** page the job.

#### **REAL**

Requests real storage. The system **cannot** page the job and must place each step of the job in real storage.

### Defaults

If no ADDRSPC parameter is specified, the default is VIRT.

### Overrides

The JOB statement ADDRSPC parameter applies to all steps of the job and overrides any EXEC statement ADDRSPC parameters.

Code EXEC statement ADDRSPC parameters when each job step requires different types of storage. The system uses an EXEC statement ADDRSPC parameter only when no ADDRSPC parameter is on the JOB statement and only during the job step.

### Relationship to the JOB REGION Parameter

**When ADDRSPC = REAL:** Code a REGION parameter to specify how much real storage the job needs. If you omit the REGION parameter, the system uses an installation default specified at JES initialization.

**When ADDRSPC = VIRT or ADDRSPC is Omitted:** Do not code a REGION parameter. The system uses an installation default specified at JES initialization.

## **JOB: ADDRSPC**

### **Examples of the ADDRSPC Parameter**

```
//PEH JOB ,BAKER,ADDRSPC=VIRT
```

The ADDRSPC parameter requests virtual (pageable) storage. The space available to the job is the installation-specified default.

---

```
//DEB JOB ,ERIC,ADDRSPC=REAL,REGION=100K
```

The ADDRSPC parameter requests real (nonpageable) storage. The REGION parameter specifies 100K of storage for the job.



## CLASS Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the CLASS parameter to assign the job to a class. The class you should request depends on the characteristics of the job and your installation's rules for assigning classes.

**References:** For more information on the CLASS parameter, see "Assigning a Job to a Job Class in JES2" on page 5-19 or "Assigning a Job to a Job Class in JES3" on page 5-19. In JES3 systems, you can also code a CLASS parameter on a JES3 `//*MAIN` statement. For information on the `//*MAIN` statement, see "`//*MAIN` Statement" on page 17-23.

**Syntax:**

```
CLASS=jobclass
```

### Subparameter Definition

#### **jobclass**

Identifies the class for the job. The jobclass is one character, A through Z or 0 through 9, and must be a valid class specified at system initialization.

### Defaults

The default is based on the source of the job: The system makes the job's class the same as the installation-specified default class for the particular card reader, work station, or time-sharing user that submitted the job. The installation default is specified at JES initialization.

### Overrides

A JES3 `//*MAIN` statement CLASS parameter overrides a JOB statement CLASS parameter.

### Example of the CLASS Parameter

```
//SETUP JOB 1249,SMITH,CLASS=M
```

This statement assigns the job to class M.

# JOB: COND

## COND Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the COND parameter to specify the return code tests the system uses to determine whether a job will continue processing. Before each job step is executed, the system performs the COND parameter tests against the return codes from completed job steps. If none of these tests is satisfied, the system executes the job step; if any test is satisfied, the system bypasses all remaining job steps and terminates the job.

**References:** For more information on the COND parameter, see "Conditionally Executing Job Steps" on page 5-5.

**Syntax:**

```
COND=( (code,operator) [, (code,operator)] ... )
```

- One return code test is: (code,operator)
- You can omit the outer parentheses if you code only one return code test.
- Specify up to eight return code tests for a job.

## Subparameter Definition

**code**

Specifies a number that the system compares to the return code from each job step. code is a decimal number from 0 through 4095.

*Note:* Specifying a decimal number greater than 4095 could result in invalid return code testing or invalid return codes in messages.

**operator**

Specifies the type of comparison to be made to the return code. Operators and their meanings are:

Operator	Meaning
GT	Greater than
GE	Greater than or equal to
EQ	Equal to
NE	Not equal to
LT	Less than
LE	Less than or equal to

**Overrides**

If you code the COND parameter on the JOB statement and on one or more of the job's EXEC statements, and if a return code test on the JOB statement is satisfied, the job terminates. In this case, the system ignores any EXEC statement COND parameters.

If the tests on the JOB statement are not satisfied, the system then performs the return code tests on the EXEC statement. If a return code test is satisfied, the step is bypassed.

**Summary of COND Parameters**

See Figure 10-1 for a summary of how to code tests in the COND parameter to cause the job to be continued or terminated.

Test in COND Parameter	Return Code (RC) from Just Completed Step	
	Continue Job	Terminate Job
COND=(code,GT)	RC ≥ code	RC < code
COND=(code,GE)	RC > code	RC ≤ code
COND=(code,EQ)	RC = code	RC ≠ code
COND=(code,LT)	RC ≤ code	RC > code
COND=(code,LE)	RC < code	RC ≥ code
COND=(code,NE)	RC ≠ code	RC = code

Figure 10-1. Continuation or Termination of the Job Based on COND Parameter

**Examples of the COND Parameter**

```
//TYPE JOB (611,402),BOURNE,COND=(7,LT)
```

The COND parameter specifies that if 7 is less than the return code, the system terminates the job. Any return code less than or equal to 7 allows the job to continue.

```
//TEST JOB 501,BAXTER,COND=((20,GE),(30,LT))
```

The COND parameter specifies that if 20 is greater than or equal to the return code or if 30 is less than the return code, the system terminates the job. Any code of 21 through 30 allows the job to continue.

# JOB: GROUP

## GROUP Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the GROUP parameter to specify a RACF-defined group to which a RACF-defined user is to be connected. RACF places each RACF-defined user in a default group; the GROUP parameter is needed only to specify a group other than a user's default group.

The USER, the PASSWORD, and, optionally, the GROUP parameters are required on JOB statements only for the following:

- Batch jobs submitted through an input stream, such as a card reader, (1) if the job requires access to RACF-protected resources or (2) if the installation requires that all jobs have RACF identification.
- Jobs submitted by one TSO user for another user. In this case, the JOB statement must specify the other user's userid and password. The group id is optional.
- Jobs that execute at another network node that uses RACF protection.

Otherwise, the USER, PASSWORD, and GROUP parameters can be omitted from JOB statements. RACF uses the userid, password, and group id of the submitting TSO user or job.

**References:** For more information on the GROUP parameter, see "Controlling Access to RACF-Protected Data Sets" on page 6-2, and on RACF-protected facilities, see *Resource Access Control Facility (RACF) General Information Manual*.

**Syntax:**

GROUP=group-name
------------------

### Subparameter Definition

**group-name**

Identifies the group with which the system is to associate the user. group-name is 1 to 8 alphanumeric or national characters. The first character must be alphabetic or national.

### Defaults

If you do not code the GROUP parameter, but do code the USER and PASSWORD parameters, the system assigns a default group name it associates with the userid.

**Relationship to Other Parameters**

Code the **USER** and **PASSWORD** parameters on the **JOB** statement when you code the **GROUP** parameter.

**Example of the GROUP Parameter**

```
//TEST JOB 'D83,123456',GROUP=MYGROUP,USER=MYNAME,PASSWORD=ABC
```

This statement requests that the system connect RACF-defined user **MYNAME** to the group named **MYGROUP** for the duration of the job.

# JOB: MSGCLASS

## MSGCLASS Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the MSGCLASS parameter to assign the job log to an output class. The job log is a record of job-related information for the programmer. Depending on the JOB statement MSGLEVEL parameter, the job log can consist of:

- Only the JOB statement.
- All JCL statements.
- Cataloged procedure statements.
- JCL messages.
- JES and operator messages about the job.

**References:** For more information on the MSGCLASS parameter, see “Job Log” on page 3-14 and “MSGCLASS Parameter” on page 3-16.

**Syntax:**

```
MSGCLASS=class-name
```

### Subparameter Definition

**class-name**

Identifies the output class for the job log. The class-name is one character, A through Z or 0 through 9, and must be a valid output class specified at system initialization.

### Defaults

The default is based on the source of the job: The system places the job log in the same output class as the installation-specified default class for the particular card reader, work station, or time-sharing user that submitted the job. The installation default is specified at JES initialization.

### Significance of Output Classes

To print the job log and any output data sets on the same output listing, code one of the following:

- The same output class in the DD SYSOUT parameter as in the JOB MSGCLASS parameter.
- DD SYSOUT = \* to default to the JOB MSGCLASS output class.
- DD SYSOUT = (,) to default to one of the following:
  1. The CLASS parameter in an explicitly or implicitly referenced OUTPUT JCL statement. In this case, the OUTPUT JCL CLASS parameter should specify the same output class as the JOB MSGCLASS parameter.

- The JOB MSGCLASS output class, if no OUTPUT JCL statement is referenced or if the referenced OUTPUT JCL statement contains CLASS=\*

### Examples of the MSGCLASS Parameter

```
//EXMP1 JOB ,GEORGE,MSGCLASS=F
```

In this example, the JOB statement specifies output class F for the job log.

---

```
//EXMP2 JOB ,MENTLE,MSGLEVEL=(2,0)
```

This JOB statement does not specify an output class. In this case, the output class defaults to the installation default output class for the device from which the job was submitted.

---

```
//A1403 JOB ,BLACK,MSGCLASS=L
//STEP1 EXEC PGM=PRINT
//OUTDD1 DD SYSOUT=L
```

In this example, the JOB statement and sysout DD statement OUTDD1 both specify the same output class. Consequently, the job log and data set OUTDD1 are written on the same output listing.

---

```
//B209 JOB ,WHITE,MSGCLASS=M
//STEPA EXEC PGM=PRINT
//OUTDDX DD SYSOUT=*
```

In this example, the JOB statement specifies that the system route the job log to output class M. The system also routes sysout data set OUTDDX to class M because SYSOUT=\* is specified.

# JOB: MSGLEVEL

## MSGLEVEL Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the MSGLEVEL parameter to control the contents of the job log. You can request that the system print the following:

- Only the JOB statement.
- All JCL statements.
- Cataloged procedure statements for any procedure a job step calls, including the internal representation of procedure statement parameters after symbolic parameter substitution.
- JCL messages.
- JES and operator messages about the job: the allocation of devices and volumes, disposition of data sets, and termination of job steps and the job.

**References:** For more information on the MSGLEVEL parameter, see “MSGLEVEL Parameter” on page 3-14.

**Syntax:**

```
MSGLEVEL=( [statements] [ ,messages] )
```

You can omit the parentheses if you code only the first subparameter.

## Subparameter Definition

### statements

Indicates which job control statements the system is to print in the job log. statements is one of the following numbers:

- 0 The system prints only the JOB statement.
- 1 The system prints all JCL statements, the cataloged procedure statements, and the internal representation of procedure statement parameters after symbolic parameter substitution.
- 2 The system prints only JCL statements.

### messages

Indicates which messages the system is to print in the job log. messages is one of the following numbers:

- 0 The system prints only JCL messages. It prints JES and operator messages only if the job abnormally terminates.
- 1 The system prints JCL messages and all JES and operator messages.



## Defaults

If you do not code the MSGLEVEL parameter, JES uses an installation default specified at initialization.

## Examples of the MSGLEVEL Parameter

```
//EXMP3 JOB ,GEORGE,MSGLEVEL=(2,1)
```

In this example, the JOB statement requests that the system print only JCL statements, JCL messages, and JES and operator messages.

---

```
//EXMP4 JOB ,MENTLE,MSGLEVEL=0
```

In this example, the JOB statement requests that the system print only the JOB statement and that JES is to use the installation default for messages.

---

```
//EXMP5 JOB ,MIKE,MSGLEVEL=(,0)
```

In this example, the JOB statement requests that JES use the installation default for printing JCL statements and the system is not to print JES and operator messages unless the job abnormally terminates.

## JOB: NOTIFY

### NOTIFY Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the NOTIFY parameter to request that the system send a message to your TSO userid or another TSO userid when this background job completes processing.

**References:** For more information on the NOTIFY parameter, see "TSO" on page 3-18.

**Syntax:**

```
NOTIFY=userid
```

### Subparameter Definition

**userid**

Identifies the user that the system is to notify. The userid is 1 to 7 alphanumeric characters and must be a valid TSO userid.

### Relationship to JES2 /\*JOBPARM SYSAFF Parameter

If you submit a job with a JOB statement NOTIFY parameter or a JES2 /\*NOTIFY statement, then the mode of the job (independent or not) must match that of the system at which the job is submitted. That is, for TSO-submitted jobs, you cannot change the system affinity using the JES2 /\*JOBPARM SYSAFF parameter.

### Receiving Notification of Job Completion

**In a JES2 System:** If you are logged on to the member of the JES2 multi-access spool from which you submitted the job, the system immediately notifies you when the job completes. If you are not logged on, the system saves the message until you log on to the member from which you originally submitted the job.

**In a JES3 System:** If you are logged on, the system immediately notifies you when the job completes. If you are not logged on, the system saves the message until you log on to the system from which you originally submitted the job.

To receive notification that a job you submitted through batch processing has completed, supply a main-name on the ACMAIN parameter of the JES3 /\*MAIN statement in addition to the JOB statement NOTIFY parameter. The ACMAIN parameter should specify the processor on which your TSO system is running.

**Example of the NOTIFY Parameter**

```
//SIGN JOB ,JEEVES,NOTIFY=POK1
```

When the job SIGN completes processing, the system sends a message to userid POK1.

JOB

## JOB: PASSWORD

### PASSWORD Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the PASSWORD parameter to identify a current RACF password or specify a new RACF password. You can specify a new password at any time and must specify a new password when your current one expires.

**Note:** If your installation uses the early authorization verification option and has an exit routine to implement this option, a new password specified in the PASSWORD parameter takes effect when the job is read in, even if the job fails because of JCL errors or even if the job is executed later. When changing the password, other jobs that use the new or old password may fail, depending on when their passwords are verified.

The USER, the PASSWORD, and, optionally, the GROUP parameters are required on JOB statements only for the following:

- Batch jobs submitted through an input stream, such as a card reader, (1) if the job requires access to RACF-protected resources or (2) if the installation requires that all jobs have RACF identification.
- Jobs submitted by one TSO user for another user. In this case, the JOB statement must specify the other user's userid and password. The group id is optional.
- Jobs that execute at another network node that uses RACF protection.

Otherwise, the USER, PASSWORD, and GROUP parameters can be omitted from JOB statements. RACF uses the userid, password, and group id of the submitting TSO user or job.

**References:** For more information on the PASSWORD parameter, see "Controlling Access to RACF-Protected Data Sets" on page 6-2. For more information on using RACF-protected facilities, see *Resource Access Control Facility (RACF) General Information Manual*.

**Syntax:**

```
PASSWORD=(password[,new-password])
```

- You can omit the parentheses if you code only the first subparameter.
- The PASSWORD parameter must be on the first statement if the JOB statement is continued.

## Subparameter Definition

### password

Specifies the user's current RACF password. The password is 1 to 8 alphanumeric or national characters.

### new-password

Specifies the user's new RACF password. The new-password is 1 to 8 alphanumeric or national characters. The installation's security administrator can impose additional restrictions on passwords; follow your installation's rules.

JOB

## Relationship to Other Parameters

You must code a PASSWORD parameter when you code a USER or GROUP parameter on a JOB statement.

You must code a USER parameter when you code a PASSWORD parameter.

## Examples of the PASSWORD Parameter

```
//TEST1 JOB 'D83,123456',PASSWORD=ABCDE,USER=MYNAME
```

This JOB statement identifies ABCDE as the current password for the RACF user.

---

```
//TEST2 JOB 'D83,123456',PASSWORD=(BCH,A12),USER=RAC1,GROUP=GRP1
```

This JOB statement requests that the system change the RACF password from BCH to A12.

# JOB: PERFORM

## PERFORM Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the PERFORM parameter to specify the performance group for the job. The installation-defined performance groups determine the rate at which associated jobs have access to the processor, storage, and channels.

**References:** For information on performance groups, see “Establishing job processing balance in JES3” on page 5-19, “Performance of Jobs and Job Steps in JES2” on page 5-22, or “Performance of Jobs and Job Steps in JES3” on page 5-23.

**Syntax:**

```
PERFORM=n
```

### Subparameter Definition

**n**

Indicates the performance group. The n is a number from 1 through 999 and must identify a performance group that has been defined by your installation. The specified performance group should be appropriate for your job type according to your installation's rules.

### Defaults

If no PERFORM parameter is specified or if the specified PERFORM number fails validity checks, the system uses an installation default specified at initialization. If the installation did not specify a default, the system uses a built-in default:

- 1 for non-TSO jobs
- 2 for TSO sessions

See *SPL: Initialization and Tuning Guide* for details.

### Overrides

A JOB statement PERFORM parameter applies to all steps of the job and overrides any EXEC statement PERFORM parameters.

Code EXEC statement PERFORM parameters when each job step executes in a different performance group. The system uses an EXEC statement PERFORM parameter only when no PERFORM parameter is on the JOB statement and only during the job step.

**Example of the PERFORM Parameter**

```
//STEP1 JOB ,MARLA,CLASS=D,PERFORM=25
```

In this example, CLASS=D determines the class in which the system will execute the job. Once in the system, the job will run in performance group 25. The installation must have defined the significance of this performance group.

JOB

## JOB: Programmer's Name

### Programmer's Name Parameter

**Parameter Type:** Positional, required (according to installation procedures)

**Purpose:** Use the programmer's name parameter to identify the person or group responsible for a job.

**References:** For more information on the programmer's name parameter, see "Programmer Information: The programmer-name parameter" on page 3-5.

**Syntax:**

```
programmer's-name
```

**Location:** Place the programmer's name parameter immediately after the accounting information parameter and before all keyword parameters.

**Omission:** Do not code a comma to indicate the absence of the programmer's name parameter. For example:

```
//YOURJOB JOB 'D58/706',MSGCLASS=A
```

### Parameter Definition

**programmer's-name**

Identifies the job's owner. The name must not exceed 20 characters, including all special characters.

**Special Characters:** Enclose the programmer's name in apostrophes when:

- The name contains special characters, other than hyphens, leading periods, or embedded periods. For example:

```
//YOURJOB JOB 'BUILD/PAUL'  
//YOURJOB JOB 'MAE BIRDSALL'
```

- The last character of the name is a period. For example:

```
//YOURJOB JOB 'TIIU.'
```

- Code each apostrophe that is part of the name as two consecutive apostrophes. For example, code O'DONNELL as 'O'DONNELL'.



**Examples of the Programmer's Name Parameter**

```
//APP JOB ,G.M.HILL
```

This JOB statement specifies a programmer's name with no accounting information. The leading comma may be optional; check with your installation.

---

```
//DELTA JOB 'T.O''NEILL'
```

The programmer's name contains special characters. The installation requires no accounting information. The imbedded apostrophe is coded as two consecutive apostrophes; the entire name must be enclosed in apostrophes.

---

```
//#308 JOB (846349,GROUP12),MATTHEW
```

This JOB statement specifies an account number, additional accounting information, and a programmer's name.

---

```
//JOBA JOB 'NICOLLE.'
```

Because this programmer's name ends with a period, it is enclosed in apostrophes.

## JOB: PRTY

### PRTY Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the PRTY parameter to assign:

- **In a JES2 system**, the queue selection priority for your job and all of its output, except the JES2 hard-copy log.

*Note:* Depending on the JES2 initialization options in use at your installation, JES2 may ignore the PRTY parameter.

- **In a JES3 system**, the job's initiation or selection priority within its job class.

A job with a higher priority is selected for execution sooner.

**References:** For more information about priority, see "Assigning a Priority to a Job for JES2" on page 5-20 and *SPL: JES2 Initialization and Tuning* or "Assigning a Priority to a Job in JES3" on page 5-20.

**Syntax:**

```
PRTY=priority
```

### Subparameter Definition

#### priority

Requests a priority for the job. The priority is a number from 0 through 15 for JES2 and from 0 through 14 for JES3. The highest priority is 15 or 14.

Follow your installation's rules in coding a priority.

### Defaults in a JES3 System

If no PRTY parameter is specified, JES3 use an installation default specified at initialization. If the PRTY specified is invalid, JES3 issues an error message.

### Relationship to Other Control Statements in a JES2 System

Instead of coding the PRTY parameter on a JOB statement, JES2 users can code the JES2 /\*PRIORITY control statement. For information on coding the /\*PRIORITY statement, see "/\*PRIORITY Statement" on page 16-22. If a /\*PRIORITY statement is not present or if JES2 ignores the /\*PRIORITY statement, the system derives the priority from the following, in override order:

1. The PRTY parameter on the JOB statement.
2. The accounting information on a /\*JOBPARM statement.
3. The accounting information on the JOB statement.
4. An installation default specified at JES2 initialization.

**Example of the PRTY Parameter**

```
//JOBA JOB 1, 'JIM WEBSTER', PRTY=12
```

This job has a priority of 12.

JOB

## RD Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the RD (restart definition) parameter to:

- Request that the operator perform automatic step restart if the job fails.
- Suppress, partially or totally, the action of the assembler language CHKPT macro instruction or the DD statement CHKPT parameter.

The system can perform automatic restart only if the job has a job journal. A job journal is a sequential data set that contains job-related control blocks needed for restart. For JES2, specify a job journal by one of the following:

- An installation option during JES2 initialization.
- RD = R or RD = RNC on either the JOB statement or any one EXEC statement in the job.
- RESTART parameter on the JOB statement.

For JES3, specify a job journal by one of the following:

- An installation option during JES3 initialization.
- RD = R or RD = RNC on either the JOB statement or any one EXEC statement in the job.
- JOURNAL = YES on a JES3 `//*MAIN` statement in the job.

**References:** For more information on the RD parameter, see “The RD Parameter on the JOB Statement” on-page 5-27, and on restarting jobs, see:

- “Restarting a Job at a Step or Checkpoint” on page 5-26.
- “RESTART Parameter” on page 10-33.
- “RD Parameter” on page 11-26.

For JES2 systems, see the RESTART parameter on the `/*JOBPARM` control statement in “`/*JOBPARM` Statement” on page 16-4.

For JES3 systems, see the FAILURE parameter on the `//*MAIN` control statement in “`//*MAIN` Statement” on page 17-23.

For detailed information on deferred checkpoint restart, see *Checkpoint/Restart*.

**Syntax:**

RD = { R   RNC   NC   NR }
----------------------------

## Subparameter Definition

### R (Restart)

Indicates that the operator is to perform automatic step restart if the job fails.

RD = R does not suppress checkpoint restarts:

- If the processing program executed in a job step does not include a CHKPT macro instruction, RD = R allows the system to restart execution at the beginning of the abnormally terminated step.
- If the program includes a CHKPT macro instruction, RD = R allows the system to restart execution at the beginning of the step, if the step abnormally terminates before the CHKPT macro instruction is executed.
- If the step abnormally terminates after the CHKPT macro instruction is executed, only checkpoint restart can occur. If you cancel the affects of the CHKPT macro instruction before the system performs a checkpoint restart, the request for automatic step restart is again in effect.

### RNC (Restart and No Checkpoint)

Indicates that the operator is to perform automatic step restart if the job fails.

RD = RNC suppresses automatic and deferred checkpoint restarts. It suppresses:

- Any CHKPT macro instruction in the processing program: That is, the operator is not to perform an automatic checkpoint restart, and the system is not to perform a deferred checkpoint restart if the job is resubmitted.
- The DD statement CHKPT parameter.
- The checkpoint at end-of-volume (EOV) facility; see “SYSCKEOV DD Statement” on page 13-19.

### NC (No Checkpoint)

Indicates that the operator is **not** to perform automatic step restart if the job fails.

RD = NC suppresses automatic and deferred checkpoint restarts. It suppresses:

- Any CHKPT macro instruction in the processing program.
- The DD statement CHKPT parameter.
- The checkpoint at EOV facility.

## JOB: RD

### NR (No Automatic Restart)

Indicates that the operator is **not** to perform automatic step restart if the job fails.

RD=NR suppresses automatic checkpoint restart but permits deferred checkpoint restarts. It permits:

- A CHKPT macro instruction to establish a checkpoint.
- The job to be resubmitted for restart at the checkpoint. On the JOB statement when resubmitting the job, specify the checkpoint in the RESTART parameter.

If the system fails, RD=NR does not prevent the job from restarting.

### Defaults

If no RD parameter is specified, the terminated job step is eligible for automatic checkpoint/restart, if its program requested checkpoints with a CHKPT macro instruction.

### Overrides

A JOB statement RD parameter applies to all steps of the job and overrides any EXEC statement RD parameters.

Code EXEC statement RD parameters when each job step requires different restart types. The system uses an EXEC statement RD parameter only when no RD parameter is on the JOB statement and only during the job step.

### Relationship to Other Control Statements

RD=NC or RD=RNC suppresses the action of the DD statement CHKPT parameter.

### Examples of the RD Parameter

```
//JILL JOB 333,TOM,RD=R
```

RD=R specifies that the operator is to perform automatic step restart if the job fails.

```
//TRY56 JOB 333,DICK,RD=RNC
```

RD=RNC specifies that, if the job fails, the operator is to perform automatic step restart beginning with the step that abnormally terminates. RD=RNC suppresses automatic and deferred checkpoint restarts.

```
//PASS JOB (721,994),HARRY,RD=NR
```

RD=NR specifies that the operator is not to perform automatic step restart or automatic checkpoint restart. However, a CHKPT macro instruction can establish checkpoints to be used later for a deferred restart.

## REGION Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the REGION parameter to specify the amount of space that the job requires.

The specified or default region size sets an upper boundary to limit region size for variable-length GETMAINS. The system uses the upper boundary for variable-length GETMAINS as long as the region still has available at least the minimum amount of storage requested.

In addition, the IBM- or installation-supplied routine IEALIMIT uses the region size to establish a second limiting value. The system uses this second value for:

- Fixed-length GETMAINS.
- Variable-length GETMAINS when the space remaining in the region is less than the minimum requested.

If the minimum requested length for variable-length GETMAINS exceeds this second value, the job or job step abnormally terminates.

REGION=0K gives the job all the storage available in the private area, that is, from the top of the system region to the bottom of the common service area (CSA). The resulting size of the region is unpredictable.

**References:** For more information on the REGION parameter, see “The REGION Parameter” on page 5-25. Also, see “ADDRSPC Parameter” on page 10-7. For more information on the region size and the routine, see *OS/VS2 Supervisor Services and Macro Instructions*. and *SPL: Supervisor*.

**Syntax:**

```
REGION=valueK
```

### Subparameter Definition

**valueK**

Specifies the required storage in thousands (1024) of bytes. The value is 1 to 5 decimal numbers. Code an even number. For example, REGION=66K. If you code an odd number, the system treats it as the next highest even number.

### Defaults

If no REGION parameter is specified, the system uses an installation default specified at JES initialization.

## JOB: REGION

### Overrides

A JOB statement REGION parameter applies to all steps of the job and overrides any EXEC statement REGION parameters.

Code EXEC statement REGION parameters when each job step requires a different region size. The system uses an EXEC statement REGION parameter only when no REGION parameter is on the JOB statement and only during the job step.

### Relationship to the JOB ADDRSPC Parameter

**When ADDRSPC=REAL:** Code a REGION parameter to specify how much real storage the job needs. If you omit the REGION parameter, the system uses the default.

**When ADDRSPC=VIRT or ADDRSPC is Omitted:** Do not code a REGION parameter. The system uses the default.

### Examples of the REGION Parameter

```
//ACCT1 JOB A23,SMITH,REGION=100K,ADDRSPC=REAL
```

This JOB statement indicates that the job requires 100K of real storage.

---

```
//ACCT4 JOB 175,FRED,REGION=250K
```

This JOB statement indicates that the job requires 250K of virtual storage. When the ADDRSPC parameter is omitted, the system defaults to ADDRSPC=VIRT.



## RESTART Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the RESTART parameter to restart a job. You can specify that the system perform either of two restarts:

- **Deferred step restart**, which is a restart at the beginning of a job step.
- **Deferred checkpoint restart**, which is a restart from a checkpoint taken during step execution by a CHKPT macro instruction.

**References:** For more information on the RESTART parameter, see “The RESTART Parameter on the JOB Statement” on page 5-28, and on restarting jobs, see:

- “Restarting a Job at a Step or Checkpoint” on page 5-26.
- “The RD Parameter on the JOB Statement” on page 5-27.
- “The RD Parameter on the EXEC Statement” on page 5-28.
- “RD Parameter” on page 10-28.
- “RD Parameter” on page 11-26.

For JES2 systems, see the RESTART parameter on the /\*JOBPARM control statement in “/\*JOBPARM Statement” on page 16-4.

For JES3 systems, see the FAILURE parameter on the /\*MAIN control statement in “/\*MAIN Statement” on page 17-23.

For detailed information on the deferred checkpoint restart, see *Checkpoint/Restart*.

See “Restrictions on Use of SYSCHK DD Statement and DD Statement RESTART Parameter” on page v.

### Syntax:

```
RESTART=( { * | stepname | stepname.procstepname } [ , checkid ] )
```

You can omit the parentheses if you code only the first subparameter.

# JOB: RESTART

## Subparameter Definition

\*

Indicates that the system is to restart execution (1) at the beginning of or within the first job step or (2), if the first job step calls a cataloged or in-stream procedure, at the beginning of or within the first procedure step.

### **stepname**

Indicates that the system is to restart execution at the beginning of or within a job step. Stepname identifies the EXEC statement of the job step.

### **stepname.procstepname**

Indicates that the system is to restart execution at the beginning of or within a step of a cataloged procedure. Stepname identifies the EXEC statement of the job step that calls the procedure; procstepname identifies the EXEC statement of the procedure step.

### **checkid**

Specifies the name of the checkpoint at which the system is to restart execution. This checkpoint must be in the job step specified in the first subparameter.

Omit checkid to request restart at the beginning of the specified job step.

When the name contains special characters, enclose it in apostrophes. Code each apostrophe that is part of the name as two consecutive apostrophes. For example, code CHPT'1 as 'CHPT''1'.

## Relationship to Other Control Statements

When the system is to restart execution in a job step, place a SYSCHK DD statement immediately following the JOB statement. The SYSCHK DD statement defines the data set on which the system entered the checkpoint for the step being restarted.

When preparing for a deferred checkpoint, code the DISP abnormal termination disposition subparameter in the step's DD statements as follows:

- KEEP, to keep all data sets that the restart step is to use.
- CATLG, to catalog all data sets that you are passing from steps preceding the restart step to steps following the restart step.

For more information, see "DISP Parameter" on page 12-57.

## Cautions When Coding the RESTART Parameter

Before resubmitting a job:

- Check all backward references to steps before the restart step. Eliminate all backward references in EXEC statement PGM parameters and DD statement VOLUME = REF parameters.

- Review all EXEC statement COND parameters. If any of the COND parameters reference a step before the restart step, be aware that the system ignores the return code tests for those steps.

## Generation Data Sets in Restarted Jobs

In the restart step or following steps, do not use the original relative generation numbers to refer to generation data sets that were created and cataloged before the restart step. Instead, refer to a generation data set by its present relative generation number.

For example, if the last generation data set created and cataloged was assigned a generation number of +2, refer to it as 0 in the restart step and following steps. If generation data set +1 was also created and cataloged, refer to it as -1. For more information on using generation data sets, see "Creating and Retrieving Generation Data Sets" on page 8-25.

If generation data sets created in the restart step were kept instead of cataloged, that is, DISP=(NEW,CATLG,KEEP) was coded, then refer to them by the same relative generation numbers used to create them.

## Examples of the RESTART Parameter

```
//LINES JOB '1/17/85',RESTART=COUNT
```

This JOB statement indicates that the system is to restart execution at the beginning of the job step named COUNT.

---

```
//@LOC5 JOB '4/11/86',RESTART=(PROCESS,CHKPT3)
//SYSCHK DD DSNAME=CHK,UNIT=3330,DISP=OLD
```

The JOB statement indicates that the system is to restart execution at checkpoint CHKPT3 in job step PROCESS. The SYSCHK DD statement must follow the JOB statement; it defines the data set on which the system wrote checkpoint CHKPT3.

---

```
//WORK JOB ,PORTER,RESTART=(*,CKPT2)
//SYSCHK DD DSNAME=CHKPT,UNIT=3330,DISP=OLD
```

The JOB statement indicates that the system is to restart execution at checkpoint CKPT2 in the first job step. The SYSCHK DD statement defines the data set on which the system wrote checkpoint CKPT2.

---

```
//CLIP5 JOB ,JONES,RESTART=(PAY.WEEKLY,CHECK8)
//SYSCHK DD DSNAME=CHKPT,UNIT=3350,DISP=OLD
```

The JOB statement indicates that the system is to restart execution at checkpoint CHECK8 in procedure step WEEKLY. PAY is the name field on the EXEC statement that calls the cataloged procedure that contains procedure step WEEKLY. The SYSCHK DD statement defines the data set on which the system wrote checkpoint CHECK8.

## JOB: TIME

### TIME Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the TIME parameter to specify the maximum length of time that a job is to use the processor and to find out through messages how much processor time the job used.

The system terminates a job that exceeds the specified time limit unless a user exit routine extends the time.

**References:** For more information on the TIME parameter, see "Limiting Job and Job Step Execution Time" on page 5-16.

**Syntax:**

```
TIME={1440 | ([minutes] [,seconds])}
```

You can omit the parentheses if you code only 1440 or the processor time in minutes.

If you omit the seconds, do not code a null subparameter. For example, TIME=(60,) is invalid.

### Subparameter Definition

**1440**

Indicates that the job can use the processor for an unlimited amount of time; 1440 literally means 24 hours. Code TIME=1440 for the following reasons:

- To obtain job accounting information.
- To specify that the system is to allow any of the job's steps to remain in a wait state for more than the installation-established time limit.

**minutes**

Specifies the maximum number of minutes the job can use the processor. The minutes must be a number from 1 through 1439.

Do not code TIME=0 on the JOB statement. The results are unpredictable.

**seconds**

Specifies the maximum number of seconds that the job can use the processor, in addition to any minutes that are specified. The seconds must be a number from 1 through 59.

## Overrides

For a JOB statement TIME parameter other than TIME=1440, the system sets the time limit for each step to:

- The step time limit specified on the EXEC statement TIME parameter.
- If no EXEC TIME parameter was specified, (1) the default time limit or (2) the job time remaining after execution of previous steps, whichever is smaller.

## Defaults

If no TIME parameter is specified, JES uses an installation default specified at initialization.

## Time Handling

**How the System Converts the Time Value:** The job time limit or the time remaining after execution of previous steps in a job is converted by the system to seconds and then rounded to the nearest unit, where 1 unit=1.048576 seconds. Thus, a step can begin execution with up to one-half unit more or one-half unit less time than expected. If the time remaining for the job is less than one-half unit, a step will begin execution with zero time, resulting in an abnormal termination.

**Time Checking:** Because the system checks the processor time-used field about every 10.5 seconds, the actual time that a job uses the processor can exceed the specified TIME value by up to 10.5 seconds. For example, the system checks the job's time-used field and finds 0.5 seconds remaining. Because the system does not again check the job's time-used field for about 10.5 seconds, the job can execute for an additional 10.5 seconds and thus exceed the coded TIME value by 10 seconds.

## Examples of the TIME Parameter

```
//STD1 JOB ACCT271,TIME=(12,10)
```

This statement specifies that the maximum amount of time the job can use the processor is 12 minutes, 10 seconds.

---

```
//TYPE41 JOB ,GORDON,TIME=(,30)
```

This statement specifies that the maximum amount of time the job can use the processor is 30 seconds.

---

```
//FORMS JOB ,MORRILL,TIME=5
```

This statement specifies that the maximum amount of time the job can use the processor is 5 minutes.

---

```
//RAINCK JOB 374231,MORRISON,TIME=1440
```

This statement specifies an unlimited amount of time for job execution; the job can use the processor and remain in wait state for an unspecified period of time. The system will issue messages telling how much processor time the job used.

# JOB: TIME

## Examples of the TIME Parameter on JOB and EXEC Statements

```
//FIRST JOB      ,SMITH,TIME=2  
//STEP1 EXEC    PGM=READER,TIME=1
```

```
•  
•  
//STEP2 EXEC    PGM=WRITER,TIME=1
```

In this example, the job is allowed 2 minutes for execution and each step is allowed 1 minute. If either step continues executing beyond 1 minute, the entire job abnormally terminates beginning with that step.

---

```
//SECOND JOB    ,JONES,TIME=3  
//STEP1 EXEC    PGM=ADDER,TIME=2
```

```
•  
•  
//STEP2 EXEC    PGM=PRINT,TIME=2
```

In this example, the job is allowed 3 minutes for execution, and each step is allowed 2 minutes. If either step continues executing beyond 2 minutes, the entire job abnormally terminates beginning with that step. If STEP1 executes for 1.74 minutes and STEP2 tries to execute beyond 1.26 minutes, the job abnormally terminates because of the 3-minute limit specified on the JOB statement.

## TYPRUN Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the TYPRUN parameter to tell the system to:

- Place a job on hold until a special event occurs. When the event occurs, the operator, following your directions, must release the job from its hold to allow the system to select the job for processing.
- Scan a job's JCL for syntax errors.
- In a JES2 system, request that JES2 convert the input job stream directly to a system output data set and schedule it for output processing.

**References:** For more information on the TYPRUN parameter, see "Bypassing Job Initiation" on page 3-25.

**For JES2 systems,** see related information in "Delaying Initiation of Your Job in JES2" on page 3-23, "Copying JCL Input Without Execution in JES2" on page 3-25, and "/\*SETUP Statement" on page 16-28.

**For JES3 systems,** see related information in "Delaying Initiation of Other Jobs (JES3)" on page 3-24, "Testing JCL Without Execution (JES3)" on page 3-25, "JES3 SETUP Parameter" on page 3-26, "Deadline Scheduling for JES3" on page 3-27, and "Dependent Job Control for JES3: The Job Net" on page 3-27.

**Syntax:**

```
TYPRUN= { HOLD | JCLHOLD | SCAN | COPY }
```

### Subparameter Definition

#### **HOLD**

Requests that the system hold the job before execution until the operator releases it. The operator should release the job when a particular event occurs. If an error occurs during input service processing, JES does not hold the job.

#### **JCLHOLD (JES2 only)**

Requests that JES2 hold the job before completing JCL processing. JES2 holds the job until the operator releases it.

*Note:* JCLHOLD is supported only in JES2 systems.

# JOB: TYPRUN

## SCAN

Requests that the system scan this job's JCL for syntax errors, without executing the job or allocating devices. This parameter asks the system to check for:

- Invalid keywords.
- Invalid characters.
- Parentheses errors.
- Parameter value errors or excessive parameters in a JES3 system, but not in a JES2 system.
- Invalid syntax on JCL statements in cataloged procedures invoked by EXEC statements in the job.

The system does not check for misplaced statements.

## COPY (JES2 only)

Requests that JES2 convert the input job stream, as submitted, directly to a system output data set and schedule it for output processing. The class of this sysout data set is the same as the message class of the job and is controlled by the MSGCLASS parameter.

*Note:* COPY is supported only in JES2 systems. This feature is available in JES3 by using the JES3 `//*PROCESS` statement. See "`//*PROCESS` Statement" on page 17-44.

## Example of the TYPRUN Parameter

```
//UPDATE JOB      ,HUBBARD
//STEP1  EXEC    PGM=LIBUTIL
.
.
.
//LIST   JOB      ,HUBBARD, TYPRUN=HOLD
//STEPA  EXEC    PGM=LIBLIST
.
.
.
```

Jobs UPDATE and LIST are submitted for execution in the same input stream. UPDATE executes a program that adds and deletes members of a library; LIST executes a program that lists the members of that library. For an up-to-date listing of the library, LIST must execute after UPDATE. To force this execution order, code `TYPRUN=HOLD` on JOB statement LIST.

If a MONITOR JOBNAME command is executed from the input stream or by the operator, the system notifies the console operator when UPDATE completes. The operator can then release LIST, allowing the system to select LIST for execution.



## USER Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Code the USER parameter to identify to the system the person submitting the job. The userid is used by the Resource Access Control Facility (RACF), the system resources manager (SRM), and other system components.

The USER, the PASSWORD, and, optionally, the GROUP parameters are required on JOB statements only for the following:

- Batch jobs submitted through an input stream, such as a card reader, (1) if the job requires access to RACF-protected resources or (2) if the installation requires that all jobs have RACF identification.
- Jobs submitted by one TSO user for another user. In this case, the JOB statement must specify the other user's userid and password. The group id is optional.
- Jobs that execute at another network node that uses RACF protection.

Otherwise, the USER, PASSWORD, and GROUP parameters can be omitted from JOB statements. RACF uses the userid, password, and group id of the submitting TSO user or job.

**References:** For more information on the USER parameter, see "Controlling Access to RACF-Protected Data Sets" on page 6-2, and on RACF-protected facilities, see *Resource Access Control Facility (RACF) General Information Manual*.

**Syntax:**

```
USER=userid
```

### Subparameter Definition

**userid**

Identifies a user to the system. The userid consists of alphanumeric or national characters; the first character must be alphabetic or national. It is 1 through 8 characters or, for TSO users, 1 through 7 characters.

### Defaults

If neither the JOB statement nor the submitting TSO user supplies identification information, RACF assigns a default userid and group id, unless the job enters the system via a JES internal reader. In that case, the user and group identification of the submitting TSO user or job is used.

# JOB: USER

## Relationship to Other Parameters

Code the USER parameter when you code the GROUP or PASSWORD parameters on the JOB statement.

## Example of the USER Parameter

```
//TEST JOB 'D83,123456',USER=MYNAME,PASSWORD=ABCD
```

This statement identifies the user submitting this job as MYNAME.

## Chapter 11. Coding the EXEC Statement

**Purpose:** Use the EXEC (execute) statement to identify the program or cataloged or in-stream procedure that this job step is to execute and to tell the system how to process the job step. The EXEC statement marks the beginning of each step in a job or a procedure.

A job can have a maximum of 255 job steps. This maximum includes all steps in any procedures the EXEC statements call.

The parameters you can specify for step processing are arranged alphabetically in the following pages.

**References:** For more information on coding EXEC-related parameters, see Chapter 4, “Guide to Step Control” on page 4-1 and Chapter 5, “Guide to Job and Step Control” on page 5-1. For information about the JES initialization parameters that provide installation defaults, see *SPL: JES2 Initialization and Tuning* and *SPL: JES3 Initialization and Tuning*.

**Syntax:**

```
//[stepname] EXEC positional-parameter[,keyword-parameter]... comments
```

The EXEC statement consists of the characters // in columns 1 and 2 and four fields: name, operation (EXEC), parameter, and comments.

An EXEC statement is required for each job step.

### Name Field

A stepname is optional, but is needed for the following:

- Coding backward references to the step.
- Overriding parameters on an EXEC statement or DD statement in a cataloged or in-stream procedure step.
- Adding DD statements to a cataloged or in-stream procedure step.
- Performing a step or checkpoint restart at or in the step.

# EXEC

Code a stepname as follows:

- Each stepname must be unique within the job.
- The stepname must begin in column 3.
- The stepname is 1 through 8 alphanumeric or national characters.
- The first character must be alphabetic or national.
- The stepname must be followed by at least one blank.

## Parameter Field

An EXEC statement has two kinds of parameters: positional and keyword.

Do not use EXEC statement parameter keywords as symbolic parameters, names, or labels.

**Positional Parameters:** An EXEC statement must contain **one** of the following positional parameters. The positional parameter must precede all keyword parameters.

PGM  
PROC  
procedure name

**Keyword Parameters:** An EXEC statement can contain the following keyword parameters. You can code any of the keyword parameters in any order in the parameter field after the positional parameter.

ACCT  
ADDRSPC  
COND  
DPRTY  
DYNAMNBR  
PARM  
PERFORM  
RD  
REGION  
TIME

**Keyword Parameters on EXEC Statement that Calls a Procedure:** When the EXEC statement positional parameter calls a cataloged procedure, all of the EXEC statement's keyword parameters override matching EXEC keyword parameters in the called procedure. If a keyword parameter is to override a parameter on only one EXEC statement in the procedure, code **.procstepname** immediately following the keyword. The procstepname is the name field of the procedure EXEC statement containing the keyword parameter to be overridden. For example:

```
//STEP1 EXEC PROC=WKREPORT,ACCT.PSTEPWED=5670
```

The accounting information **5670** applies only to step **PSTEPWED** in the procedure **WKREPORT**.

## Comments Field

The comments field follows the parameter field after at least one intervening blank.

## Location in the JCL

An EXEC statement must be the first statement in each job step or cataloged or in-stream procedure step.

## Examples of EXEC Statements

```
//STEP4 EXEC PGM=DREC,PARM='3018,NO'
```

The EXEC statement named STEP4 invokes a program named DREC and passes the value in the PARM parameter to DREC.

---

```
// EXEC PGM=ENTRY,TIME=(2,30)
```

This EXEC statement, which does not have a stepname, invokes a program named ENTRY and specifies the maximum processor time for execution of the step.

---

```
//FOR EXEC PROC=PROC489
```

The EXEC statement named FOR invokes a cataloged or in-stream procedure named PROC489.

## EXEC: ACCT

### ACCT Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the ACCT parameter to specify one or more subparameters of accounting information that apply to this step. The system passes the accounting information to the installation's accounting routines.

**References:** For more information on the ACCT parameter, see "Installation Management Information: The ACCT Parameter" on page 4-10, and on how to add accounting routines, see *SPL: System Management Facilities*.

**Syntax:**

```
ACCT[.procstepname]=(accounting-information)
```

**Single Subparameter:** You can omit the parentheses if the accounting information consists of only one subparameter.

**Length:** The entire accounting-information must not exceed 142 characters:

- Including any commas, which are considered part of the information.
- Excluding any enclosing parentheses or apostrophes, which are not considered part of the information.

**Multiple Subparameters:** When the accounting-information consists of more than one subparameter, separate the subparameters by commas and enclose the information in parentheses or apostrophes. For example, ACCT=(5438,GROUP6) or ACCT='5438,GROUP6'.

**Special Characters:** When a subparameter contains special characters, other than hyphens, enclose it in apostrophes and the information in parentheses or enclose all of the information in apostrophes. For example, ACCT=(387,'72/159') or ACCT='387,72/159'.

Code each apostrophe that is part of the accounting-information as two consecutive apostrophes. For example, code DEPT'D58 as ACCT='DEPT''D58'

**Continuation onto Another Statement:** Enclose the accounting-information in parentheses. End each statement with a comma after a complete subparameter. For example:

```
//STEP1 EXEC PGM=WRITER,ACCT=(1417,J318,'D58/920','CHG=2',  
//      '33.95')
```

## Subparameter Definition

### accounting-information

Specifies one or more subparameters of accounting information, as defined by the installation.

## On EXEC Statement that Calls a Procedure

If the EXEC statement calls a cataloged or in-stream procedure, the ACCT parameter overrides the ACCT parameter on or is added to:

- The EXEC statement named in the procstepname qualifier. The information applies only to the named procedure step. The EXEC statement can have as many ACCT.procstepname parameters as the procedure has steps; each ACCT parameter must specify a unique procstepname.
- All EXEC statements in the procedure if procstepname is not coded. Then the information applies to all steps in the called procedure.

## Examples of the ACCT Parameter

```
//STEP1 EXEC PGM=JP5,ACCT=(LOCATION8,'CHGE+3')
```

This EXEC statement executes program JP5 and specifies accounting information for this job step.

```
//STP3 EXEC PROC=LOOKUP,ACCT=(' /83468')
```

This EXEC statement calls cataloged or in-stream procedure LOOKUP. The accounting information applies to this job step, STP3, and to all the steps in procedure LOOKUP.

```
//STP4 EXEC PROC=BILLING,ACCT.PAID=56370,ACCT.LATE=56470,
// ACCT.BILL='121+366'
```

This EXEC statement calls cataloged or in-stream procedure BILLING. The statement specifies different accounting information for each of the procedure steps: PAID, LATE, and BILL.

## EXEC: ADDRSPC

### ADDRSPC Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the ADDRSPC parameter to indicate to the system that the job step requires virtual storage (pageable) or real storage (nonpageable).

**References:** For more information on the ADDRSPC parameter, see "Requesting Storage for Execution" on page 5-23 and "The REGION Parameter" on page 5-25.

**Syntax:**

```
ADDRSPC [.procstepname] = {VIRT | REAL}
```

### Subparameter Definition

#### **VIRT**

Requests virtual storage. The system **can** page the job step.

#### **REAL**

Requests real storage. The system **cannot** page the job step and must place the job step in real storage.

### Defaults

If no ADDRSPC parameter is specified, the default is VIRT.

### Overrides

The JOB statement ADDRSPC parameter applies to all steps of the job and overrides any EXEC statement ADDRSPC parameters.

Code EXEC statement ADDRSPC parameters when each job step requires different types of storage. The system uses an EXEC statement ADDRSPC parameter only when no ADDRSPC parameter is on the JOB statement and only during the job step.

### Relationship to the JOB or EXEC REGION Parameter

Code a REGION parameter to specify how much storage the job step needs. If you omit the REGION parameter, the system uses an installation default specified at JES initialization.



## On EXEC Statement that Calls a Procedure

If this EXEC statement calls a cataloged or in-stream procedure, the ADDRSPC parameter overrides the ADDRSPC parameter on or is added to:

- The EXEC statement named in the procstepname qualifier. The parameter applies only to the named procedure step. The EXEC statement can have as many ADDRSPC.procstepname parameters as the procedure has steps; each ADDRSPC parameter must specify a unique procstepname.
- All EXEC statements in the procedure if procstepname is not coded. Then the parameter applies to all steps in the called procedure.

## Examples of the ADDRSPC Parameter

```
//CAC1 EXEC PGM=A,ADDRSPC=VIRT
```

This EXEC statement executes program A and requests virtual (pageable) storage. Because the REGION parameter is not specified, the storage available to this job step is the installation default or the region size specified on the JOB statement.

---

```
//CAC2 EXEC PROC=B,ADDRSPC=REAL,REGION=80K
```

This EXEC statement calls procedure B and requests real (nonpageable) storage. The REGION parameter specifies 80K of storage.

# EXEC: COND

## COND Parameter

**Parameter Type:** Keyword, optional, JES2 only

**Purpose:** Use the COND parameter to specify the return code tests the system is to use to determine whether to bypass this job step. The system performs each COND parameter test against the return code from every previous job step or from the named previous job step(s). If none of these tests is satisfied, the system executes this job step; if any test is satisfied, the system bypasses this job step.

If a job step fails, the system normally bypasses all following job steps. To make the system execute a following step, for instance, to write a dump, code EVEN or ONLY on that step's EXEC statement. The EVEN or ONLY subparameters are interpreted first. If they indicate that the job step should be executed, then the return code tests, if specified, are performed. If no return code tests were coded or if none of the coded tests is satisfied, the system executes the step.

Instead of coding a JOB statement COND parameter, code an EXEC statement COND parameter when you want to:

- Specify different tests for each job step.
- Bypass only one step, rather than all subsequent steps in the job.
- Name a specific step whose return code the system is to test.
- Specify special conditions for executing a job step.

The tests are made against return codes from the current execution of the job.

**Note:** The EXEC COND parameter is supported only on JES2 systems. JES3 processes all jobs as though each step will execute.

**References:** For more information on the COND parameter, see "Conditionally Executing Job Steps" on page 5-5.

### Syntax:

```
COND [.procstepname] = ( (code,operator [,stepname] [.procstepname])  
                        [, (code,operator [,stepname] [.procstepname])] ... [ ,EVEN ] )  
                        [ ,ONLY ] )
```

- One return code test is: (code,operator)
- You can omit the outer parentheses if you code only one return code test or only EVEN or ONLY.
- Specify up to eight return code tests. However, if you code EVEN or ONLY, specify up to seven return code tests.
- You can omit all return code tests and code only EVEN or ONLY.
- Place the EVEN or ONLY subparameters before, between, or after the return code tests.

## Subparameter Definition

### code

Specifies a number that the system compares to the return codes from all previous steps in the job or from specific steps. `code` is a decimal number from 0 through 4095.

*Note:* Specifying a decimal number greater than 4095 could result in invalid return code testing or invalid return codes in messages.

### operator

Specifies the type of comparison to be made to the return code. Operators and their meanings are:

Operator	Meaning
GT	Greater than
GE	Greater than or equal to
EQ	Equal to
NE	Not equal to
LT	Less than
LE	Less than or equal to

### stepname

Identifies the EXEC statement of the earlier job step that issues the return code to be used in the test. If the specified step is in a procedure, this step must be in the same procedure; otherwise, the specified step must not be in a procedure.

### stepname.procstepname

Identifies a step in a cataloged or in-stream procedure called by an earlier job step. `stepname` identifies the EXEC statement of the calling job step; `procstepname` identifies the EXEC statement of the procedure step that issues the return code to be used in the test.

### EVEN

Specifies that this job step is to be executed **even if** a preceding job step abnormally terminated. When **EVEN** is coded, the system:

- Does not test the return code of any steps that terminated abnormally.
- Does test the return code of any steps that terminated normally. If none of the return code tests for these steps is satisfied, this job step is executed.

If the operator terminated a job step with a **CANCEL** command, the system ignores **EVEN**.

### ONLY

Specifies that this job step is to be executed **only if** a preceding step abnormally terminated. When **ONLY** is coded, the system:

- Does not test the return code of any steps that terminated abnormally.
- Does test the return code of any steps that terminated normally. If none of the return code tests for these steps is satisfied, this job step is executed.

If the operator terminated a job step with a **CANCEL** command, the system ignores **ONLY**.

## EXEC: COND

### Overrides

If you code the COND parameter on the JOB statement and on one or more of the job's EXEC statements, and if a return code test on the JOB statement is satisfied, the job terminates. In this case, the system ignores any EXEC statement COND parameters.

If the tests on the JOB statement are not satisfied, the system then performs the return code tests on the EXEC statement. If a return code test is satisfied, the step is bypassed.

### On EXEC Statement that Calls a Procedure

If this EXEC statement calls a cataloged or in-stream procedure, the COND parameter overrides the COND parameter on or is added to:

- The EXEC statement named in the procstepname qualifier, which is to the left of the equals sign. The parameter applies only to the named procedure step. The EXEC statement can have as many COND.procstepname parameters as the procedure has steps; each COND parameter must specify a unique procstepname.
- All EXEC statements in the procedure if procstepname is not coded. Then the parameter applies to this job step and to all steps in the called procedure.

### Cautions when Specifying COND Parameters

**Backward References to Data Sets:** If a step is bypassed because of its COND parameter or if a step abnormally terminates, a data set that was to have been created or cataloged in the step may not exist, may not be cataloged, or may be incomplete. Thus, a job step that specifies the EVEN or ONLY subparameter should not refer to a data set being created or cataloged in a step that could be bypassed or abnormally terminated.

**JOBLIB and COND=ONLY:** If the job contains a JOBLIB DD statement and ONLY is specified in a job step, the JOBLIB unit and volume information are not passed to the next step; when the next step is executed, the system searches the catalog for the JOBLIB data set.

**Job Time Out:** The system abnormally terminates a job with a system completion code of 322 if the EXEC or JOB statement TIME parameter or the default time limit specified at JES initialization is exceeded. This time out occurs regardless of any COND parameters.

**When the JOB Statement Contains a RESTART Parameter:** When restarting a job, do not specify in the deferred restart step or in any following steps a COND parameter that refers to a stepname or stepname.procstepname for a step before the restart step. The system ignores any COND parameters that refer to preceding steps. For information on submitting a job for restart, see "RESTART Parameter" on page 10-33 and "Restarting a Job at a Step or Checkpoint" on page 5-26.

## Summary of COND Parameters

See Figure 11-1 for a summary of how to code tests in the COND parameter to cause the current step to be executed or bypassed. See Figure 11-2 for the effect of the EVEN and ONLY subparameters on step execution.

Test in COND Parameter	Return Code (RC) from Previous Step	
	Execute Current Step	Bypass Current Step
COND=(code,GT)	RC ≥ code	RC < code
COND=(code,GE)	RC > code	RC ≤ code
COND=(code,EQ)	RC = code	RC ≠ code
COND=(code,LT)	RC ≤ code	RC > code
COND=(code,LE)	RC < code	RC ≥ code
COND=(code,NE)	RC = code	RC ≠ code

Figure 11-1. Execution or Bypassing of Current Step Based on COND Parameter

EVEN or ONLY Specified?	Any Preceding Abend?	Any Tests Satisfied	Current Step Execute?
EVEN	No	No	Yes
EVEN	No	Yes	No
EVEN	Yes	No	Yes
EVEN	Yes	Yes	No
ONLY	No	No	No
ONLY	No	Yes	No
ONLY	Yes	No	Yes
ONLY	Yes	Yes	No
Neither	No	No	Yes
Neither	No	Yes	No
Neither	Yes	No	No
Neither	Yes	Yes	No

Figure 11-2. Effect of EVEN and ONLY Subparameters on Step Execution

## EXEC: COND

### Examples of the COND Parameter

```
//STEP6 EXEC PGM=DISKUTIL,COND=(4,GT,STEP3)
```

In this example, if the return code from STEP3 is 0 through 3, the system bypasses STEP6. If the return code is 4 or greater, the system executes STEP6. Because neither EVEN nor ONLY is specified, the system does not execute this step if a preceding step abnormally terminates.

---

```
//TEST2 EXEC PGM=DUMPINT,COND=((16,GE),(90,LE,STEP1),ONLY)
```

The system executes this step ONLY if two conditions are met:

1. A preceding job step abnormally terminated.
2. No return code tests are satisfied.

Therefore, the system executes this step only when all three of the following are true:

- A preceding job step abnormally terminated.
- The return codes from all preceding steps are 17 or greater.
- The return code from STEP1 is 89 or less.

The system bypasses this step if any one of the following is true:

- All preceding job steps terminated normally.
  - The return codes from all preceding steps are 0 through 16.
  - The return code from STEP1 is 90 or greater.
- 

```
//STP4 EXEC PROC=BILLING,COND.PAID=((20,LT),EVEN),  
// COND.LATE=(60,GT,FIND),  
// COND.BILL=((20,GE),(30,LT,CHGE))
```

This statement calls cataloged or in-stream procedure BILLING. The statement specifies different return code tests for each of the procedure steps: PAID, LATE, and BILL. The system executes step PAID even if a preceding step abnormally terminates unless the accompanying return code is satisfied.

## DPRTY Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the DPRTY parameter to assign a dispatching priority to the address space for this job step. The system uses the dispatching priority to determine the order in which to execute tasks.

**References:** For more information on the DPRTY parameter, see “Assigning a Dispatching Priority to Job Steps” on page 5-21.

**Syntax:**

```
DPRTY[.procstepname]=([value1][,value2])
```

- You can omit the parentheses if you code only value1.
- You must include the parentheses and code a comma before value2 if you code only value2.

### Subparameter Definition

#### value1

Indicates whether this job step is to have the same or a different priority than the job. value1 is a number from 0 through 15.

JES2 determines the job priority from one of the following:

- The JES2 /\*PRIORITY statement.
- A value calculated from the accounting information on the JOB statement or the JES2 /\*JOBPARM statement.
- An installation default.

#### value2

Specifies a number to be added to value1 to form the dispatching priority. value2 is a number from 0 through 15. The system forms the internal dispatching priority as follows:

$$\text{dispatching priority} = (\text{value1})(16) + \text{value2}$$

# EXEC: DPRTY

## Defaults

If you omit the DPRTY parameter, the system assigns the job step the APG (automatic priority group) priority.

If you omit value1 or it is equal to the APG priority, the system assigns the step the APG priority and ignores value2. In this case, the system obtains value2 from the Installation Performance Specification (IPS) using the performance group associated with the job step. See *SPL: Initialization and Tuning Guide* for information on IPS. If value2 is not specified in the IPS, the system makes value2 equal to 6.

## On EXEC Statement that Calls a Procedure

If this EXEC statement calls a cataloged or in-stream procedure, the DPRTY parameter overrides the DPRTY parameter on or is added to:

- The EXEC statement named in the procstepname qualifier. The parameter applies only to the named procedure step. The EXEC statement can have as many DPRTY.procstepname parameters as the procedure has steps; each DPRTY parameter must specify a unique procstepname.
- All EXEC statements in the procedure if procstepname is not coded. Then the parameter applies to this job step and to all steps in the called procedure.

## Examples of the DPRTY Parameter

```
//BP2 EXEC PGM=FOUR,DPRTY=(13,9)
```

The system uses the values in this DPRTY parameter to form a dispatching priority for this step. Because the numbers are relatively high, the dispatching priority will be high: 217.

---

```
//STEP EXEC PROC=CLEAN,DPRTY=(11,7)
```

This EXEC statement calls a cataloged procedure named CLEAN, which has three steps. The DPRTY parameter applies to all three steps. The dispatching priority is 183.

---

```
//STEP EXEC PROC=CLEAN,DPRTY.UP=(13,7)
```

In this statement, the DPRTY parameter applies only to the procedure step UP. The dispatching priority for UP is 167.



## DYNAMNBR Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the DYNAMNBR parameter to tell the system to hold a number of resources in anticipation of reuse. Code DYNAMNBR instead of several DD statements with DYNAM parameters.

**Reference:** For more information on the DYNAMNBR parameter, see “Dynamically Allocating and Deallocating Data Sets” on page 4-12.

**Syntax:**

```
DYNAMNBR[.procstepname]=n
```

EXEC

### Subparameter Definition

**n**  
Specifies a value used to calculate the maximum number of data set allocations that the system can hold in anticipation of reuse. *n* is a decimal number from 0 through the value: 1635 minus the number of DD statements in the step.

The number of resources that the system actually holds in anticipation of reuse equals *n* plus the number of DD statements in the step, including any DD statements in a cataloged or in-stream procedure called by the step.

### Defaults

If no DYNAMNBR parameter is specified, the default is 0. If you code DYNAMNBR incorrectly, the system uses the default of 0 and issues a JCL warning message.

### On EXEC Statement that Calls a Procedure

If this EXEC statement calls a cataloged or in-stream procedure, the DYNAMNBR parameter overrides the DYNAMNBR parameter on or is added to:

- The EXEC statement named in the procstepname qualifier. The parameter applies only to the named procedure step. The EXEC statement can have as many DYNAMNBR.procstepname parameters as the procedure has steps; each DYNAMNBR parameter must specify a unique procstepname.
- All EXEC statements in the procedure if procstepname is not coded. Then the parameter applies to all steps in the called procedure.

## EXEC: DYNAMNBR

### Example of the DYNAMNBR Parameter

```
//STEP1 EXEC PROC=ACCT,DYNAMNBR.CALC=12
```

For the procedure step CALC, this statement specifies that the system should hold the following data set allocations for reuse: 12 plus the number of DD statements following this EXEC statement and the number of DD statements in procedure ACCT.

## PARM Parameter

Use the PARM parameter to pass variable information to the processing program executed by this job step.

**References.:** For more information on the PARM parameter, see “Passing Information to the Program in Execution” on page 4-8. For details on the format of the passed information, see *SPL: Supervisor Services and Macro Instructions*.

### Syntax:

```
PARM[.procstepname]=information
```

## Subparameter Definition

### information

Consists of the information to be passed to the processing program.

**Length:** The entire information passed must not exceed 100 characters:

- Including any commas, which are passed to the processing program.
- Excluding any enclosing parentheses or apostrophes, which are not passed.

For example, PARM = 'P1,123,MT5' is received by the program as P1,123,MT5.

**Commas:** When the information consists of more than one expression, separate the expressions by commas and enclose the information in parentheses or apostrophes. For example, PARM = (P1,123,MT5) or PARM = 'P1,123,MT5'.

**Special Characters:** When an expression contains special characters, enclose it in apostrophes and the information in parentheses or all the information in apostrophes. For example, PARM = (P50,'12 + 80') or PARM = 'P50,12 + 80'.

Code each apostrophe and ampersand that is part of the information as two consecutive apostrophes or ampersands. For example, code 3462&5 as PARM = '3462&&5'.

**Continuation onto Another Statement:** Enclose the information in parentheses. End each statement with a comma after a complete expression. For example:

```
//STEP1 EXEC PGM=WORK,PARM=(DECK,LIST,'LINECNT=80',
// '12+80',NOMAP)
```

## EXEC: PARM

### On EXEC Statement that Calls a Procedure

If this EXEC statement calls a cataloged or in-stream procedure, the PARM parameter overrides the PARM parameter on or is added to:

- The EXEC statement named in the procstepname qualifier. The information applies only to the named procedure step. The EXEC statement can have as many PARM.procstepname parameters as the procedure has steps; each PARM parameter must specify a unique procstepname.
- The EXEC statement in the procedure if procstepname is not coded; the system nullifies any PARM parameters on any following EXEC statements in the procedure.. The information applies to only the first step in the called procedure.

### Examples of the PARM Parameter

```
//RUN3 EXEC PGM=APG22,PARM='P1,123,P2=5'
```

The system passes P1,123,P2=5 to the processing program named APG22.

---

```
// EXEC PROC=PROC81,PARM=MT5
```

The system passes MT5 to the first step of the procedure named PROC81. If PROC81 contains more steps and their EXEC statements contain PARM parameters, the system nullifies those PARM parameters.

---

```
//STP6 EXEC PROC=ASMFCLG,PARM.LKED=(MAP,LET)
```

The system passes MAP,LET to the procedure step named LKED in procedure ASMFCLG. If any other procedure steps in ASMFCLG contain a PARM parameter, those PARM parameters remain in effect.

---

```
//RUN4 EXEC PGM=IFOX00,PARM=(NOBJECT,'LINECNT=50',  
// DECK)
```

The system passes NOBJECT,LINECNT=50,DECK to processing program IFOX00. Because the PARM parameter is continued on a second statement, the information is enclosed in parentheses; notice that the continuation occurs after a comma following a complete expression.

## PERFORM Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the PERFORM parameter to specify the performance group for the job step. The installation-defined performance groups determine the rate at which associated steps have access to the processor, storage, and channels.

**References:** For more information on the performance groups, see “Performance of Jobs and Job Steps in JES2” on page 5-22 and “Performance of Jobs and Job Steps in JES3” on page 5-23.

**Syntax:**

```
PERFORM[.procstepname]=n
```

EXEC

### Subparameter Definition

**n**

Requests a performance group. The *n* is a number from 1 through 999 and must identify a performance group that has been defined by your installation. The specified performance group should be appropriate for your step type according to your installation's rules.

### Defaults

If no PERFORM parameter is specified or if the specified PERFORM number fails validity checks, the system uses an installation default specified at initialization. If the installation did not specify a default, the system uses a built-in default:

- 1 for non-TSO job steps
- 2 for TSO sessions

See *SPL: Initialization and Tuning Guide* for details.

### Overrides

A JOB statement PERFORM parameter applies to all steps of the job and overrides any EXEC statement PERFORM parameters.

Code EXEC statement PERFORM parameters when each job step is to execute in a different performance group. The system uses an EXEC PERFORM parameter only when no PERFORM parameter is on the JOB statement and only during the job step.

# EXEC: PERFORM

## On EXEC Statement that Calls a Procedure

If this EXEC statement calls a cataloged or in-stream procedure, the PERFORM parameter overrides the PERFORM parameter on or is added to:

- The EXEC statement named in the procstepname qualifier. The parameter applies only to the named procedure step. The EXEC statement can have as many PERFORM.procstepname parameters as the procedure has steps; each PERFORM parameter must specify a unique procstepname.
- All EXEC statements in the procedure if procstepname is not coded. Then the parameter applies to all steps in the called procedure.

## Example of the PERFORM Parameter

```
//STEP4 EXEC PGM=ADDER,PERFORM=60
```

This job step will be run in performance group 60 if it passes validity checks. The installation must have defined the significance of this performance group.

## PGM Parameter

**Parameter Type:** Positional, optional

**Purpose:** Use the PGM parameter to name the program that the system is to execute. The specified program must be a member of a partitioned data set used as a system library, a private library, or a temporary library.

**References:** For more information on naming programs and on temporary and private libraries, see “Processing Program Information” on page 4-4.

**Syntax:**

```

      {program-name                }
PGM={*.stepname.ddname           }
      {*.stepname.procstepname.ddname}

```

The EXEC statement parameter field must begin with a PGM parameter or a PROC parameter. These two parameters must not appear on the same EXEC statement.

## Subparameter Definition

### **program-name**

Specifies the member name or alias of the program to be executed. The program-name is 1 through 8 alphanumeric or national characters; the first character must be alphabetic or national.

### **\*.stepname.ddname**

Refers to a DD statement that defines, as a member of a partitioned data set, the program to be executed. Stepname identifies the EXEC statement of the earlier job step that contains the DD statement with ddname in its name field.

This form of the parameter is usually used when the previous job step created a temporary partitioned data set to store a program until it is required.

### **\*.stepname.procstepname.ddname**

Refers to a DD statement that defines, as a member of a partitioned data set, the program to be executed. The DD statement is in a cataloged or in-stream procedure that is called by an earlier job step. Stepname identifies the EXEC statement of the calling job step; procstepname identifies the EXEC statement of the procedure step that contains the DD statement with ddname in its name field.

## EXEC: PGM

### Checking JCL Syntax without Executing the Step

In a JES3 system, code PGM=JCLTEST or PGM=JSTTEST to scan the job step's JCL for syntax errors without executing the job or allocating devices. JCLTEST or JSTTEST provide the same function as provided by the JOB statement TYPRUN=SCAN parameter.

For more information, see:

- For PGM=JCLTEST or PGM=JSTTEST, "Testing JCL Without Execution (JES3)" on page 3-25.
- For the JOB statement TYPRUN=SCAN parameter, "Bypassing Job Initiation" on page 3-25 and "TYPRUN Parameter" on page 10-39.

### Examples of the PGM Parameter

```
//JOB8 JOB ,BOB,MSGLEVEL=(2,0)
//JOBLIB DD DSNAME=DEPT12.LIB4,DISP=(OLD,PASS)
//STEP1 EXEC PGM=USCAN
```

These statements indicate that the system is to search the private library DEPT12.LIB4 for the member named USCAN, read the member into storage, and execute the member.

---

```
//PROCESS JOB ,MARY,MSGCLASS=A
//CREATE EXEC PGM=IEWL
//SYSLMOD DD DSNAME=&&PARTDS(PROG),UNIT=3350,DISP=(MOD,PASS),
// SPACE=(1024,(50,20,1))
//GO EXEC PGM=*.CREATE.SYSLMOD
```

The EXEC statement named GO contains a backward reference to DD statement SYSLMOD, which defines a library created in the step named CREATE. Program PROG is a member of the partitioned data set &&PARTDS, which is a temporary data set. Step GO executes program PROG. The data set &&PARTDS is deleted at the end of the job.

---

```
//JOB01 JOB ,JOHN,MSGCLASS=H
//STEP2 EXEC PGM=UPDT
//DDA DD DSNAME=SYS1.LINKLIB(P40),DISP=OLD
//STEP3 EXEC PGM=*.STEP2.DDA
```

The EXEC statement named STEP3 contains a backward reference to DD statement DDA, which defines system library SYS1.LINKLIB. Program P40 is a member of SYS1.LINKLIB; STEP3 executes program P40.

---

```
//CHECK EXEC PGM=IEFBR14
```

This EXEC statement specifies execution of the program IEFBR14, which is a two-line user-written program that consists of an entry point and a branch to the contents of register 14. This program is handy during testing: execute it to test JCL space allocation and disposition requests before executing your program. The system checks all the job control statements in the job for syntax.

---



```

//USUAL      JOB      A2317P,'MAE BIRDSALL'
//ASM        EXEC     PGM=IEV90,REGION=256K,          EXECUTES ASSEMBLER
//          PARM=('OBJECT','NODECK,LINECOUNT(50)')
//SYSPRINT   DD      SYSOUT=*,DCB=BLKSIZE=3509        THE ASSEMBLY LISTING
//SYSLIB     DD      DSNAME=SYS1.MACLIB,DISP=SHR      THE MACRO LIBRARY
//SYSUT1     DD      DSNAME=&&SYSUT1,UNIT=SYSDA,      A WORK DATA SET
//          SPACE=(CYL,(10,1))
//SYSLIN     DD      DSNAME=&&OBJECT,UNIT=SYSDA,      THE OUTPUT OBJECT MODULE
//          SPACE=(TRK,(10,2)),DCB=BLKSIZE=3120,DISP=(,PASS)
//SYSIN      DD      *                                IN-STREAM SOURCE CODE
.
.
code
.
//LKED      EXEC     PGM=HEWL,                        EXECUTES LINKAGE EDITOR
//          PARM='XREF,LIST,LET',COND=(8,LE,ASM)
//SYSPRINT   DD      SYSOUT=*                        LINKEDIT MAP PRINTOUT
//SYSLIN     DD      DSNAME=&&OBJECT,DISP=(OLD,DELETE) INPUT OBJECT MODULE
//SYSUT1     DD      DSNAME=&&SYSUT1,UNIT=SYSDA,      A WORK DATA SET
//          SPACE=(CYL,(10,1))
//SYSLMOD    DD      DSNAME=&&LOADMOD,UNIT=SYSDA,      THE OUTPUT LOAD MODULE
//          DISP=(MOD,PASS),SPACE=(1024(50,20,1))
//GO         EXEC     PGM=*.LKED.SYSLMOD,TIME=(,30), EXECUTES THE PROGRAM
//          COND=(8,LE,ASM),(8,LE,LKED)
//SYSUDUMP   DD      SYSOUT=*                        IF FAILS, DUMP LISTING
//SYSPRINT   DD      SYSOUT=*,                      OUTPUT LISTING
//          DCB=(RECFM=FBA,LRECL=121)
//OUTPUT     DD      SYSOUT=A,                      PROGRAM DATA OUTPUT
//          DCB=(LRECL=100,BLKSIZE=3000,RECFM=FBA)
//INPUT      DD      *                                PROGRAM DATA INPUT
.
.
data
.
/*
//

```

This example shows JCL that can be used to:

- Assemble object code entered in the input stream: the step named ASM.
- Linkedit the object module, if the assembly did not result in a return code of 8 or higher: the step named LKED.
- Execute the linkedited module, if neither the assembly nor the linkage editing resulted in a return code of 8 or higher: the step named GO.

## EXEC: PROC and Procedure Name

### PROC and Procedure Name Parameters

**Parameter Type:** Positional, optional

**Purpose:** Use the PROC parameter to specify that the system is to call and execute a cataloged or in-stream procedure.

**References:** For more information on the PROC parameter, see "Processing Program Information" on page 4-4, and on cataloged and in-stream procedures, see Chapter 9, "Guide to Cataloged and In-Stream Procedures."

**Syntax:**

```
{PROC=procedure-name}  
{procedure-name }
```

- The EXEC statement parameter field must begin with a PGM parameter or a PROC parameter. These two parameters must not appear on the same EXEC statement.
- You can omit PROC = and code only the procedure-name.

### Subparameter Definition

#### **procedure-name**

Identifies the procedure to be called and executed:

- The member name or alias of a cataloged procedure.
- The name of an in-stream procedure. The in-stream procedure must appear earlier in this job.

The procedure-name is 1 through 8 alphanumeric or national characters; the first character must be alphabetic or national.

### Effect of PROC Parameter on Other Parameters and Following Statements

Because this EXEC statement calls a cataloged or in-stream procedure, the other parameters on the statement are added to or override corresponding parameters on the EXEC statements in the called procedure. See the descriptions of the other parameters for details of their effects.

Any DD statements following this EXEC statement are added to the procedure or override or nullify corresponding DD statements in the procedure. For details, see "DD Statements for Cataloged and In-stream Procedures" on page 12-3.

## Examples of the PROC Parameter

```
//SP3 EXEC PROC=PAYWKRS
```

This statement calls the cataloged or in-stream procedure named PAYWKRS.

---

```
//BK EXEC OPERATE
```

This statement calls the cataloged or in-stream procedure named OPERATE.

EXEC

## RD Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the RD (restart definition) parameter to:

- Specify that the operator is to perform automatic step restart if the job fails.
- Suppress, partially or totally, the action of the assembler language CHKPT macro instruction or the DD statement CHKPT parameter.

The system can perform automatic restart only if the job has a job journal. A job journal is a sequential data set that contains job-related control blocks needed for restart. For JES2, specify a job journal in one of the following:

- An installation option during JES2 initialization to specify a checkpoint data set.
- RESTART parameter on the JOB statement.
- RD=R or RD=RNC on either the JOB statement or any one EXEC statement in the job.

For JES3, specify a job journal in one of the following:

- An installation option during JES3 initialization.
- RD=R or RD=RNC on either the JOB statement or any one EXEC statement in the job.
- JOURNAL=YES on a JES3 `//*MAIN` statement in the job.

**References:** For more information on the RD parameter, see “The RD Parameter on the EXEC Statement” on page 5-28, and on restarting jobs, see:

- “Restarting a Job at a Step or Checkpoint” on page 5-26.
- “RD Parameter” on page 10-28.

For JES2 systems, see the RESTART parameter on the `/*JOBPARM` control statement in “`/*JOBPARM` Statement” on page 16-4.

For JES3 systems, see the FAILURE parameter on the `//*MAIN` control statement in “`//*MAIN` Statement” on page 17-23.

For detailed information on deferred checkpoint restart, see *Checkpoint/Restart*.

**Syntax:**

```
RD[.procstepname]={R|RNC|NC|NR}
```

## Subparameter Definition

### R (Restart)

Indicates that the operator is to perform automatic step restart if the job step fails.

RD=R does not suppress checkpoint restarts:

- If the processing program executed in a job step does not include a CHKPT macro instruction, RD=R allows the system to restart execution at the beginning of the abnormally terminated step.
- If the program includes a CHKPT macro instruction, RD=R allows the system to restart execution at the beginning of the step, if the step abnormally terminates before the CHKPT macro instruction is executed.
- If the step abnormally terminates after the CHKPT macro instruction is executed, only checkpoint restart can occur. If you cancel the affects of the CHKPT macro instruction before the system performs a checkpoint restart, the request for automatic step restart is again in effect.

### RNC (Restart and No Checkpoint)

Indicates that the operator is to perform automatic step restart if the job step fails.

RD=RNC suppresses automatic and deferred checkpoint restarts. It suppresses:

- Any CHKPT macro instruction in the processing program: That is, the operator is not to perform an automatic checkpoint restart, and the system is not to perform a deferred checkpoint restart if the job is resubmitted.
- The DD statement CHKPT parameter.
- The checkpoint at end-of-volume (EOV) facility; see “SYSCKEOV DD Statement” on page 13-19.

### NC (No Checkpoint)

Indicates that the operator is **not** to perform automatic step restart if the job step fails.

RD=NC suppresses automatic and deferred checkpoint restarts. It suppresses:

- Any CHKPT macro instruction in the processing program.
- The DD statement CHKPT parameter.
- The checkpoint at EOV facility.

## EXEC: RD

### NR (No Automatic Restart)

Indicates that the operator is **not** to perform automatic step restart if the job fails.

RD=NR suppresses automatic checkpoint restart but permits deferred checkpoint restarts. It permits:

- A CHKPT macro instruction to establish a checkpoint.
- The job to be resubmitted for restart at the checkpoint. On the JOB statement when resubmitting the job, specify the checkpoint in the RESTART parameter.

If you code RD=NR and the system fails, RD=NR does not prevent the job from restarting.

### Defaults

If no RD parameter is specified, the terminated job step is eligible for automatic checkpoint/restart, if its program requested checkpoints with a CHKPT macro instruction.

### Overrides

- A JOB statement RD parameter applies to all steps of the job and overrides any EXEC statement RD parameters.

When no RD parameter is on the JOB statement, the system uses an EXEC statement RD parameter, but only during the job step. Code EXEC statement RD parameters when you want to specify different restart types for each job step.

- A request by a CHKPT macro instruction for an automatic checkpoint restart overrides a request by a JOB or EXEC statement RD=R parameter for automatic step restart.

### Relationship to Other Control Statements

Code RD=NC or RD=RNC to suppress the action of the DD statement CHKPT parameter.

### On EXEC Statement that Calls a Procedure

If the EXEC statement calls a cataloged or in-stream procedure, the RD parameter is added to or overrides the RD parameter on:

- The EXEC statement named in the procstepname qualifier. The information applies only to the named procedure step. The EXEC statement can have as many RD.procstepname parameters as the procedure has steps; each RD parameter must specify a unique procstepname.
- All EXEC statements in the procedure if procstepname is not coded. Then the parameter applies to all steps in the called procedure.

## Examples of the RD Parameter

```
//STEP1 EXEC PGM=GIIM, RD=R
```

RD=R specifies that the operator is to perform automatic step restart if the job step fails.

---

```
//NEST EXEC PGM=T18, RD=RNC
```

RD=RNC specifies that, if the step fails, the operator is to perform automatic step restart. RD=RNC suppresses automatic and deferred checkpoint restarts.

---

```
//CARD EXEC PGM=WTE, RD=NR
```

RD=NR specifies that the operator is not to perform automatic step restart or automatic checkpoint restart. However, a CHKPT macro instruction can establish checkpoints to be used later for a deferred restart.

---

```
//STP4 EXEC PROC=BILLING, RD.PAID=NC, RD.BILL=NR
```

This statement calls a cataloged or in-stream procedure BILLING. The statement specifies different restart requests for each of the procedure steps: PAID and BILL.

EXEC

# EXEC: REGION

## REGION Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the REGION parameter to specify the amount of space that the step requires.

The specified or default region size sets an upper boundary to limit region size for variable-length GETMAINS. The system uses the upper boundary for variable-length GETMAINS as long as the region still has available at least the minimum amount of storage requested.

In addition, the IBM- or installation-supplied routine IEALIMIT uses the region size to establish a second limiting value. The system uses this second value for:

- Fixed-length GETMAINS.
- Variable-length GETMAINS when the space remaining in the region is less than the requested minimum.

When the minimum requested length for variable-length GETMAINS exceeds this second value, the job step abnormally terminates.

REGION=0K gives the job step all the storage available in the private area, that is, from the top of the system region to the bottom of the common service area (CSA). The resulting size of the region is unpredictable.

**References:** For more information on the REGION parameter, see "Requesting Storage for Execution" on page 5-23. Also, see "ADDRSPC Parameter" on page 11-6. For further information on the region size, see *OS/VS2 Supervisor Services and Macro Instructions*.

**Syntax:**

```
REGION[.procstepname]=valueK
```

## Subparameter Definition

**valueK**

Specifies the required storage in thousands (1024) of bytes. The value is 1 to 5 decimal numbers. Code an even number. For example, REGION=66K. If you code an odd number, the system treats it as the next highest even number.



## Defaults

If no REGION parameter is specified, the system uses an installation default specified at JES initialization.

## Overrides

A JOB statement REGION parameter applies to all steps of the job and overrides any EXEC statement REGION parameters.

When no REGION parameter is on the JOB statement, the system uses an EXEC statement REGION parameter, but only during the job step. Code EXEC statement REGION parameters when you want to specify a different region size for each job step.

EXEC

## Relationship to the EXEC ADDRSPC Parameter

**When ADDRSPC=REAL:** Code a REGION parameter to specify how much real storage the job needs. If you omit the REGION parameter, the system uses the default.

**When ADDRSPC=VIRT or ADDRSPC is Omitted:** Do not code a REGION parameter. The system uses the default.

## Examples of the REGION Parameter

```
//MKBOYLE EXEC PROC=A,ADDRSPC=REAL,REGION=40K
```

The system assigns 40K bytes of real storage to this job step.

---

```
//STP6 EXEC PGM=CONT,REGION=120K
```

The system assigns a region of 120K bytes. When the ADDRSPC parameter is not specified, the system defaults to ADDRSPC=VIRT.

# EXEC: TIME

## TIME Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the TIME parameter to specify the maximum length of time that a job step is to use the processor and to find out through messages how much processor time the step used.

A step that exceeds its allotted time abnormally terminates and causes termination of the job, unless a user exit routine extends the time for the job.

**References.:** For more information on the TIME parameter, see "Limiting Job and Job Step Execution Time" on page 5-16.

**Syntax:**

```
TIME[.procstepname]={1440|([minutes][,seconds])}
```

You can omit the parentheses if you code only 1440 or the processor time in minutes.

If you omit the seconds, do not code a null subparameter. For example, TIME=(60,) is invalid.

### Subparameter Definition

**1440**

Indicates that the step can use the processor for an unlimited amount of time; 1440 literally means 24 hours. Code TIME=1440 for the following reasons:

- To obtain job step accounting information.
- To specify that the system is to allow this step to remain in a wait state for more than the installation-established time limit.

**minutes**

Specifies the maximum number of minutes the step can use the processor. The minutes must be a number from 1 through 1439.

Code TIME=0 on the EXEC statement to indicate that the step is to use the unexpired time from the previous step. If this step exceeds that unexpired time, it abnormally terminates.

**seconds**

Specifies the maximum number of seconds that the step can use the processor, in addition to any minutes that are specified. The seconds must be a number from 1 through 59.

## Defaults

If no TIME parameter is specified, JES uses an installation default specified at initialization.

## Overrides

For a JOB statement TIME parameter other than TIME=1440, the system sets the time limit for each step to:

- The step time limit specified on the EXEC statement TIME parameter.
- If no EXEC TIME parameter was specified, (1) the default time limit or (2) the job time remaining after execution of previous steps, whichever is smaller.

EXEC

## Time Handling

**How the System Converts the Time Value:** The job time limit or the time remaining after execution of previous steps in a job is converted by the system to seconds and then rounded to the nearest unit, where 1 unit = 1.048576 seconds. Thus a step can begin execution with up to one-half unit more or one-half unit less time than expected. If the time remaining for the job is less than one-half unit, a step will begin execution with zero time, resulting in an abnormal termination.

**Time Checking:** Because the system checks the processor time-used field about every 10.5 seconds, the actual time that a job uses the processor can exceed the specified TIME value by up to 10.5 seconds. For example, the system checks the job's time-used field and finds 0.5 seconds remaining. Because the system does not again check the job's time-used field for about 10.5 seconds, the job can execute for an additional 10.5 seconds and thus exceed the coded TIME value by 10 seconds.

## Examples of the TIME Parameter

```
//STEP1 EXEC PGM=GRYS,TIME=(12,10)
```

This statement specifies that the maximum amount of time the step can use the processor is 12 minutes, 10 seconds.

---

```
//FOUR EXEC PGM=JPLUS,TIME=(,30)
```

This statement specifies that the maximum amount of time the step can use the processor is 30 seconds.

---

```
//INT EXEC PGM=CALC,TIME=5
```

This statement specifies that the maximum amount of time the step can use the processor is 5 minutes.

---

## EXEC: TIME

```
//LONG EXEC PGM=INVANL,TIME=1440
```

This statement specifies that the step can have unlimited use of the processor. Therefore, the step can use the processor and can remain in a wait state for an unspecified period of time, if not restricted by the JOB statement TIME parameter.

---

```
//STP4 EXEC PROC=BILLING,TIME.PAID=(45,30),TIME.BILL=(112,59)
```

This statement calls cataloged or in-stream procedure BILLING. The statement specifies different time limits for each of the procedure steps: PAID and BILL.

---

For examples of TIME coded on both the JOB and EXEC statements, see "Examples of the TIME Parameter on JOB and EXEC Statements" on page 10-38.

## Chapter 12. Coding the DD Statement

Use the DD (data definition) statement to describe a data set and to specify the input and output facilities needed for the data set.

The parameters you can specify for data set definition are arranged alphabetically in the following pages.

**References:** For more information on coding DD statement parameters, see Chapter 7, “Guide to Specifying Data Set Information” on page 7-1. For information about the JES initialization parameters that provide installation defaults, see *SPL: JES2 Initialization and Tuning* and *SPL: JES3 Initialization and Tuning*.

### Syntax:

```
//{ddname          } DD [positional-parameter][,keyword-parameter]... comments
   {procstepname.ddname}
```

- The DD statement consists of the characters // in columns 1 and 2 and four fields: name, operation (DD), parameter, and comments.
- A DD statement is required for each data set.
- The maximum DD statements per job step are:
  - 1635, in a JES2 system.
  - Determinated by the installation, in a JES3 system.

### Name Field

Code a ddname as follows:

- Each ddname should be unique within the job step. If duplicate ddnames appear in a job step, processing is as follows:
  - **In a JES2 system:** The system performs device and space allocation and disposition processing for both DD statements; however, it directs all references to the first DD statement in the step.
  - **In a JES3 system:** If both DD statements request JES3- or jointly-managed devices, the system cancels the job during JES3 interpretation. If only one or if neither DD statement requests JES3- or jointly-managed devices, the system performs device and

# DD

space allocation and disposition processing for both DD statements; however, it directs all references to the first DD statement in the step.

- The ddname must begin in column 3.
- The ddname is 1 through 8 alphanumeric or national characters.
- The first character must be alphabetic or national.
- The ddname must be followed by at least one blank.

**Omitting the ddname:** Do not code a ddname in two cases:

- The DD statement defines a data set that is concatenated to the data set of the preceding DD statement.
- The DD statement is the second or third consecutive DD statement for an indexed sequential data set.

**Special ddnames:** Use the following special ddnames only when you want to use the facilities these names represent to the system. These facilities are explained in Chapter 13, "Coding Special DD Statements" on page 13-1.

JOBCAT  
JOBLIB  
STEPCAT  
STEPLIB  
SYSABEND  
SYSCHK  
SYSCKEOV  
SYSMDUMP  
SYSUDUMP

The following ddnames have special meaning to JES3; do not use them on a DD statement in a JES3 system.

JCBIN  
JCBLOCK  
JCBTAB  
JCLIN  
JESInnnn  
JESJCL  
JESMSG  
JOURNAL  
JST  
SYSMSG

## Parameter Field

A DD statement contains two kinds of parameters: positional and keyword. All parameters are optional. However, do not leave the parameter field blank unless the DD statement overrides a DD statement that defines a concatenated data set in a cataloged or in-stream procedure.

**Positional Parameters:** A DD statement can contain **one** of the following positional parameters. It must precede all keyword parameters.

```
*
DATA
DUMMY
DYNAM
```

**Keyword Parameters:** A DD statement can contain the following keyword parameters. You can code any of the keyword parameters in any order in the parameter field after a positional parameter, if coded.

DD

ACCODE	DDNAME	FLASH	PROTECT	UNIT
AMP	DEST	FREE	QNAME	VOLUME
BURST	DISP	HOLD	SPACE	
CHARS	DLM	LABEL	SUBSYS	
CHKPT	DSID	MODIFY	SYSOUT	
CNTL	DSNAME	MSVGP	TERM	
COPIES	FCB	OUTLIM	UCS	
DCB		OUTPUT		

Do not use DD statement parameter or subparameter keywords as symbolic parameters, names, or labels.

## Comments Field

The comments field follows the parameter field after at least one intervening blank. If you do not code any parameters on a DD statement, do not code any comments.

## Location in the JCL

Most DD statements define data sets to be used in a step of a job or of a cataloged or in-stream procedure; these appear after the EXEC statement for the step. Some DD statements define data sets for the job, for example, the JOBLIB DD statement; these appear after the JOB statement and before the first EXEC statement.

## DD Statements for Cataloged and In-stream Procedures

When a job step calls a cataloged or in-stream procedure, DD statements in the calling step (1) override, nullify, or add parameters to DD statements in the procedure or (2) are added to the procedure. These changes affect only the current execution of the job step; the procedure itself is not changed.

# DD

**Location in the JCL:** Place DD statements that override, nullify, or add parameters immediately following the EXEC statement that calls the procedure.

Place added DD statements after all overriding DD statements.

**Order of Overriding DD Statements:** To override more than one DD statement in a procedure, place the overriding DD statements in the same order as the overridden DD statements in the procedure.

**Coding an Overriding DD Statement:** To override parameters on a procedure DD statement, code in the name field of the overriding DD statement the name of the procedure step containing the DD statement, followed by a period, followed by the ddname of the procedure DD statement to be overridden. For example:

```
//PROCSTP1.PROCDD DD parameters
```

**Coding an Added DD Statement:** To add DD statements to a procedure step, code in the name field of the added DD statement the name of the procedure step, followed by a period, followed by a ddname of your choosing.

**In-stream Data for a Procedure:** To supply a procedure step with data from the input stream, code a DD \* or DD DATA statement in the calling step after the last overriding and added DD statement. The name field of this statement must contain the name of the procedure step, followed by a period, followed by a ddname. The ddname can be of your choosing or predefined in the procedure. If it is predefined, it appears in a DDNAME parameter on a procedure DD statement.

**Words Prohibited as Symbolic Parameters:** Do not use the following DD statement keywords as symbolic parameters in procedures to be started by a START command from the operator console.

AFF <sup>1</sup>	DISP	LABEL	SUBALLOC <sup>1</sup>
AMP	DLM	MODIFY	SUBSYS
BURST	DSID	MSVGP	SYSOUT
CHARS	DSNAME	OUTLIM	TERM
CHKPT	DSN	PROTECT	UCS
COPIES	FCB	QNAME	UNIT
DCB	FLASH	SEP <sup>1</sup>	VOLUME
DDNAME	FREE	SPACE	VOL
DEST	HOLD	SPLIT <sup>1</sup>	

<sup>1</sup>These DD statement keywords are from previous releases of MVS. For a description of how they are currently handled, see "JCL Statements no Longer Supported or Supported Differently" on page v.



## Examples of DD Statements and ddnames

```
//INPUT DD DSNAME=FGLIB,DISP=(OLD,PASS)
//      DD DSNAME=GROUP2,DISP=SHR
```

In this example, because the ddname is missing from the second DD statement, the system concatenates the data sets defined in these statements.

---

```
//PAYROLL.DAY DD DSNAME=DESK,DISP=SHR
```

In this example, if procedure step PAYROLL contains a DD statement named DAY, this statement overrides parameters on DD statement DAY. If the step does not contain DD statement DAY, the system adds this statement to procedure step PAYROLL for the duration of the job step.

---

```
//STEPSIX.DD4 DD DSNAME=WRITER,DISP=(NEW,PASS)
//           DD DSNAME=ART,DISP=SHR
```

In this example, the second data set is concatenated to the first, and both are added to procedure step STEPSIX. The ddname is omitted from the second DD statement in order to concatenate data set ART to data set WRITER.

## DD: \* Parameter

### \* Parameter

**Parameter Type:** Positional, optional

**Purpose:** Use the \* parameter to begin an in-stream data set. The data records immediately follow the DD \* statement; the records must be in BCD or EBCDIC. The data records end when the system reads in the input stream a delimiter:

```
/*  
// to indicate another JCL statement  
The two-character delimiter specified by a DLM parameter on this DD statement
```

The data can also end when the input stream runs out of card images.

Use a DATA parameter instead of the \* parameter if any of the data records start with //.

**Syntax:**

```
//ddname DD *[,parameter]...
```

### Defaults

When you do not code DCB=BLKSIZE and DCB=LRECL, JES uses installation defaults specified at initialization.

### Relationship to Other Parameters

The **only** DD parameters that you can code with the \* parameter follow. All other parameters are a JCL error.

```
DCB=BLKSIZE  
DCB=BUFNO  
DCB=LRECL  
DLM  
DSID  
VOLUME=SER
```

**For 3540 Diskette Input/Output Units:** VOLUME=SER, DCB=BUFNO, and DSID parameters on a DD \* statement are ignored except when they are detected by a diskette reader as a request for an associated data set. See *IBM 3540 Programmer's Reference*. On a DD \* or DD DATA statement processed by a diskette reader, you can specify DSID and VOLUME=SER parameters to indicate that a diskette data set is to be merged into the input stream following the DD statement.

**For JES3 SNA RJP Input:** The only parameters you can specify for JES3 systems network architecture (SNA) remote job processing (RJP) input devices are DCB=BLKSIZE and DCB=LRECL.

## Relationship to Other Control Statements

Do not refer to an earlier DD \* statement in DCB or DSNAME parameters on following DD statements.

## Location in the JCL

A DD \* statement begins an in-stream data set.

***In-stream Data for Cataloged or In-stream Procedures:*** A cataloged or in-stream procedure cannot contain a DD \* statement. When you call a procedure, you can add input stream data to a procedure step by placing in the calling step one or more DD \* or DD DATA statements, each followed by data.

***Multiple In-stream Data Sets for a Step:*** You can code more than one DD \* or DD DATA statement in a job step in order to include several distinct groups of data for the processing program. Precede each group with a DD \* or DD DATA statement and follow each group with a delimiter statement. If you omit a DD statement before input data, the system provides a DD \* statement with the ddname of SYSIN; if you omit a following delimiter statement, the system ends the data when it reads a JCL statement or runs out of card images.

DD

## Unread Records

If the processing program does not read all the data in an in-stream data set, the system skips the remaining data without abnormally terminating the step.

## Examples of the \* Parameter

```
//INPUT1  DD  *
          .
          .
          data
          .
//INPUT2  DD  *
          .
          .
          data
          .
/*
```

This example defines two groups of data in the input stream.

---

## DD: \* Parameter

```
//STEP2          EXEC PROC=FRESH
//SETUP.WORK     DD  UNIT=3400-6,LABEL=(,NSL)
//SETUP.INPUT1  DD  *
                .
                .
                data
                .
/*
//PRINT.FRM     DD  UNIT=180
//PRINT.INP     DD  *
                .
                .
                data
                .
/*
```

This example defines two groups of data in the input stream. The input data defined by DD statement SETUP.INPUT1 is to be used by the cataloged procedure step named SETUP. The input data defined by DD statement PRINT.INP is to be used by the cataloged procedure step named PRINT.

## ACCODE Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the ACCODE parameter to specify or change an accessibility code for an ISO/ANSI/FIPS Version 3 tape output data set. An installation-written file-access exit routine verifies the code, after the code is written to tape. If the code is authorized, the job step's program can use the data set; if not, the system issues messages and may abnormally terminate the job step.

A data set protected by an accessibility code should reside only on a volume protected by RACF or a volume accessibility code. The volume should not contain any unprotected data sets.

**Note:** ACCODE is supported only for ISO/ANSI/FIPS Version 3 tape data sets. ACCODE is ignored for SL (IBM standard) label tapes.

**References:** For more information on ISO/ANSI/FIPS Version 3 tape data sets, see *MVS/370 Magnetic Tape Labels and File Structure*.

**Syntax:**

```
ACCODE=access-code
```

### Subparameter definition

#### access-code

Specifies an accessibility code. The access-code is 1 through 8 characters; the first character is an upper case letter from A through Z.

**Note:** Only the first character is used as the ISO/ANSI/FIPS Version 3 accessibility code; the other seven characters can be used by the installation. If the first character is other than an upper case letter from A through Z, the installation does not give control to the file-access exit routine.

### Defaults

If no accessibility code is specified on a DD statement that defines an ISO/ANSI/FIPS Version 3 tape data set, the system writes a blank character (X'40') in the tape label: a blank authorizes unlimited access to the tape's data sets.

If the installation does not supply a file-access exit routine, the system prevents access to any ISO/ANSI/FIPS Version 3 tape volume.

## DD: ACCODE

### Overrides

If PASSWORD or NOPWREAD is coded on the DD statement LABEL parameter, password access overrides the ACCODE parameter.

### Example of the ACCODE Parameter

```
//TAPE DD UNIT=2400,VOLUME=SER=T49850,DSNAME=TAPEDS,  
        LABEL=(,AL),ACCODE=Z
```

In this example, the DD statement ACCODE parameter specifies an accessibility code of Z for tape volume T49850. The volume has ISO/ANSI/FIPS Version 3 labels. The data set TAPEDS is first on the tape.

## AMP Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the AMP parameter to complete information in an access method control block (ACB) for a VSAM data set. The ACB is a control block for entry-sequenced, key-sequenced, and relative record data sets.

**Note:** AMP is supported only for VSAM data sets.

**References:** For more information on AMP and the ACB, see *VSAM Programmer's Guide*.

### Syntax:

```
AMP=(subparameter)
AMP=('subparameter[,subparameter]...')
```

The subparameters are:

```
AMORG
BUFND=number
BUFNI=number
BUFSP=bytes
CROPS={RCK|NCK|NRE|NRC}
OPTCD={I|L|IL}
RECFM={F|FB|V|VB}
STRNO=number
SYNAD=modulename
TRACE
```

**Parentheses:** The subparameter or subparameters are always enclosed in one set of parentheses. For example, AMP=(AMORG).

**Multiple Subparameters:** When the parameter contains more than one subparameter, separate the subparameters by commas and enclose the subparameter list in apostrophes inside the parentheses. For example, AMP=('AMORG,STRNO=4').

**Special Characters:** When the parameter contains only one subparameter and that subparameter contains special characters, enclose the subparameter in apostrophes inside the parentheses. For example, AMP=('STRNO=4').

**Note:** Do not enclose a subparameter in a subparameter list in apostrophes.

**Continuation onto Another Statement:** Enclose the subparameter list in only one set of parentheses. Enclose all the subparameters on each statement in apostrophes. End each statement with a comma after a complete subparameter. For example:

```
//DS1 DD DSNAME=VSAMDATA,AMP=('BUFSP=200,OPTCD=IL,RECFM=FB,'
//      'STRNO=6,TRACE')
```

DD

# DD: AMP

## Subparameter Definition

### AMORG

Indicates that the DD statement defines a VSAM data set. Code AMORG for either of the following reasons:

- When data set access is through an ISAM interface program and the DD statement contains VOLUME and UNIT parameters or contains a DUMMY parameter.
- To open an ACB for a VSAM data set if the data set is not fully defined at the beginning of the job step.

### BUFND = number

Specifies the number of I/O buffers that VSAM is to use for data records. The minimum is 1 plus the STRNO subparameter number. If you omit STRNO, BUFND must be at least 2.

If you omit BUFND from AMP and from the ACB macro instruction, the system uses the STRNO number plus 1.

### BUFNI = number

Specifies the number of I/O buffers that VSAM is to use for index records. If you omit BUFNI from AMP and from the ACB macro instruction, VSAM uses as many index buffers as the STRNO subparameter number; if you omit both BUFNI and STRNO, VSAM uses 1 index buffer.

If data access is through the ISAM interface program, specify for the BUFNI number 1 more than the STRNO number, or specify 2 if you omit STRNO, to simulate having the highest level of an ISAM index resident. Specify a BUFNI number 2 or more greater than the STRNO number to simulate having intermediate levels of the index resident.

### BUFSP = bytes

Specifies the maximum number of bytes for the data and index buffers in the user area.

If BUFSP specifies fewer bytes than the BUFFERSPACE parameter of the access method services DEFINE command, the BUFFERSPACE number overrides the BUFSP number.

### CROPS = {RCK|NCK|NRE|NRC}

Requests a checkpoint/restart option. For more information, see *Checkpoint/Restart*.

### RCK

Requests a data-erase test and data set post-checkpoint modification tests. If the CROPS subparameter is omitted, RCK is the default.



**NCK**

Requests no data set post-checkpoint modification tests.

**NRE**

Requests no data-erase test.

**NRC**

Requests neither a data-erase test nor data set post-checkpoint modification tests.

If you request an inappropriate option, such as the data-erase test for an input data set, the system ignores the option.

**OPTCD = {I|L|IL}**

Indicates how the ISAM interface program is to process records that the step's processing program flags for deletion.

**I**

Requests, when the data control block (DCB) contains OPTCD=L, that the ISAM interface program is not to write into the data set records marked for deletion by the processing program.

If AMP=('OPTCD=I') is specified without OPTCD=L in the DCB, the system ignores deletion flags on records.

**L**

Requests that the ISAM interface program is to keep in the data set records marked for deletion by the processing program.

If records marked for deletion are to be kept but OPTCD=L is not in the DCB, AMP=('OPTCD=L') is required.

*Note:* This parameter has the same meaning and restrictions for the ISAM interface as it has for ISAM. While it was not required in the ISAM job control language, you should code it in the AMP parameter.

**IL**

Requests that the ISAM interface program is not to write into the data set records marked for deletion by the processing program. If the processing program had read the record for update, the ISAM interface program deletes the record from the data set.

AMP=('OPTCD=IL') has the same effect as AMP=('OPTCD=I') coded with OPTCD=L in the DCB.

**RECFM = {F|FB|V|VB}**

Identifies the ISAM record format used by the processing program. You must code this RECFM subparameter when the record format is not specified in the DCB.

*Note:* This parameter has the same meaning and restrictions for the ISAM interface as it has for ISAM. While it was not required in the ISAM job control language, you should code it in the AMP parameter.

## DD: AMP

All VSAM requests are for unblocked records. If the processing program requests blocked records, the ISAM interface program sets the overflow-record indicator for each record to indicate that each is being passed to the program unblocked.

### F

Indicates fixed-length records.

### FB

Indicates blocked fixed-length records.

### V

Indicates variable-length records. If no RECFM is specified in the AMP parameter or in the DCB, V is the default.

### VB

Indicates blocked variable-length records.

### STRNO = number

Indicates the number of request parameter lists the processing program uses concurrently. The number must at least equal the number of BISAM and QISAM requests that the program can issue concurrently. If the program creates subtasks, add together the number of requests for each subtask plus 1 for each subtask that sequentially processes the data set. For details, see *VSAM Programmer's Guide*. exit

### SYNAD = modulename

Names a SYNAD exit routine. The ISAM interface program is to load and exit to this routine if a physical or logical error occurs when the processing program is gaining access to the data set.

The SYNAD parameter overrides a SYNAD exit routine specified in the EXLST or GENCB macro instruction that generates the exit list. The address of the intended exit list is specified in the access method control block that links this DD statement to the processing program. If no SYNAD exit is specified, the system ignores the AMP SYNAD parameter.

### TRACE

Indicates that the generalized trace facility (GTF) executes with your job to gather information about opening and closing data sets and end-of-volume processing. You can use the AMDPRDMP program to print the trace output; see *SPL: Service Aids*.

## Relationship to Other Parameters

Do not code the following parameters with the AMP parameter.

*	DLM	QNAME
BURST	DYNAM	SPACE
CHARS	FCB	SYSOUT
COPIES	FLASH	TERM
DATA	MODIFY	UCS
DCB	OUTPUT	

**Invalid ddnames:** The following ddnames are invalid for VSAM data sets:

JOBLIB  
STEPLIB  
SYSABEND  
SYSUDUMP  
SYSCHK

**Invalid DSNAMES:** When you code the AMP parameter, the DSNNAME must not contain parentheses, a minus (hyphen), or a plus (+) sign. The forms of DSNNAME valid for ISAM, partitioned access method (PAM), and generation data groups (GDG) are invalid with VSAM data sets.

## Buffer Requirements

For a key-sequenced data set, the total minimum buffer requirement is three: two data buffers and one index buffer. For an entry-sequenced data set, two data buffers are required.

If the number of buffers specified in the BUFND and BUFNI subparameters causes the virtual storage requirements to exceed the BUFSP space, the number of buffers is reduced to fit in the BUFSP space.

If BUFSP specifies more space than required by BUFND and BUFNI, the number of buffers is increased to fill the BUFSP space.

## Examples of the AMP Parameter

```
//VSAMDS1 DD DSNNAME=DSM.CLASS,DISP=SHR,AMP=('BUFSP=200,BUFND=2',
//          'BUFNI=3,STRNO=4,SYNAD=ERROR')
```

In this example, the DD statement defines the size of the user area for data and index buffers, specifies the number of data and index buffers, specifies the number of requests that require concurrent data set positioning, and specifies an error exit routine named ERROR.

```
//VSAMDS2 DD DSNNAME=DSM.CLASS,DISP=SHR,AMP=('BUFSP=23456,BUFND=5',
//          'BUFNI=10,STRNO=6,SYNAD=ERROR2,CROPS=NCK,TRACE')
```

In this example, the DD statement defines the values for BUFSP, BUFNI, STRNO, and SYNAD, as in the previous example. It also specifies that a data set post-checkpoint modification test is not to be performed when restarting at a checkpoint and that GTF is to provide a trace.

## DD: BURST

### BURST Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the BURST parameter to specify that 3800 Printing Subsystem output is to go to:

- The burster-trimmer-stacker, to be burst into separate sheets.
- The continuous forms stacker, to be left in continuous fanfold.

If the specified stacker is different from the last stacker used, or if a stacker was not previously requested, JES issues a message to the operator to thread the paper into the required stacker.

**Note:** BURST is valid for any output data set that is printed on a 3800 equipped with a burster-timmer-stacker.

**Syntax:**

```
BURST={ [YES|Y] }  
      { [NO|N] }
```

### Subparameter Definition

#### YES

Requests that the printed output is to be burst into separate sheets. This subparameter can also be coded as Y.

#### NO

Requests that the printed output is to be in a continuous fanfold. This subparameter can also be coded as N.

### Default

If you do not code a BURST parameter, but you code a DD SYSOUT parameter and the output data set is printed on a 3800 that has a burster-timmer-stacker, JES uses an installation default specified at initialization.

If you do not code a BURST parameter or a DD SYSOUT parameter, the default is NO.

### Overrides

A BURST parameter on a sysout DD statement overrides an OUTPUT JCL BURST parameter.

## Relationship to Other Parameters

Do not code the following parameters with the BURST parameter.

*	DLM	MSVGP
AMP	DSID	PROTECT
DATA	DSNAME	QNAME
DDNAME	DYNAM	VOLUME
DISP	LABEL	

## Relationship to Other Control Statements

The burster-trimmer-stacker can also be requested using the following:

- The BURST parameter on the OUTPUT JCL statement. See “BURST Parameter” on page 14-6.
- The STACKER parameter on the JES3 `//*FORMAT PR` statement. See “`//*FORMAT PR Statement`” on page 17-9.
- The BURST parameter on the JES2 `/*OUTPUT` statement. See “`/*OUTPUT Statement`” on page 16-13.

## Example of the BURST Parameter

```
//RECORD DD SYSOUT=A,BURST=Y
```

In this example, the DD statement requests that JES send the output to the burster-trimmer-stacker of the 3800. The stacker separates the printed output into separate sheets instead of stacking it in a continuous fanfold.

## DD: CHARS

### CHARS Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the CHARS parameter to specify the name of one or more character-arrangement tables for printing the data set on a 3800 Printing Subsystem.

**Note:** CHARS is valid for any output data set that is printed on a 3800.

**References:** For further information on character-arrangement tables, see "Requesting Character Arrangements with a 3800 Printer" on page 7-60 and the *IBM 3800 Printing Subsystem Programmer's Guide*. Refer to *System Generation Reference* for information on how to choose during system generation particular groups, other than the Basic group, which is always available.

**Syntax:**

```
          {table-name          }  
CHARS={ (table-name[,table-name]... ) }  
          {DUMP                }  
          {(DUMP[,table-name]... ) }
```

- You can omit the parentheses if you code only one table-name.
- Null positions in the CHARS parameter are invalid. For example, you **cannot** code CHARS=(,table-name) or CHARS=(table-name,,table-name).

### Subparameter Definition

**table-name**

Names a character-arrangement table. Each table-name is 1 to 4 alphanumeric or national characters. Code from one to four names.

**DUMP**

Requests a high-density dump of 204-character print lines from a 3800. If more than one table-name is coded, DUMP must be first.

**Note:**

- DUMP is supported only on JES3 systems.
- DUMP is valid only on a SYSABEND, SYSMDUMP, or SYSUDUMP DD statement.

## Defaults

If you do not code the DD CHARS parameter, JES uses the following, in order:

1. The CHARS parameter on an OUTPUT JCL statement, if referenced by the DD statement.
2. The DD UCS parameter value, if coded.
3. The UCS parameter on an OUTPUT JCL statement, if referenced.

If no character-arrangement table is specified on the DD or OUTPUT JCL statements, JES uses an installation default specified at initialization.

## Overrides

A CHARS parameters on a sysout DD statement overrides the OUTPUT JCL CHARS parameter.

DD

For a data set scheduled to the Print Services Facility (PSF), the PSF uses the following parameters, in override order, to select the font list:

1. Font list in the SYS1.IMAGELIB member specified by an OUTPUT JCL PAGEDEF parameter.
2. DD CHARS parameter.
3. OUTPUT JCL CHARS parameter.
4. DD UCS parameter.
5. OUTPUT JCL UCS parameter.
6. JES installation default for the device.
7. Font list on the PAGEDEF parameter in the PSF cataloged procedure.

See “PAGEDEF Parameter” on page 14-43 for more information.

## Relationship to Other Parameters

Do not code the following parameters with the CHARS parameter.

*	DLM	MSVGP
AMP	DSID	PROTECT
DATA	DSNAME	QNAME
DDNAME	DYNAM	VOLUME
DISP	LABEL	

## DD: CHARS

### Relationship to Other Control Statements

CHARS can also be coded on the following:

- The OUTPUT JCL statement. See "CHARS Parameter" on page 14-7.
- The JES3 `//*FORMAT PR` statement. See "`//*FORMAT PR Statement`" on page 17-9.
- The JES2 `/*OUTPUT` statement. See "`/*OUTPUT Statement`" on page 16-13.

### Printing Device Reassignment

The output device might not be a 3800, for example, if printing were reassigned to a 3211. See the *IBM 3800 Printing Subsystem Programmer's Guide* for restrictions that apply.

### Requesting a High-Density Dump in a JES3 System

You can request a high-density dump on the 3800 in a JES3 system through two parameters on the DD statement for the dump data set or on an OUTPUT JCL statement referenced by the dump DD statement:

- `FCB=STD3`. This parameter produces dump output at 8 lines per inch.
- `CHARS=DUMP`. This parameter produces 204-character print lines.

You can code one or both of these parameters. You can place both on the same statement or one on each statement.

### Examples of the CHARS Parameter

```
//DD1 DD SYSOUT=A,CHARS=(GS10,GU12)
```

In this example, the CHARS parameter specifies two character-arrangement tables to be used when printing the data set. GS10 refers to a Gothic character set, GU12 refers to a Gothic underscored character set.

---

```
//SYSABEND DD UNIT=3800,CHARS=DUMP,FCB=STD3
```

The CHARS parameter on this SYSABEND DD statement specifies a high-density dump with 204 characters per line. The FCB parameter requests the dump output at 8 lines per inch.



## CHKPT Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the CHKPT parameter to request that a checkpoint be written when each end-of-volume is reached on the multivolume data set defined by this DD statement. Checkpoints are written for all volumes except the last. Checkpoints can be requested for input or output data sets.

**Note:** CHKPT is supported only for multivolume QSAM or BSAM data sets. CHKPT is ignored for single-volume QSAM or BSAM data sets or for ISAM, BDAM, BPAM, or VSAM data sets.

**References** For more information, see *Checkpoint/Restart*.

**Syntax:**

```
CHKPT=EOV
```

DD

### Subparameter Definition

**EOV**

Requests a checkpoint at each end-of-volume.

### Overrides

- The RD parameter values of NC and RNC on the JOB or EXEC statements override the CHKPT parameter. For more information, see “RD Parameter” on page 10-28 and “RD Parameter” on page 11-26.
- The CHKPT parameter overrides cataloged procedure values or START command values for checkpoints at end-of-volume.

### Relationship to Other Parameters

Do not code the following parameters with the CHKPT parameter.

*	DLM	QNAME
DATA	DYNAM	SYSOUT
DDNAME	OUTPUT	

### Relationship to the SYSCKEOV DD Statement

If you specify CHKPT, you must also provide a SYSCKEOV DD statement in the job or step. See “SYSCKEOV DD Statement” on page 13-19.

## DD: CHKPT

### Checkpointing Concatenated Data Sets

For concatenated BSAM or QSAM data sets, CHKPT must be coded on each DD statement in the concatenation, if checkpointing is desired for each data set in the concatenation.

### Examples of the CHKPT Parameter

```
//DS1 DD  DSNAME=INDS,DISP=OLD,CHKPT=EOV,  
//      UNIT=SYSSQ,VOLUME=SER=(TAPE01,TAPE02,TAPE03)
```

In this example, the DD statement defines data set INDS, a multivolume QSAM or BSAM data set for which a checkpoint is to be written twice: once when end-of-volume is reached on TAPE01 and once when end-of-volume is reached on TAPE02.

---

```
//DS2 DD  DSNAME=OUTDS,DISP=(NEW,KEEP),  
//      CHKPT=EOV,UNIT=SYSDA,VOLUME=(, , , 8)
```

In this example, OUTDS is a multivolume data set that is being created. The data set requires eight volumes. Seven checkpoints will be written: when the end-of-volume is reached on volumes one through seven.

## CNTL Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the CNTL parameter to reference a CNTL statement that appears earlier in the job. The reference causes the system to execute the program control statements within the referenced CNTL/ENDCNTL group.

The system executes the statements following the first CNTL statement it finds with a label that matches the **label** in the CNTL parameter. If the system finds no match, the system issues an error message.

**References:** For more information on program control statements, see “CNTL Statement” on page 15-5.

**Syntax:**

```
CNTL={*.label                }
      {*.stepname.label      }
      {*.stepname.procstepname.label}
```

### Subparameter Definition

**\*.label**

Identifies an earlier CNTL statement, named label. The system searches for the CNTL statement first in this step, then before the first EXEC statement of the job.

**\*.stepname.label**

Identifies an earlier CNTL statement, named label, that appears in an earlier step, stepname, in the same job.

**\*.stepname.procstepname.label**

Identifies a CNTL statement, named label, in a cataloged or in-stream procedure. Stepname is the name of the job step that calls the procedure; procstepname is the name of the procedure step that contains the CNTL statement named label.

### Examples of the CNTL Parameter

```
//MONDAY DD CNTL=*.WKLYPGM
```

In this example, the DD statement requests that the system use the program control statements following the CNTL statement named WKLYPGM and located earlier in this step or preceding the first step.

```
//TUESDAY DD CNTL=*.SECOND.BLOCKS
```

In this example, the DD statement requests that the system use the program control statements following the CNTL statement named BLOCKS and located in a preceding step named SECOND.

## DD: CNTL

```
//WEDNES DD CNTL=* .THIRD.PROCTWO.CANETTI
```

In this example, the DD statement requests that the system use the program control statements following the CNTL statement named CANETTI and located in the procedure step PROCTWO of the procedure called in the preceding job step THIRD.

## COPIES Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the COPIES parameter to specify how many copies of the output data set are to be printed. The printed output is in page sequence for each copy.

For printing on a 3800 Printing Subsystem, this parameter can instead specify how many copies of each page are to be printed before the next page is printed.

**References:** For more information on the COPIES parameter, see “Requesting Multiple Copies of an Output Data Set Using JES2” and “Requesting Multiple Copies of an Output Data Set Using JES3” on page 7-58.

**Syntax:**

```

      {nnn
COPIES={ (nnn, (group-value[, group-value]...)) }
      { (, (group-value[, group-value]...)) }

```

You can omit the parentheses if you code only COPIES = nnn.

The following are **not** valid:

- A null group-value, for example, COPIES = (5,(,)) or COPIES = (5,)
- A zero group-value, for example, COPIES = (5,(1,0,4))
- A null within a list of group-values, for example, COPIES = (5,(1,,4))

DD

### Subparameter Definition

**nnn**

Specifies how many copies of the data set are to be printed; each copy will be in page sequence order. nnn is a number from 1 through 255 in a JES2 system and from 1 through 254 in a JES3 system

For a data set printed on a 3800, JES ignores nnn if any group-values are specified.

**group-value**

Specifies how many copies of each page are to be printed before the next page is printed. Each group-value is a number from 1 through 255 in a JES2 system and from 1 through 254 in a JES3 system. You can code a maximum of eight group-values. Their sum must not exceed 255 or 254. The total copies of each page equals the sum of the group-values.

*Note:*

- This subparameter is valid only for 3800 output.
- For 3800 output, this subparameter overrides an nnn subparameter, if coded.

# DD: COPIES

## Defaults

If you do not code a COPIES parameter on any of the following, code it incorrectly, or code COPIES=0, the system uses a default of 1, which is the default for the DD COPIES parameter.

DD statement

OUTPUT JCL statement

For JES2, the /\*OUTPUT statement

For JES3, the /\*FORMAT PR or /\*FORMAT PU statement or, if neither is specified, the SYSOUT initialization statement

## Overrides

A COPIES parameter on a sysout DD statement overrides an OUTPUT JCL COPIES parameter.

If this DD statement references an OUTPUT JCL statement and that OUTPUT JCL statement contains a FORMDEF parameter, which specifies a SYS1.IMAGELIB member, the COPYGROUP parameter on a FORMDEF statement in that member overrides any group-value subparameters on the OUTPUT JCL COPIES parameter or the sysout DD COPIES parameter. For more information, see "FORMDEF Parameter" on page 14-31.

## Relationship to Other Parameters

Do not code the following parameters with the COPIES parameter.

*	DISP	LABEL
AMP	DLM	MSVGP
DATA	DSNAME	QNAME
DDNAME	DYNAM	VOLUME

**Relationship to FLASH Parameter:** If this DD statement or a referenced OUTPUT JCL statement also contains a FLASH parameter, JES prints with the forms overlay the number of copies specified in one of the following:

- COPIES = nnn, if the FLASH count is larger than nnn. For example, if COPIES = 10 and FLASH = (LTHD,12) JES prints 10 copies, all with the forms overlay.
- The sum of the group-values specified in the COPIES parameter, if the FLASH count is larger than the sum. For example, if COPIES = (,2,3,4) and FLASH = (LTHD,12) JES prints nine copies in groups, all with the forms overlay.
- The count subparameter in the FLASH parameter, if the FLASH count is smaller than nnn or the sum from the COPIES parameter. For example, if COPIES = 10 and FLASH = (LTHD,7) JES prints seven copies with the forms overlay and three copies without.

**Restriction When Coding UNIT Parameter:** The COPIES parameter is normally coded with the SYSOUT parameter. If, however, both COPIES and UNIT appear on a DD statement, JES handles the COPIES parameter as follows:

- nnn defaults to 1.
- Only the first group-value is used, if group-values are specified and printing is on a 3800.

### Relationship to Other Control Statements

For JES2, if you request copies of the entire job on the JES2 /\*JOBPARM COPIES parameter and also copies of the data set on the DD COPIES or OUTPUT JCL COPIES parameter, and if this is a sysout data set, JES2 prints the number of copies equal to the **product** of the two requests.

The number of copies can also be specified on the COPIES parameter of the following:

- The OUTPUT JCL statement. See “COPIES Parameter” on page 12-25.
- The JES2 /\*OUTPUT statement. See “/\*OUTPUT Statement” on page 16-13.
- The JES3 /\*FORMAT PR statement. See “/\*FORMAT PR Statement” on page 17-9.
- The JES3 /\*FORMAT PU statement. See “/\*FORMAT PU Statement” on page 17-18.

### Examples of the COPIES Parameter

```
//RECORD1 DD SYSOUT=A,COPIES=32
```

This example requests 32 copies of the data set defined by DD statement RECORD1 when printing on an impact printer or a 3800.

---

```
//RECORD2 DD SYSOUT=A,COPIES=(0,(1,2))
```

In this example, when printing on a 3800, three copies of the data set are printed in two groups. The first group contains one copy of each page. The second group contains two copies of each page. When printing on an impact printer, one copy (the default for nnn) is printed.

---

```
//RECORD3 DD SYSOUT=A,COPIES=(8,(1,3,2))
```

In this example, when printing on a 3800, six copies of the data set are printed in three groups. The first group contains one copy of each page, the second group contains three copies of each page, and the last group contains two copies of each page. When the output device is not a 3800, the system prints eight collated copies.

---

```
//RECORD4 DD UNIT=3800,COPIES=(1,(2,3))
```

Because the UNIT parameter is coded and the device is a 3800, the system prints only the first group-value: two copies of each page are printed.

## DD: DATA

### DATA Parameter

**Parameter Type:** Positional, optional

**Purpose:** Use the DATA parameter to begin an in-stream data set that contains statements with // in columns 1 and 2. The data records immediately follow the DD DATA statement; the records must be in BCD or EBCDIC. The data records end when the system reads in the input stream a delimiter:

/\*  
The two-character delimiter specified by a DLM parameter on this DD statement

The data can also end when the input stream runs out of card images.

Note that, unlike a DD \* statement, the data is not ended by the // that indicates another JCL statement.

**Syntax:**

```
//ddname DD DATA[,parameter]...
```

### Defaults

When you do not code DCB=BLKSIZE and DCB=LRECL, JES uses installation defaults specified at initialization.

### Relationship to Other Parameters

**Restrictions in a JES2 system:** For JES2, the only DD parameters that you can code with the DATA parameter follow. All other parameters are a JCL error.

DCB  
DLM  
DSID  
VOLUME

**Restrictions in a JES3 System:** For JES3, the only DD parameters that you can code with the DATA parameter follow. All other parameters are a JCL error.

DCB=BLKSIZE  
DCB=BUFNO  
DCB=LRECL  
DCB=MODE=C  
DLM  
DSID  
VOLUME=SER



*For 3540 Diskette Input/Output Units:* VOLUME=SER, DCB=BUFNO, and DSID parameters on a DD DATA statement are ignored except when they are detected by a diskette reader as a request for an associated data set. See *IBM 3540 Programmer's Reference*. On a DD \* or DD DATA statement processed by a diskette reader, you can specify DSID and VOLUME=SER parameters to indicate that a diskette data set is to be merged into the input stream following the DD statement.

## Relationship to Other Control Statements

Do not refer to an earlier DD DATA statement in DCB or DSNAME parameters on following DD statements.

## Location in the JCL

A DD DATA statement begins an in-stream data set.

*In-stream Data for Cataloged or In-stream Procedures:* A cataloged or in-stream procedure cannot contain a DD DATA statement. When you call a procedure, you can add input stream data to a procedure step by placing in the calling step one or more DD \* or DD DATA statements, each followed by data.

*Multiple In-stream Data Sets for a Step:* You can code more than one DD \* or DD DATA statement in a job step in order to include several distinct groups of data for the processing program. Precede each group with a DD \* or DD DATA statement and follow each group with a delimiter statement. If you omit a DD statement before input data, the system provides a DD \* statement with the ddname of SYSIN; if you omit a following delimiter statement, the system ends the data when it reads a JCL statement or runs out of card images.

## Unread Records

If the processing program does not read all the data in an in-stream data set, the system skips the remaining data without abnormally terminating the step.

## Examples of the DATA Parameter

```
//GROUP1 DD DATA
        .
        data
        .
/*
//GROUP2 DD DATA
        .
        data
        .
/*
```

This example defines two groups of data in the input stream.

## DD: DATA

```
//STEP2      EXEC PROC=UPDATE
//PREP.DD4   DD  DSNAME=A.B.C,VOLUME=SER=D88,
//           UNIT=3350,SPACE=(TRK,(10,5)),
//           DISP=(,CATLG,DELETE)
//PREP.IN1   DD  DATA
            .
            .
            data
            .
/*
//ADD.IN2    DD  *
            .
            .
            data
            .
/*
```

This example defines two groups of data in the input stream. The input defined by DD statement PREP.IN1 is for use by the cataloged procedure step named PREP. This data contains job control statements. The input defined by DD statement ADD.IN2 is for use by the cataloged procedure step named ADD. Because this data is defined by a DD \* statement, it must not contain job control statements.

## DCB Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the DCB parameter to complete during execution the information in the data control block (DCB) for a data set.

The data control block is constructed by the DCB macro instruction in assembler language programs or file definition statements or language-defined defaults in programs in other languages.

**References:** For more information on constructing the data control block, see *Data Management Services Guide*.

**Syntax:**

```
[ DCB=(subparameter [,subparameter]...) ]
[ ( { dsname } ) ]
[ ( { *.ddname } ) ]
[ DCB=( { *.stepname.ddname } [,subparameter]... ) ]
[ ( { *.stepname.procstepname.ddname } ) ]
```

DD

**Parentheses:** You can omit the parentheses if you code:

- Only one keyword subparameter.
- Only a data set name, dsname, without any subparameters.
- Only a backward reference without any subparameters. A backward reference is a reference to an earlier DD statement in the job or in a cataloged or in-stream procedure called by a job step. A backward reference is in the form \*.ddname or \*.stepname.ddname or \*.stepname.procstepname.ddname.

For example, DCB=RECFM=FB or DCB=WKDATA or DCB=\*.STEP3.DD2

**Multiple Subparameters:** When the parameter contains more than one subparameter, separate the subparameters by commas and enclose the subparameter list in parentheses. For example, DCB=(RECFM=FB,LRECL=133,BLKSIZE=399) or DCB=(\*.DD1,BUFNO=4)

**Continuation onto Another Statement:** Enclose the subparameter list in only one set of parentheses. End each statement with a comma after a complete subparameter. For example:

```
//INPUT DD DSN=WKDATA,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,
//          BUFL=800,BUFNO=4)
```

# DD: DCB

## Subparameter Definition

### **subparameter**

Specifies a DCB keyword subparameter needed to complete the data control block. An alphabetic summary of the DCB keyword subparameters follows this parameter description.

### **dsname**

Names a cataloged data set. The system is to copy DCB information from the data set's label. The data set must reside on a direct access volume, and the volume must be mounted before the job step is executed.

The dsname cannot contain special characters, except for periods used in qualifying the name. Do not specify a generation data group (GDG) name.

### **\*.ddname**

Specifies the ddname of an earlier DD statement in the same step. The system is to copy DCB information from the DD statement. The DCB parameter of the referenced DD statement must contain subparameters; it cannot name a cataloged data set or refer to another DD statement.

### **\*.stepname.ddname**

Specifies the ddname of a DD statement in an earlier step, stepname, in the same job. The system is to copy DCB information from the DD statement. The DCB parameter of the referenced DD statement must contain subparameters; it cannot name a cataloged data set or refer to another DD statement.

### **\*.stepname.procstepname.ddname**

Specifies the ddname of a DD statement in a cataloged or in-stream procedure called by an earlier job step. Stepname is the name of the job step that calls the procedure and procstepname is the name of the procedure step that contains the DD statement. The system is to copy DCB information from the DD statement. The DCB parameter of the referenced DD statement must contain subparameters; it cannot name a cataloged data set or refer to another DD statement.

## Defaults

The system obtains DCB information from the following sources, in override order:

- The processing program, that is, the DCB macro instruction in assembler language programs or file definition statements or language-defined defaults in programs in other languages.
- The DCB subparameter of the DD statement.
- The data set label.

Therefore, if you supply information for the same DCB field in your processing program and on a DD statement, the system ignores the DD DCB subparameter. If a DD statement and the data set label supply information for the same DCB field, the system ignores the data set label information.

## Relationship to Other Parameters

Do not code the following parameters with the DCB parameter.

AMP  
DYNAM

With the DDNAME parameter, code **only** the BLKSIZE, BUFNO, and DIAGNS DCB subparameters.

The following are also mutually exclusive:

### DCB

#### Subparameter      Mutually Exclusive With

CPRI	DD parameter OUTLIM
FRID	DD or OUTPUT JCL FCB parameter
FUNC	Data-set-sequence-number of the DD LABEL parameter
THRESH	DD parameter OUTLIM

DD

**Mutually Exclusive DCB Subparameters:** The DCB subparameters CODE, KEYLEN, MODE, PRTSP, STACK, and TRTCH apply to different device types; because only one can apply to a data set, they use the same DCB field. If one of these subparameters is specified on a DD statement for a device different from the type to which it applies, the system interprets the value incorrectly.

DCB subparameters CPRI and THRESH are mutually exclusive.

**For 3540 Diskette Input/Output Units:** The VOLUME =SER, DCB = BUFNO, and DSID parameters on a DD \* or DD DATA statement are ignored except when they are detected by a diskette reader as a request for an associated data set. See *IBM 3540 Programmer's Reference*.

## Completing the Data Control Block

**For Assembler Language Programs:** You must code the DCB macro instruction in a processing program written in assembler language. You can specify some DCB options, particularly those that are different for each execution of the program, on the DD statement DCB parameter, or you can let the system read them from a data set label.

**For Programs in Other Languages:** If your processing program is written in a language other than assembler, DCB options may be (1) specified as part of file definition statements in your program, as DCB subparameters on a DD statement, or in data set label fields, or (2) taken from language-defined default values via the DCB open exit routine. Refer to the programmer's guide for your language to determine how to code DCB options. Refer to *Data Management Services Guide* for a description of the DCB open exit routine.

**DCB Information from the DD DCB Parameter:** Code DCB keyword subparameters to provide the information required to complete the data control block. You must supply DCB options on the DD statement DCB parameter if your processing program, the data set label, or your language's defined values do not complete the data control block.

## DD: DCB

**DCB Information from the Label of a Cataloged Data Set:** Code DCB=dsname to copy the following DCB information from the data set label of a cataloged data set on a currently mounted direct access volume.

DSORG (used in a backward reference)  
RECFM  
OPTCD  
BLKSIZE  
LRECL  
KEYLEN  
RKP

If you do not specify the volume sequence number, system code, and expiration date of the cataloged data set, the system copies them from the data set label.

If you code any DCB subparameters after the dsname, these subparameters override any of the corresponding subparameters in the data set label.

*Mounting of Volume:* A permanently resident volume is the best place from which to copy information, because it is always mounted.

To copy from a volume that is neither permanently resident nor reserved, do one of the following:

- Reference the volume in a job step **before** the step in which you copy the DCB information. This reference will ensure that the DCB information is available.
- If the processing program specifies the RDBACK option in the OPEN macro, code the volume-sequence-number subparameter in the DD VOLUME parameter to make sure that the correct volume is mounted.

**DCB Information From an Earlier DD Statement:** To copy DCB information from an earlier DD statement, code in the DCB parameter one of the following backward references:

\*.ddname  
\*.stepname.ddname  
\*.stepname.procstepname.ddname

If you code any DCB subparameters following the reference, the subparameters override the corresponding subparameters on the referenced DD statement. The system copies from the referenced DD statement only those subparameters not specified on the referencing DD statement.

Do not reference a DD \* or a DD DATA statement.

*Note:* The system also copies the UCS and FCB parameters from the referenced DD statement, unless you override them in the referencing DD statement.

## Examples of the DCB Parameter

```
//DD1  DD      DSNAME=ALP,DISP=(,KEEP),VOLUME=SER=44321,
//          UNIT=3400-6,DCB=(RECFM=FB,LRECL=240,BLKSIZE=960,
//          DEN=1,TRTCH=C)
```

DD statement DD1 defines a new data set named ALP. The DCB parameter contains the information necessary to complete the data control block.

---

```
//DD2  DD      DSNAME=BAL,DISP=OLD,DCB=(RECFM=F,LRECL=80,
//          BLKSIZE=80)
//DD3  DD      DSNAME=CNANN,DISP=(,CATLG,DELETE),UNIT=3400-6,
//          LABEL=(,NL),VOLUME=SER=663488,DCB=*.DD2
```

DD statement DD3 defines a new data set named CNANN and requests that the system copy the DCB subparameters from DD statement DD2, which is in the same job step.

---

```
//DD4  DD      DSNAME=JST,DISP=(NEW,KEEP),UNIT=3350,
//          SPACE=(CYL,(12,2)),DCB=(A.B.C,KEYLEN=8)
```

DD statement DD4 defines a new data set named JST and requests that the system copy the DCB information from the data set label of the cataloged data set named A.B.C. If the data set label contains a key length specification, it is overridden by the KEYLEN coded on this DD statement.

---

```
//DD5  DD      DSNAME=SMAE,DISP=OLD,UNIT=3350,
//          DCB=(*.STEP1.PROCSTP5.DD8,BUFNO=5)
```

DD statement DD5 defines an existing data set named SMAE and requests that the system copy DCB subparameters from DD statement DD8, which is contained in the procedure step named PROCSTP5. The cataloged procedure is called by EXEC statement STEP1. Any of the DCB subparameters coded on DD statement DD8 are ignored if they are specified in the program. If the DCB BUFNO subparameter is not specified in the program, five buffers are assigned.

DD

# DD: DCB

Sub-Parameters	Access Method											Description of Subparameters
	BDAM	BISAM	BFAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM		
BFALN	X	X	X	X		X		X	X			<p>BFALN = {F D}</p> <p>Specifies that each buffer starts either on a word boundary that is not also a doubleword boundary or on a doubleword boundary. If both BFALN and BFTEK are specified, they must be specified from the same source.</p> <p>Default: D (doubleword)</p>
BFTEK	X			X	X				X			<p>BFTEK = R for BDAM and BSAM            BFTEK = D for BTAM            BFTEK = {S E A} for QSAM</p> <p>R Specifies that the data set is being created for or contains variable-length spanned records.            D Specifies that dynamic buffering is to be used in the processing program; if dynamic buffering is specified, a buffer pool must also be defined.            S, E, and A            Specify simple, exchange, or locate mode logical record interface for spanned records. S, E, or A can be coded only when RECFM = VS.</p> <p>If both BFALN and BFTEK are specified, they must be specified from the same source.</p>
BLKSIZE	X		X	X		X		X	X	X		<p>BLKSIZE = number-of-bytes</p> <p>Specifies the maximum length, in bytes, of a block. The maximum is 32760. The number you specify for BLKSIZE depends on the device type and the record format for the data set. For ASCII data sets on magnetic tape, the minimum value for BLKSIZE is 18 bytes and the maximum is 2048 bytes. If you code the BLKSIZE subparameter in the DCB macro instruction or on a DD statement that defines an existing data set with standard labels, the DCB BLKSIZE overrides the block size specified in the label. BLKSIZE can be coded but will have no effect on EXCP processing.</p>
BUFIN										X		<p>BUFIN = number-of-buffers</p> <p>Specifies the number of buffers to be assigned initially for receiving operations for each line in the line group. The combined BUFIN and BUFOUT values must not be greater than the number of buffers in the buffer pool for this line group (not including those for disk activity only).</p> <p>Default: 1</p>
BUFL	X	X	X	X		X		X	X	X		<p>BUFL = number-of-bytes</p> <p>Specifies the length, in bytes, of each buffer in the buffer pool. The maximum is 32760.</p>
BUFMAX										X		<p>BUFMAX = number-of-buffers</p> <p>Specifies the maximum number of buffers to be allocated to a line at one time. Number must be 2 through 15 and must be equal to or greater than the larger of the numbers specified by the BUFIN and BUFOUT subparameters.</p> <p>Default: 2</p>



Sub-parameters	Access Method											Description of Subparameters
	BDAM	BISAM	BFAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM		
BUFNO	X	X	X	X	X	X		X	X			<p>BUFNO = number-of-buffers</p> <p>Specifies the number of buffers to be assigned to the DCB. The maximum normally is 255, but can be less because of the size of the region.</p>
BUFOFF				X						X		<p>BUFOFF = {n L}</p> <p>n Specifies the length, in bytes, of the block prefix used with an ASCII tape data set. For input, n can be 0 through 99. For output, n must be 0 for writing an output data set with fixed-length or undefined-length records.</p> <p>L Specifies that the block prefix is 4 bytes and contains the block length. BUFOFF=L is valid only with RECFM=D. For output, only BUFOFF=L is valid.</p>
BUFOUT											X	<p>BUFOUT = number-of-buffers</p> <p>Specifies the number of buffers to be assigned initially for sending operations for each line in the line group. The combined number of BUFIN and BUFOUT values must not be greater than the number of buffers in the buffer pool for this line group (not including those for disk activity only) and cannot exceed 15.</p> <p>Default: 2</p>
BUFSIZE											X	<p>BUFSIZE = number-of-bytes</p> <p>Specifies the length, in bytes, of each of the buffers to be used for all lines in a particular line group. Length must be 31 through 65535 bytes.</p>
CODE				X	X						X	<p>CODE = {A B C F I N T}</p> <p>Specifies the paper tape code used for punched data. The subparameters CODE, KEYLEN, MODE, PRTPSP, STACK, and TRTCH are mutually exclusive.</p> <p>A ASCII (8 track)  B Burroughs (7 track)  C National Cash Register (8 track)  F Friden (8 track)  I IBM BCD perforated tape transmission code (8 track)  N No conversion required  T Teletype<sup>1</sup> (5 track)</p> <p>Default: I</p>
CPRI											X	<p>CPRI = {R E S}</p> <p>Specifies the relative transmission priority assigned to the lines in this line group.</p> <p>R Specifies that CPU receiving has priority over CPU sending.  E Specifies that receiving and sending have equal priority.  S Specifies that CPU sending has priority over CPU receiving.</p> <p><i>Note:</i> Subparameter CPRI is mutually exclusive with subparameter THRESH and with DD parameter OUTLIM.</p>

<sup>1</sup>Trademark of Teletype Corporation, Skokie, Ill.

Sub-Parameters	Access Method										Description of Subparameters																						
	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM																							
CYLOFL								X			<p>CYLOFL = number-of-tracks</p> <p>Specifies the number of tracks on each cylinder to hold the records that overflow from other tracks on that cylinder. The maximum is 99.</p> <p>Specify CYLOFL only when OPTCD=Y.</p>																						
DEN				X		X				X	<p>DEN = {0 1 2 3 4}</p> <p>Specifies the magnetic density, in number of bytes-per-inch, used to write a magnetic tape data set.</p> <table border="1"> <thead> <tr> <th>DEN</th> <th>7-track tape</th> <th>9-track tape</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>200</td> <td>-</td> </tr> <tr> <td>1</td> <td>556</td> <td>-</td> </tr> <tr> <td>2</td> <td>800</td> <td>800 (NRZI)</td> </tr> <tr> <td>3</td> <td>-</td> <td>1600 (PE)</td> </tr> <tr> <td>4</td> <td>-</td> <td>6250 (GCR)</td> </tr> </tbody> </table> <p>NRZI Non-return-to-zero inverted recording mode.  PE Phase encoded recording mode.  GCR Group coded recording mode.</p> <p>Default: 800 bpi assumed for 7-track tape and 9-track without dual density.  1600 bpi assumed for 9-track with dual density or phase-encoded drives.  6250 bpi assumed for 9-track with 6250/1600 bpi dual density or group coded recording tape.</p>	DEN	7-track tape	9-track tape	0	200	-	1	556	-	2	800	800 (NRZI)	3	-	1600 (PE)	4	-	6250 (GCR)				
DEN	7-track tape	9-track tape																															
0	200	-																															
1	556	-																															
2	800	800 (NRZI)																															
3	-	1600 (PE)																															
4	-	6250 (GCR)																															
DIAGNS	X	X	X	X	X	X	X	X	X	X	<p>DIAGNS = TRACE</p> <p>Specifies the OPEN/CLOSE/EOV trace option, which gives a module-by-module trace of OPEN/CLOSE/EOV's work area and the DCB. If the generalized trace facility (GFT) is not running and tracing user events, DIAGNS is ignored.</p>																						
DSORG	X	X	X	X	X	X	X	X	X	X	<p>DSORG = data-set-organization</p> <p>Specifies the organization of the data set and indicates whether the data set contains any location-dependent information that would make the data set unmovable.</p> <table border="1"> <thead> <tr> <th>Organization</th> <th>Access Method</th> </tr> </thead> <tbody> <tr> <td>PS Physical sequential data set</td> <td>BSAM,EXCP,QSAM,TCAM</td> </tr> <tr> <td>PSU Physical sequential data set that contains location-dependent information</td> <td>BSAM,QSAM,EXCP</td> </tr> <tr> <td>DA Direct access data set</td> <td>BDAM,EXCP</td> </tr> <tr> <td>DAU Direct access data set that contains location-dependent information</td> <td>BDAM,EXCP</td> </tr> <tr> <td>IS Indexed sequential data set</td> <td>BISAM,QISAM,EXCP</td> </tr> <tr> <td>ISU Indexed sequential data set that contains location-dependent information</td> <td>QISAM,EXCP</td> </tr> <tr> <td>PO Partitioned data set</td> <td>BPAM,EXCP</td> </tr> <tr> <td>POU Partitioned data set that contains location-dependent information</td> <td>BPAM,EXCP</td> </tr> <tr> <td>CX Communications line group</td> <td>BTAM</td> </tr> <tr> <td>GS Graphic data control block</td> <td>GAM</td> </tr> </tbody> </table>	Organization	Access Method	PS Physical sequential data set	BSAM,EXCP,QSAM,TCAM	PSU Physical sequential data set that contains location-dependent information	BSAM,QSAM,EXCP	DA Direct access data set	BDAM,EXCP	DAU Direct access data set that contains location-dependent information	BDAM,EXCP	IS Indexed sequential data set	BISAM,QISAM,EXCP	ISU Indexed sequential data set that contains location-dependent information	QISAM,EXCP	PO Partitioned data set	BPAM,EXCP	POU Partitioned data set that contains location-dependent information	BPAM,EXCP	CX Communications line group	BTAM	GS Graphic data control block	GAM
Organization	Access Method																																
PS Physical sequential data set	BSAM,EXCP,QSAM,TCAM																																
PSU Physical sequential data set that contains location-dependent information	BSAM,QSAM,EXCP																																
DA Direct access data set	BDAM,EXCP																																
DAU Direct access data set that contains location-dependent information	BDAM,EXCP																																
IS Indexed sequential data set	BISAM,QISAM,EXCP																																
ISU Indexed sequential data set that contains location-dependent information	QISAM,EXCP																																
PO Partitioned data set	BPAM,EXCP																																
POU Partitioned data set that contains location-dependent information	BPAM,EXCP																																
CX Communications line group	BTAM																																
GS Graphic data control block	GAM																																

Sub-Parameters	Access Method										Description of Subparameters
	BDAM	BISAM	BFAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	
EROPT					X				X		EROPT = n BTAM: Requests the BTAM on-line terminal test option. n = T QSAM: Specifies the option to be executed if an error occurs in reading or writing a record. n = ACC System is to accept the block causing the error. SKP System is to skip the block causing the error. ABE System is to cause abnormal end of task. Default: ABE
FRID				X							FRID = identifier  Specifies a 1- to 4-character load module name identifying the first format record of the 3886 Optical Character Reader data set. FRID is mutually exclusive with the DD or OUTPUT JCL FCB parameter.
FUNC				X					X		FUNC = {I R P W D X T}  Specifies the type of data set to be opened for a 3505 Card Reader or 3525 Card Punch. Unpredictable results will occur if coded for other than a 3505 or 3525.  <i>Note:</i> Subparameter FUNC is mutually exclusive with the data-set-sequence-number of the DD LABEL parameter. I Data set is for punching and printing cards. R Data set is for reading cards. P Data set is for punching cards. W Data set is for printing. D Protected data set is for punching. X Data set is for both punching and printing. T Two-line print option.  Default: P, for output data set. R, for input data set.  The only valid combinations of these values are: I        WT        RWT        RPWXT        PWX R        RP        RPW        RPWD        RPWX P        RPD        PWXT        RWX        RWX W        RW        RPW        RWXT
GNCP						X					GNCP = number-of-channel-programs  Specifies the maximum number of I/O macro instructions that the program will issue before a WAIT macro instruction.
INTVL									X		INTVL = {integer 0}  Specifies the interval, in seconds, between passes through an invitation list.  Default: 0
IPLTXID									X		IPLTXID = member-name  Specifies the member name of the partitioned data set that you want loaded into a 3704/3705 Communications Controller. The DCB IPLTXID subparameter overrides IPLTXID in the TERMINAL macro representing the NCP.

DD

# DD: DCB

Sub-Parameters	Access Method										Description of Subparameters
	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	
KEYLEN	X		X	X		X		X		X	<p>KEYLEN = number-of-bytes</p> <p>Specifies the length, in bytes, of the keys used in a data set. The number is from 1 through 255. KEYLEN=0 specified in the DCB parameter is ignored. For an existing data set, the key length can be supplied from the data set label. If a key length is not specified or supplied, input or output requests must not require keys. The subparameters CODE, KEYLEN, MODE, PRTSP, STACK, and TRTCH are mutually exclusive.</p>
LIMCT	X										<p>LIMCT = number-of-blocks-or-tracks</p> <p>Specifies how many blocks (if relative block addressing is used) or how many tracks (if relative track addressing is used) are to be searched for a free block or available space. This kind of search occurs only when DCB OPTCD=E is also specified; otherwise, LIMCT is ignored. If the LIMCT number equals or exceeds the number of blocks or tracks in the data set, the entire data set is searched.</p>
LRECL			X	X		X		X	X	X	<p>LRECL = number-of-bytes</p> <p>Specifies (1) the length, in bytes, for fixed-length records or (2) the maximum length, in bytes, for variable-length records. When the DCB RECFM is F or U, the length must not exceed the DCB BLKSIZE. For RECFM=D or V, the length must not exceed BLKSIZE minus 4. For RECFM=VS, the length can exceed BLKSIZE. For unblocked records when DCB RKP=0, the length is for only the data portion of the record.</p> <p>LRECL = nnnnnK</p> <p>Specifies the length in kilobytes for variable-length spanned records in ISO/ANSI/FIPS Version 3 tape data sets that are processed the Data Facility Product using the extended logical record interface (XLRI). nnnnn is from 1 through 16383 and indicates multiples of 1024 bytes. The value in the DCB must be LRECL=0K or LRECL=nnnnnK. If a K is coded for any other type of data set, only the numeric value of LRECL is recognized.</p> <p>QSAM: LRECL=X</p> <p>Specifies that the logical record length exceeds 32760 bytes for variable-length spanned records. This option is not valid for ISO/ANSI/FIPS Version 3 variable-length records.</p>

Access Method Sub-Parameters	BDAM	BISAM	BFAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
MODE				X		X			X		<p>MODE = { C [O] }            MODE = { E [R] }</p> <p>Specifies the mode of operation to be used with a card reader, a card punch, or a card read-punch.</p> <p>C Card image (column binary) mode            E EBCDIC mode            O Optional mark read mode            R Read column eliminate mode</p> <p>If you specify R, you must also specify either C or E. Do not code the MODE subparameter for data entered through the input stream except in a JES3 system. The subparameters CODE, KEYLEN, MODE, PRTSP, STACK, and TRTCH are mutually exclusive.</p> <p>Do not code MODE=C for JES2 or JES3 output.</p> <p>Default: E</p>
NCP		X	X	X							<p>NCP = number-of-channel-programs</p> <p>Specifies the maximum number of READ or WRITE macro instructions that will be issued before a CHECK macro instruction is issued to test for completion of the I/O operation. The maximum number is 99, but may actually be smaller depending on the size of the region or partition. If chained scheduling is used, the number must be greater than 1.</p> <p>Default: 1</p>
NTM								X			<p>NTM = number-of-tracks</p> <p>Specifies the number of tracks to be used for a cylinder index. When the specified number of tracks has been filled, a master index is created. The DCB NTM is needed only when the DCB OPTCB=M. If you specify OPTCD=M but omit NTM, the master index option is ignored.</p>

# DD: DCB

Access Method Sub-Parameters	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
OPTCD	X	X	X	X		X		X	X	X	<p>Specifies the optional services to be performed by the control program. All optional services must be requested in one source, that is, in the data set label of an existing data set, in the DCB macro, or in the DD DCB parameter. However, the processing program can modify the DCB OPTCD field. Code the characters in any order; when coding more than one, do not code commas between the characters.</p> <p style="text-align: center;"> <math display="block">\text{BDAM: OPTCD} = \left\{ \begin{array}{l} \{A\} \\ \{R\}[E][F][W] \end{array} \right\}</math> </p> <p>A indicates that the actual device addresses are to be specified in READ and WRITE macro instructions.</p> <p>R indicates that relative block addresses are to be specified in READ and WRITE macro instructions.</p> <p>E indicates that an extended search (more than one track) is to be performed for a block of available space. LIMCT must also be coded. Do not code LIMCT=0 because it will cause an abnormal termination when a READ or WRITE macro instruction is executed.</p> <p>F indicates that feedback can be requested in READ and WRITE macro instructions and the device is to be identified in the same form as it was presented to the control program.</p> <p>W requests a validity check for write operations on direct access devices.</p> <p style="text-align: center;"> <math display="block">\text{BISAM: OPTCD} = \{L\}[R][W]</math> </p> <p>L requests that the control program delete records that have a first byte of all ones. These records will be deleted when space is required for new records. To use the delete option, the DCB RKP must be greater than zero for fixed-length records and greater than four for variable-length records.</p> <p>R requests that the control program place reorganization criteria information in certain fields of the DCB. The problem program can analyze these statistics to determine when to reorganize the data set.</p> <p>W requests a validity check for write operations on direct access devices.</p> <p>Default: R, whenever the OPTCD subparameter is omitted from all sources.</p> <p style="text-align: center;"> <math display="block">\text{BPAM: OPTCD} = \left\{ \begin{array}{l} \{C\}W\{C\} \\ \{C\}H\{C\} \\ \{C\}W\{C\}H\{C\} \end{array} \right\}</math> </p> <p>C requests chained scheduling.</p> <p>W requests a validity check for write operations on direct access devices.</p> <p>H requests that a partitioned data set being processed and residing on MSS, if opening for input, is to be staged to end of file (EOF) on the virtual DASD. Otherwise, only the directory is staged.</p>

Sub-Parameters	Access Method	BDAM	BISAM	BPAM	ESAM	FTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
OPTCD (continued)												<p>BSAM and QSAM: OPTCD = {B} {T} {U[C]} {C[T][B][U]} {H[Z][B]} {J[C][U]} {W[C][T][B][U]} {Z[C][T][B][U]} {Q[C][T][B]} {Z}</p> <p>B requests that the end-of-volume (EOV) routine disregard the end-of-file (EOF) recognition for magnetic tape. For an input data set on a standard-labeled (SL or AL) tape, the EOV routine treats EOF labels as EOV labels until the volume serial list is exhausted. This option allows SL or AL tapes to be read out of volume sequence or to be concatenated to another tape with the same data set name using one DD statement.</p> <p>C requests chained scheduling.</p> <p>H requests hopper empty exit for optical readers or bypass of DOS checkpoint records.</p> <p>J for a data set to be printed on a 3800 Printing Subsystem, instructs the system that each output data line begins with a print control character followed by a table reference character (TRC). The TRC identifies which character arrangement table in the CHARS parameter is to be used to print the line. Before specifying OPTCD=J, see the <i>IBM 3800 Printing Subsystem Programmer's Guide</i>.</p> <p>Q requests (1) that ASCII tape records in an input data set be converted to EBCDIC code when the input record has been read, or (2) an output record in EBCDIC code be converted to ASCII code before the record is written.</p> <p>T requests user totaling facility. T cannot be specified for a SYSIN or SYSOUT data set.</p> <p>U for 1403 or 3211 Printers with the Universal Character Set (UCS) feature and for the 3800, permits data checks and allows analysis by an appropriate error analysis routine. If U is omitted, data checks are not recognized as errors.</p> <p>U for MSS, requests window processing to reduce the amount of staging space required to process large sequential data sets. The DCB DSORG must be PS, the allocation must be in cylinders, and the type of I/O accessing must be INPUT only or OUTPUT only.</p> <p>W requests a validity check for write operations on direct access devices.</p> <p>Z for magnetic tape input, requests that the control program shorten its normal error recovery procedure. When specified, a data check is considered permanent after five unsuccessful attempts to read a record.</p> <p>Z for direct access storage device input, specifies search direct (SD) for sequential data sets.</p> <p>Z for direct access input, specifies the search direct technique.</p> <p>OPTCD=Z is ignored if chained scheduling is used.</p> <p>EXCP: OPTCD=Z</p> <p>Z for magnetic tape input, requests that the control program shorten its normal error recovery procedure. When specified, a data check is considered permanent after five unsuccessful attempts to read a record.</p> <p>Z for direct access storage device input, specifies search direct (SD) for sequential data sets.</p>

DD

Access Method Sub-Parameters	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
OPTCD (continued)											<p>QISAM: OPTCD = {[I][L][M][R][U][W][Y]}</p> <p>I requests that ISAM use the independent overflow areas for overflow records.</p> <p>L requests that ISAM delete records that have a first byte of all ones. These records can be deleted when space is required for new records. To use the delete option, the DCB RKP must be greater than zero for fixed-length records and greater than four for variable-length records.</p> <p>M requests that the system create and maintain one or more master indexes, according to the number of tracks specified in the DCB NTM subparameter.</p> <p>R requests that the control program place reorganization criteria information in the DCB. The problem program can analyze these statistics to determine when to reorganize the data set.</p> <p>U requests that the system accumulate track index entries in storage and write them as a group for each track of the track index. U can be specified only for fixed-length records.</p> <p>W requests a validity check for write operations on direct access devices.</p> <p>Y requests that the system use the cylinder overflow areas for overflow records.</p> <p>Default: R, whenever the OPTCD subparameter is omitted from all sources.</p> <hr/> <p>TCAM: OPTCD = {C U W}</p> <p>C specifies that one byte of the work area indicates if a segment of a message is the first, middle, or last segment.</p> <p>U specifies that the work unit is a message. If U is omitted, the work unit is assumed to be a record.</p> <p>W specifies that the name of each message source is to be placed in an 8-byte field in the work area.</p>
PCI										X	<p>PCI = {[N][,N]}                      {[R][,R]}                      {[A][,A]}                      {[X][,X]}</p> <p>Specifies (1) whether or not a program-controlled interruption (PCI) is to be used to control the allocation and freeing of buffers and (2) how these operations are to be performed. The first operand applies to receiving operations and the second to sending operations.</p> <p>N specifies that no PCIs are taken while filling buffers during receiving operations or emptying buffers during sending operations.</p> <p>R specifies that after the first buffer is filled or emptied, a PCI occurs during the filling or emptying of each succeeding buffer. The completed buffer is freed, but no new buffer is allocated to take its place.</p> <p>A specifies that after the first buffer is filled or emptied, a PCI occurs during the filling or emptying of the next buffer. The first buffer is freed, and a buffer is allocated to take its place.</p> <p>X specifies that after a buffer is filled or emptied, a PCI occurs during the filling or emptying of the next buffer. The first buffer is not freed, but a new buffer is allocated.</p> <p>You can omit the parentheses if you code only the first operand.</p> <p>Default: (A,A)</p>



Access Method Sub-Parameters	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
PRTSP				X		X			X		<p>PRTSP = {0 1 2 3}</p> <p>Specifies the line spacing for an online printer. PRTSP is valid only for an online printer and only if the DCB RECFM is not A or M. PRTSP=2 is ignored if specified with the DD SYSOUT parameter. The subparameters CODE, KEYLEN, MODE, PRTSP, STACK, and TRTCH are mutually exclusive.</p> <p>0 spacing is suppressed  1 single spacing  2 double spacing  3 triple spacing</p> <p>Default: 1</p> <p>JES2 ignores PRTSP for SYSOUT data sets.</p>
RECFM	X	X	X		X			X	X	X	<p>Specifies the format and characteristics of the records in the data set. All the format and characteristics must be completely described in one source, that is, in the data set label of an existing data set, in the DCB macro, or in the DD DCB parameter. However, the processing program can modify the DCB RECFM field.</p> <p style="text-align: center;">{ U }  { V[S] }  BDAM: RECFM = { [BS] }  { F[T] }</p> <p>U indicates that the records are of undefined length.  V indicates that the records are of variable length.  VS indicates that the records are of variable length and spanned.  VBS indicates that the records are of variable length, blocked, and spanned, and that the problem program must block and segment the records.  F indicates that the records are of fixed length.  T indicates that the records may be written using the track-overflow feature.</p> <p>Default: undefined-length, unblocked records.</p> <p style="text-align: center;">{ U [T] [A] }  { [M] }  { }  { V [B] [A] }  { [T] [M] }  { [BT] }  { }  { F [B] [A] }  { [T] [M] }  { [BT] }</p> <p>A indicates that the record contains ISO/ANSI control characters.  B indicates that the records are blocked.  F indicates that the records are of fixed length.  M indicates that the records contain machine code control characters.  T indicates that the records may be written using the track-overflow feature. Chained scheduling (OPTCD=C) will be ignored.  U indicates that the records are of undefined length.  V indicates that the records are of variable length.</p> <p>Default: U</p>

DD

Access Method Sub-Parameters	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
RECFM (continued)											<p style="text-align: right;">           { U [T] [A] }            { [M] }            { }            { F [B ] [A] }            { [S ] [M] }            { [T ] }            { [BS ] }  <b>BSAM,EXCP,&amp; QSAM: RECFM =</b> { [BT ] }            { [BST] }            { }            { V [B ] [A] }            { [S ] [M] }            { [T ] }            { [BS ] }            { [BT ] }            { [BST] }         </p> <p>For BSAM, EXCP, and QSAM using ISO/ANSI/FIPS data sets on tape:</p> <p style="text-align: center;">           RECFM = { D [B] S }            { D [B] [A] }            { U [A] }            { F [B] [A] }         </p> <p>A or M cannot be specified if the PRTSP subparameter is specified.</p> <p>A indicates that the record contains ISO/ANSI device control characters.          B indicates that the records are blocked.          D indicates that the records are variable-length ISO/ANSI tape records.          F indicates that the records are of fixed length.          M indicates that the records contain machine code control characters.          S (1) For fixed-length records, indicates that the records are written as standard blocks, that is, no truncated blocks or unfilled tracks within the data set, with the exception of the last block or track. (2) For variable-length records, indicates that a record can span more than one block.          T indicates that the records can be written using the track-overflow feature, if required. Chained scheduling (OPTCD=C) is ignored.          U indicates that the records are of undefined length. U is invalid for an ISO/ANSI/FIPS Version 3 tape data set.          V indicates that the records are of variable length. V cannot be specified for (1) a variable-length ISO/ANSI tape data set (specify D for this data set), (2) a card reader data set, or (3) a 7-track tape unless the data conversion feature (TRTCH=C) is used.</p> <p>Default: U for IBM standard label tapes.</p> <p><b>QISAM: RECFM =</b> { V[B] }            { F[B] }</p> <p>B indicates that the records are blocked.          F indicates that the records are of fixed length.          V indicates that the records are of variable length; variable records cannot be in ASCII.</p> <p>When creating indexed sequential data sets, you can code the RECFM subparameter; when processing existing indexed sequential data sets, you must omit RECFM.</p> <p>Default: V</p>

Access Method Sub-Parameters	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
RECFM (continued)											TCAM: RECFM = { U } { V[B] } { F }  B indicates that the records are blocked. F indicates that the records are of fixed length. U indicates that the records are of undefined length. V indicates that the records are of variable length.  Default: U
RESERVE										X	RESERVE=(number1,number2)  Specifies the number of bytes (0 through 255) to be reserved in a buffer for insertion of data by the DATETIME and SEQUENCE macros. number1 indicates the number of bytes to be reserved in the first buffer that receives an incoming message. number2 indicates the number of bytes to be reserved in all the buffers following the first buffer in a multiple-buffer header situation.  Default: (0,0)
RKP						X				X	RKP=number  Specifies the position of the first byte of the record key in each logical record. The first byte of a logical record is position 0. If RKP=0 is specified for blocked fixed-length records, the key begins in the first byte of each record. OPTCD=L must not be specified. If RKP=0 is specified for unblocked fixed-length records, the key is not written in the data field. OPTCD=L can be specified. For variable-length records, the relative key position must be 4 or greater, if OPTCD=L is not specified; the relative key position must be 5 or greater, if OPTCD=L is specified.  Default: 0  For EXCP processing, RKP can be coded but is ignored.
STACK				X		X				X	STACK={1 2}  Specifies which stacker bin is to receive a card. The subparameters CODE, KEYLEN, MODE, PRTP, STACK, and TRTCH are mutually exclusive.  Default: 1
THRESH										X	THRESH=number  Specifies the percentage of the nonreusable disk message queue records that are to be used before a flush shutdown occurs.  Default: Shutdown occurs when 95% of the records have been used.  <i>Note:</i> Subparameter THRESH is mutually exclusive with subparameter CPRI and with DD parameter OUTLIM.

# DD: DCB

Access Method Sub-Parameters	BDAM	BISAM	BPAM	BSAM	BTAM	EXCP	GAM	QISAM	QSAM	TCAM	Description of Subparameters
TRTCH				X		X			X		<p>TRTCH = {C E T ET}</p> <p>Specifies the recording technique for 7-track tape. The subparameters CODE, KEYLEN, MODE, PRTSP, STACK, and TRTCH are mutually exclusive.</p> <p>C specifies data conversion, odd parity, and no translation.</p> <p>E specifies no data conversion, even parity, and no translation.</p> <p>T specifies no data conversion, odd parity, and that BCD to EBCDIC translation is required when reading and EBCDIC to BCD translation when writing.</p> <p>ET specifies no data conversion, even parity, and that BCD to EBCDIC translation is required when reading and EBCDIC to BCD translation when writing.</p> <p>Default: no conversion, odd parity, and no translation.</p>

## DDNAME Parameter

*Parameter Type:* Keyword, optional

*Purpose:* Use the DDNAME parameter to postpone defining a data set until later in the same job step. A DDNAME parameter on a DD statement in a cataloged or in-stream procedure allows you to postpone defining the data set until a job step calls the procedure; the data set must be defined in the calling job step.

*References:* For more information on the DDNAME parameter, see “Specifying the DDNAME Parameter” on page 7-1.

*Syntax:*

```
DDNAME=ddname
```

DD

### Subparameter Definition

#### **ddname**

Refers to a later DD statement that defines the data set. The ddname must match the ddname of the defining DD statement.

A job step or procedure step can contain up to five DD statements with DDNAME parameters. Each DDNAME parameter must refer to a different DD statement.

### Relationship to Other Parameters

The **only** DD parameters you can code with the DDNAME parameter are:

```
AMP
DCB = BLKSIZE
DCB = BUFNO
DCB = DIAGNS
```

Do not code the DDNAME parameter on a DD statement with a ddname of JOBLIB, JOBCAT, or STEPCAT.

### Overrides

If any DCB subparameter appears on both DD statements, the DCB subparameter on the referenced DD statement overrides the DCB subparameter on the DD statement that contains DDNAME.

## DD: DDNAME

### Location in the JCL

Place the DD statement referenced in the DDNAME parameter later in the job step or in a cataloged or in-stream procedure called by the job step.

If the referenced data set is to be concatenated with other data sets, the DD statements for the concatenated data sets must immediately follow the DD statement that contains the DDNAME parameter.

**Errors in Location of Referenced DD Statement:** The system treats a DDNAME parameter as though it were a DUMMY parameter and issues a warning message in both of the following cases:

- If the job step or called procedure does not contain the referenced DD statement.
- If the referenced DD statement appears earlier in the job step.

**Location of DD Statement Requesting Unit Affinity:** To use the same device, a DD statement can request unit affinity to an earlier DD statement by specifying UNIT=AFF=ddname.

If a DD statement requests unit affinity to a DD statement containing a DDNAME parameter, the DD statement requesting unit affinity must be placed after the DD statement referenced in the DDNAME parameter. If the DD statement requesting unit affinity appears before, the system treats the DD statement requesting unit affinity as a DUMMY DD statement.

```
//STEP EXEC PGM=TKM
//DD1 DD DDNAME=DD4
//DD2 DD DSNAME=A,DISP=OLD
.
.
//DD4 DD DSNAME=B,DISP=OLD
//DD5 DD UNIT=AFF=DD1
```

DD1 postpones defining the data set until DD4. DD5 requests unit affinity to DD1. Because DD1 is defined when DD5 is processed, the system assigns DD5 to the same device as DD1.

Instead of specifying UNIT=AFF=ddname, both DD statements can specify the same devices in their UNIT parameters or the same volume serials in their VOLUME parameters. For more information on unit affinity, see "Sharing a Unit Between Data Sets on Different Volumes" on page 7-32.

### Referenced DD Statement

If the DDNAME parameter appears in a procedure with multiple steps, the ddname on the referenced DD statement takes the form stepname.ddname. For example, if procedure step STEPCP1 contains:

```
//INDATA DD DDNAME=DD1
```

The referenced DD statement in the calling job step is:

```
//STEPCP1.DD1 DD *
```

The referenced DD statement must not contain a DYNAM parameter.

## Backward References

A backward reference is a reference to an earlier DD statement in the job or in a cataloged or in-stream procedure called by a job step. A backward reference is in the form \*.ddname or \*.stepname.ddname or \*.stepname.procstepname.ddname. The ddname in the reference is the ddname of the earlier DD statement. If the earlier DD statement contains a DDNAME parameter, the reference is to the ddname in the name field of the earlier statement, **not** to the ddname in the DDNAME parameter.

The DD statement referenced in a DDNAME parameter cannot refer to a DD statement between the statement containing the DDNAME parameter and itself. For example:

```
//SHOW EXEC PGM=ABLE
//DD1 DD DDNAME=INPUT
//DD2 DD DSNAME=TEMPSPAC,SPACE=(TRK,1),UNIT=SYSDA
//DD3 DD DSNAME=INCOPY,VOLUME=REF=*.DD1,
// DISP=(,KEEP),SPACE=(TRK,(5,2))
//DD4 DD DSNAME=OUTLIST,DISP=OLD
//DD5 DD DSNAME=MESSAGE,DISP=OLD,UNIT=3330,VOLUME=SER=333333
//INPUT DD DSNAME=NEWLIST,DISP=(OLD,KEEP),VOLUME=SER=333333,
// UNIT=3330
```

The DDNAME parameter on DD1 refers to DD statement INPUT.

The VOLUME parameter of DD3 specifies a backward reference to DD1, which is the ddname in the name field of the referenced statement.

DD statement INPUT identifies the volume 333333 in its VOLUME=SER=333333 parameter. DD statement INPUT cannot use a backward reference to the VOLUME parameter on DD5 because DD5 is between the referring DD1 and the referenced INPUT.

## Examples of the DDNAME Parameter

The following procedure step is the only step in a cataloged procedure named CROWE:

```
//PROCSTEP EXEC PGM=RECPGM
//DD1 DD DDNAME=WKREC
//POD DD DSNAME=OLDREC,DISP=OLD
```

DD statement DD1 is intended for weekly records in the input stream; these records are processed by this step. Because the \* and DATA parameters cannot be used in cataloged procedures, the DDNAME parameter is coded to postpone defining the data set until the procedure is called by a job step. The step that calls the procedure is:

```
//STEPS EXEC PROC=CROWE
//WKREC DD *
.
.
data
.
/*
```

## DD: DDNAME

When the procedure contains multiple steps, use the form `stepname.ddname` for the `ddname` of the referenced DD statement. For example, the following procedure steps appear in a cataloged procedure named `PRICE`:

```
//STEP1 EXEC PGM=SUGAR
//DD1   DD   DDNAME=QUOTES
      .
      .
//STEP2 EXEC PGM=MOLASS
//DD2   DD   DSNAME=WEEKB,DISP=OLD
      .
      .
```

The step that calls the procedure is:

```
//STEPA          EXEC PROC=PRICE
//STEP1.QUOTES  DD   *
              .
              .
              data
              .
/*
```



## DEST Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the DEST parameter to specify a destination for a system output data set. The DEST parameter can send a sysout data set to a remote or local terminal, a node, a node and remote work station, a local device or group of devices, or a terminal at a node.

**Note:** Code the DEST parameter only on a DD statement with a SYSOUT parameter. Otherwise, the system checks the DEST parameter for syntax, then ignores it.

**References:** For more information on the DEST parameter, see “Controlling Output Destination in a JES2 Network” on page 3-7 and “Controlling Output Destination Using JES3” on page 3-12.

### Syntax:

```
DEST=destination
```

The destination subparameter for JES2 is one of the following:

```
LOCAL
name
Nnnnn
NnnRmmmm
NnnnRmmmm
NnnnRmmmm
NnnnnRmm
Rnnnn
RMnnnn
RMTnnnn
Unnn
(node,userid)
```

The destination subparameter for JES3 is one of the following:

```
ANYLOCAL
device-name
device-address
group-name
nodename
(node,userid)
```

### Subparameter Definition for JES2 Systems

#### LOCAL

Indicates any local device.

#### name

Specifies a local or remote device by a symbolic name defined by the installation during JES2 initialization. The name is 1 to 8 alphanumeric or national characters.

## DD: DEST

### **Nnnnn**

Specifies a node. nnnn is 1 to 4 decimal numbers from 1 through 1000.

### **NnnRmmmm**

### **NnnnRmmm**

### **NnnnnRmm**

Specifies a node and a remote work station connected to the node. The node number, indicated in the format by n, is 1 to 4 decimal numbers from 1 through 1000. The remote work station number, indicated in the format by m, is 1 to 4 decimal numbers from 1 through 4000. Do not code leading zeros in n or m. The maximum number of digits for n and m combined cannot exceed six.

*Note:* R0 is equivalent to specifying LOCAL at node Nn.

### **Rnnnn**

### **RMnnnn**

### **RMTnnnn**

Specifies a remote terminal. nnnn is 1 to 4 decimal numbers from 1 through 4000.

*Note:* R0 is equivalent to LOCAL.

### **Unnn**

Specifies a local terminal with special routing. nnn is 1 to 3 decimal numbers from 1 through 255.

### **(node,userid)**

Specifies a node and a TSO or VM userid at that node. The node is a symbolic name defined by the installation during initialization statement; node is 1 to 8 alphanumeric or national characters. The userid must be defined at the node; userid for TSO is 1 to 7 alphanumeric or national characters and for VM is 1 to 8 alphanumeric or national characters. Enclose userid in apostrophes when it contains special characters or begins with a number. For example, DEST=(STL,'VM/370') and DEST=(POK,'921PPC').

A userid requires a node; therefore, code DEST=(node,userid). You **cannot** code a userid without a node.

With a **program-name** subparameter in the SYSOUT parameter, DEST=(node) is valid but DEST=(node,userid) is invalid. Therefore, you can code SYSOUT=(A,program-name),DEST=(node).

## Subparameter Definition for JES3 Systems

### **ANYLOCAL**

Indicates any local device that is attached to the global processor.

### **device-name**

Specifies a local device by a symbolic name defined by the installation during JES3 initialization. device-name is 1 to 8 alphanumeric or national characters.

**device-address**

Specifies the 3-character physical device address.

**group-name**

Specifies a group of local devices, an individual remote station, or a group of remote stations by a symbolic name defined by the installation during JES3 initialization. group-name is 1 to 8 alphanumeric or national characters.

**nodename**

Specifies a node by a symbolic name defined by the installation during JES3 initialization. nodename is 1 to 8 alphanumeric or national characters. If the nodename you specify is the same as the node you are working on, JES3 treats the output as though you specified ANYLOCAL.

**(node,userid)**

Specifies a node and a TSO or VM userid at that node. The node is a symbolic name defined by the installation during initialization statement; node is 1 to 8 alphanumeric or national characters. The userid must be defined at the node; userid for TSO is 1 to 7 alphanumeric or national characters and for VM is 1 to 8 alphanumeric or national characters. Enclose userid in apostrophes when it contains special characters or begins with a number. For example, DEST=(STL,'VM/370') and DEST=(POK,'921PPC').

A userid requires a node; therefore, code DEST=(node,userid). You **cannot** code a userid without a node.

With a **program-name** subparameter in the SYSOUT parameter, DEST=(node) is valid but DEST=(node,userid) is invalid. Therefore, you can code SYSOUT=(A,program-name),DEST=(node).

**Defaults**

If no DEST parameter is specified, JES directs the sysout data set to the default destination for the input device from which the job was submitted.

If the specified destination is invalid, the job fails.

**Overrides**

The DEST parameter on the sysout DD statement overrides an OUTPUT JCL DEST parameter.

**Relationship to Other Parameters**

Do not code the following parameters with the DEST parameter.

*	DLM
DATA	DYNAM
DDNAME	QNAME

You must code a SYSOUT parameter on a DD statement with a DEST parameter.

## DD: DEST

### Relationship to Other Control Statements

You can also code an output destination using:

- The OUTPUT JCL statement. See “DEST Parameter” on page 14-23.
- The JES2 /\*OUTPUT and /\*ROUTE control statements. See “/\*OUTPUT Statement” on page 16-13 and “/\*ROUTE Statement” on page 16-24.
- The JES3 /\*MAIN, /\*FORMAT PR, and /\*FORMAT PU control statements. See “/\*FORMAT PR Statement” on page 17-9, “/\*FORMAT PU Statement” on page 17-18, and “/\*MAIN Statement” on page 17-23.

Because DEST=(node,userid) cannot be coded on JES2 or JES3 control statements, you must code it, if needed, on a DD or OUTPUT JCL statement.

### Examples of the DEST Parameter

```
//JOB01 JOB , 'MAE BIRD',MSGCLASS=B
//STEP1 EXEC PGM=INTEREST
//DEBIT DD SYSOUT=A
//CALIF DD SYSOUT=A,DEST=R555
//FLOR DD SYSOUT=A,DEST=(BOCA, '9212U28')
```

In this example, the system sends the sysout data set defined by DD statement DEBIT to the work station that submitted the job, the data set defined by DD statement CALIF to the remote terminal 555, and the data set defined by DD statement FLOR to VM userid 9212U28 at node BOCA.

## DISP Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the DISP parameter to describe the status of a data set to the system and tell the system what to do with the data set after termination of the step or job. You can specify one disposition for normal termination and another for abnormal termination.

**Note:** Disposition of the **data set** is controlled solely by the DISP parameter; disposition of the **volume(s)** on which the data set resides is a function of the volume status when the volume is demounted.

**References:** For more information on the DISP parameter, see “Specifying a Disposition for the Data Set” on page 7-14. For more information on passing data sets, see “Passing a Data Set” on page 7-17. For information about tape data set processing, see *MVS/370 Magnetic Tape Labels and File Structure*.

DD

**Syntax:**

```
{DISP=status                                     }
{DISP=( [status] [,normal-termination-disp] [,abnormal-termination-disp] ) }

      ( [NEW] [,DELETE ] [,DELETE ] )
      ( [OLD] [,KEEP   ] [,KEEP   ] )
DISP=( [SHR] [,PASS   ] [,CATLG  ] )
      ( [MOD] [,CATLG  ] [,UNCATLG] )
      (           [,UNCATLG] )
      (           [,      ] )
```

- You can omit the parentheses if you code only the status subparameter.
- If you omit the status subparameter but code subparameters for normal or abnormal termination disposition, you must code a comma to indicate the absence of NEW.
- If you omit the second subparameter but code the third, you must code a comma to indicate the absence of the second subparameter.

### Subparameter Definition

#### Status Subparameter

##### NEW

Indicates that a new data set is to be created in this step.

##### OLD

Indicates that the data set exists before this step and that this step requires exclusive (unshared) use of the data set.

## DD: DISP

If you specify `DISP=OLD`, the system does not verify the data set name in the header label when processing a tape data set for output if the data set is not protected by RACF or a password or the data set has no expiration date.

### SHR

Indicates that the data set exists before this step and that another job can share it, that is, use it at the same time. This subparameter can also be coded as `SHARE`.

If you specify `DISP=SHR`, the system does not verify the data set name in the header label when processing a tape data set for output if the data set is not protected by RACF or a password or the data set has no expiration date.

### MOD

Indicates one of the following:

- The data set exists before this step and records are to be added to the end of the data set. The data set must be sequential.
- A new data set is to be created in this step.

In either case, `MOD` specifies exclusive (unshared) use of the data set.

When the data set is opened, the read/write mechanism is positioned after the last sequential record for an existing data set or at the beginning for a new data set. For subsequent `OPENs` within the same step, the read/write mechanism is positioned after the last sequential record.

*Note:* You cannot specify `DISP=MOD` to extend ISO/ANSI/FIPS Version 3 tape data sets, unless the ISO/ANSI/FIPS Version 3 label validation installation exit allows the extension. For information on using ISO/ANSI/FIPS Version 3 installation exits, see *MVS/370 Magnetic Tape Labels and File Structure*.

If you specify `DISP=MOD`, the system creates a data set when:

- Volume information is not specified by a `VOLUME=SER` or `VOLUME=REF` parameter and the data set is not cataloged or passed from another job step.
- A DD statement containing `DISP=MOD` refers to another DD statement that makes a nonspecific volume request and one of the following is also true:
  - The `DSNAME` parameters in the two DD statements are not the same.
  - The two DD statements request different areas of the same ISAM data set.
- The system cannot locate volume information. The system treats the DD statement as a nonspecific volume request for a new data set. Then, after it chooses a volume, if the system finds another data set with the same name on that volume, the system will try to allocate a different volume.

In a JES3 system, if you code `DISP=MOD` for a multivolume data set and any of the volumes are JES3-managed, JES3 will not execute the job until all volumes, including scratch volumes being added, are allocated. Such a job will wait on the queue until all volumes are allocated.

**Normal Termination Disposition Subparameter****DELETE**

Indicates that the data set's space on the volume is to be released at the end of this step. The space can be used for other data sets; the data set is not erased from the space.

The data set is deleted only if its retention period or expiration date has passed; otherwise the data set is kept. See the DD EXPDT or RETPD parameters.

If the system retrieves volume information from the catalog because the DD statement does not specify VOLUME = SER or VOLUME = REF, then DELETE implies UNCATALOG. The system deletes the data set and removes its catalog entry.

**KEEP**

Indicates that the data set is to be kept on the volume at the end of this step.

Only KEEP is valid for VSAM data sets. VSAM data sets should not be passed, cataloged, uncataloged, or deleted.

**PASS**

Indicates that the data set is to be passed for use by a subsequent step in the same job.

*Note:* A data set can be passed only within a job.

**CATLG**

Indicates that the data set is to be kept at the end of this step and an entry pointing to the data set is to be placed in the system or user catalog. For CVOL catalogs, any missing index levels are created. For information about the rules for cataloged data set names, refer to *Access Method Services Reference*.

An unopened tape data set is cataloged, unless the request is nonspecific or unless the data set is allocated to a dual-density tape drive and no density is specified.

**UNCATLG**

Indicates that the data set is to be kept at the end of this step. The system is to delete (1) the entry pointing to the data set in the system or user catalog and (2) unneeded indexes, except for the highest level entry.

**Abnormal Termination (Conditional) Disposition Subparameter****DELETE**

Indicates that the data set's space on the volume is to be released if this step abnormally terminates. The space can be used for other data sets; the data set is not erased from the space.

The data set is deleted only if its retention period or expiration date has passed; otherwise the data set is kept. See the DD EXPDT or RETPD parameters. If the data set was being created when the step abnormally terminates, the data set is deleted even though it has an unexpired retention period or expiration date.

If the system retrieves volume information from the catalog because the DD statement does not specify VOLUME = SER or VOLUME = REF, then DELETE implies UNCATALOG. The system deletes the data set and removes its catalog entry.

DD

## DD: DISP

For a cataloged, passed data set, the user catalog is not updated.

### KEEP

Indicates that the data set is to be kept on the volume if this step abnormally terminates.

Only KEEP is valid for VSAM data sets. VSAM data sets should not be passed, cataloged, uncataloged, or deleted.

### CATLG

Indicates that the data set is to be kept if this step abnormally terminates. An entry pointing to the data set is to be placed in the system or user catalog. For CVOL catalogs, any missing index levels are created. For a cataloged, passed data set, the user catalog is not updated.

In certain cases, a passed, not received data set is not cataloged; see "Disposition Processing of Passed Unreceived Data Sets" on page 7-18.

An unopened tape data set is cataloged, unless the request is nonspecific or unless the data set is allocated to a dual-density tape drive and no density is specified.

### UNCATLG

Indicates that the data set is to be kept if this step abnormally terminates. The system is to delete (1) the entry pointing to the data set in the system or user catalog and (2) unneeded indexes, except for the highest level entry.

For a cataloged, passed data set, the user catalog is not updated.

## Defaults

- If you omit the status subparameter, the default is NEW.
- If you omit the normal termination disposition subparameter, the default is DELETE for a NEW data set or KEEP for an existing data set.
- If you omit the abnormal termination disposition subparameter, the default is the disposition specified or implied by the second subparameter.
- If you omit the DISP parameter, the default is a NEW data set with a disposition of DELETE for both normal and abnormal termination disposition. Thus, you can omit the DISP parameter for a data set that is created and deleted during a step.

## Relationship to Other Parameters

Do not code the following parameters with the DISP parameter.

*	DATA	FLASH
BURST	DDNAME	MODIFY
CHARS	DLM	OUTPUT
CHKPT	DYNAM	QNAME
COPIES		SYSOUT



**DISP Parameters with QSAM Data Sets:** You should not code DISP=MOD if the data control block (DCB) specifies RECFM=FBS when using QSAM. If you specify RECFM=FBS in the DCB and a block is shorter than the block size you specified, QSAM assumes that the short block is the last block and starts end-of-file processing. By this action, QSAM can embed short blocks in your data set and so affect the number of records per track.

## Disposition of VSAM Data Sets

Only KEEP is valid for VSAM data sets. VSAM data sets should not be passed, cataloged, uncataloged, or deleted.

## Disposition of Temporary Data Sets

You must specify a normal termination disposition of PASS or DELETE for a temporary data set or a data set with a system-generated name, that is, when a DSNNAME parameter is omitted from the DD statement.

For a temporary data set name, the system ignores any abnormal termination disposition specified in the third subparameter.

## Disposition of Partitioned Data Sets

When you specify DISP=MOD or DISP=NEW for a partitioned data set and you also specify a member name in the DSNNAME parameter, the member name **must** be unique. If the member name is not unique, the system terminates the job.

When you specify DISP=OLD for a partitioned data set and you also specify a member name in the DSNNAME parameter, the member name **need not** be unique. If the member name is not unique, the system replaces the existing member with the new member.

When you specify DISP=MOD for a partitioned data set and you do not specify a member name, the system positions the read/write mechanism at the end of the data set. The system does not make an automatic entry into the directory.

When you specify DISP=MOD for a partitioned data set and you do specify a member name, the system positions the read/write mechanism at the end of the data set. If the member name already exists, the system terminates the job.

## DISP=MOD for a Multivolume Data Set

When you code DISP=MOD and the volume information is for a multivolume data set, normally the first volume(s) will be mounted on the device(s) allocated. Then, if the data set is opened for output, OPEN starts with the last volume. If the number of volumes is more than the number of allocated devices, the system asks the operator to demount the first volume(s) and mount the last. To have the last volume mounted without first mounting and then demounting the first volume(s):

- For DASD, code DEFER in the UNIT parameter or a volume sequence number in the VOLUME parameter. If you code VOLUME=REF, you must also code either DEFER in the UNIT parameter or a volume sequence number in the VOLUME parameter.

## DD: DISP

- For tape, code VOLUME=REF or DEFER in the UNIT parameter or a volume sequence number in the VOLUME parameter.

When you code DISP=MOD for a multivolume tape data set, use the volume count and volume sequence number subparameters of the VOLUME parameter to keep the system from positioning the read/write mechanism after the last record on the last volume. For example:

```
//DDEX1 DD DSNAME=OPER.DATA,DISP=(MOD,KEEP),VOLUME=(,1,2)
```

The volume sequence number of 1 specifies that you want to use the first volume, and the volume count of 2 specifies that the data set requires two volumes.

If you want to extend a cataloged, multivolume data set and have it properly cataloged after it is kept or passed, code the VOLUME and UNIT parameters to make the system use the values in the system catalog to process the data set. The following DD statement shows how to keep and extend a cataloged multivolume data set using the system catalog. Remember that this data set was created with a volume count of 2.

```
//DDEX2 DD DSNAME=OPER.DATA,DISP=(MOD,KEEP),  
// VOLUME=(,,3),UNIT=(,P)
```

The VOLUME parameter references the system catalog for volume information about the data set and increases the maximum number of volumes for OPER.DATA. Because the UNIT parameter requests parallel mounting, the system must allocate the same number of units as the number of volumes in the VOLUME parameter; in this case, 3.

The following is an example of the messages in the job log after the job completes.

```
IEF285I OPER.DATA KEPT  
IEF285I VOL SER NOS= 333001,333002,333003.  
IEF285I OPER.DATA RECATALOGED  
IEF285I VOL SER NOS= 333001,333002,333003.
```

If you do not reference the system catalog when extending cataloged multivolume data sets, the system does not update the system catalog with the newly referenced volumes.

### Examples of the DISP Parameter

```
//DD2 DD DSNAME=FIX,UNIT=3420-1,VOLUME=SER=44889,  
// DISP=(OLD,,DELETE)
```

DD statement DD2 defines an existing data set and implies by the omitted second subparameter that the data set is to be kept if the step terminates normally. The statement requests that the system delete the data set if the step terminates abnormally.

---

```
//STEPA EXEC PGM=FULL
//DD1 DD DSNAME=SWITCH.LEVEL18.GROUP12,UNIT=3350,
// VOLUME=SER=LOCAT3,SPACE=(TRK,(80,15)),DISP=(,PASS)
//STEPB EXEC PGM=CHAR
//DD2 DD DSNAME=XTRA,DISP=OLD
//DD3 DD DSNAME=*.STEPA.DD1,DISP=(OLD,PASS,DELETE)
//STEPC EXEC PGM=TERM
//DD4 DD DSNAME=*.STEPB.DD3,DISP=(OLD,CATLG,DELETE)
```

DD statement DD1 defines a new data set and requests that the data set be passed. If STEPA abnormally terminates, the data set is deleted because it is a new data set and an abnormal termination disposition was not coded.

DD statement DD3 in STEPB receives this passed data set and requests that the data set be passed. If STEPB abnormally terminates, the data set is deleted because of the third subparameter of DELETE.

DD statement DD4 in STEPC receives the passed data set and requests that the data set be cataloged at the end of the step. If STEPC abnormally terminates, the data set is deleted because of the abnormal termination disposition of DELETE.

DD statement DD2 defines an old data set named XTRA. When STEPB terminates, normally or abnormally, this data set is kept.

## DD: DLM

### DLM Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the DLM parameter to specify a delimiter to terminate this in-stream data set. When the DLM parameter assigns a different delimiter, the in-stream data records can include standard delimiters, such as /\* and //, in the data.

**In a JES2 system,** when the DLM delimiter appears on a DD \* statement, either the assigned delimiter or // ends the input data set. When the DLM delimiter appears on a DD DATA statement, only the assigned delimiter ends the input data set.

**In a JES3 system,** when the DLM delimiter appears on either a DD \* or DD DATA statement, only the assigned delimiter ends the input data set.

**Note:** When the DLM delimiter overrides any implied delimiter, you must terminate the data with the DLM characters. Otherwise, the system keeps reading until the reader is empty.

Except for the JES2 /\*SIGNON and /\*SIGNOFF statements, the system does not recognize JES2 and JES3 statements in an input stream between the DLM parameter and the delimiter it assigns. The JES2 /\*SIGNON and /\*SIGNOFF statements are processed by the remote work station regardless of any DLM delimiter.

**Note:** A /\*EOF statement in the input stream acts as a delimiter regardless of the DLM value. A /\*EOF statement is used only to end input to the internal reader; see "Specifying the Internal Reader" on page 7-52.

**Syntax:**

DLM=delimiter
---------------

- If the specified delimiter contains any special characters, enclose it in apostrophes. In this case, a special character is any character that is neither alphanumeric nor national.

Failing to code enclosing apostrophes produces unpredictable results.

- If the delimiter contains an ampersand or an apostrophe, code each ampersand or apostrophe as two consecutive ampersands or apostrophes. Each pair of consecutive ampersands or apostrophes counts as one character.

### Subparameter Definition

**delimiter**

Specifies two characters that indicate the end of this data set in the input stream.

## Default

/\*

If the system finds an error on the DD statement before the DLM parameter, it does not recognize the value assigned as a delimiter. The system reads records until it reads a record beginning with /\* or //.

## Relationship to Other Parameters

The **only** DD parameters you can code with the DLM parameter are:

```
*
DATA
DCB
DSID
VOLUME
```

DD

The DLM parameter has meaning only on statements defining data in the input stream, that is, DD \* and DD DATA statements. If DLM is specified on any other statement, a JCL error message is issued.

## Invalid Delimiters

If the delimiter is not two characters:

- For JES2, if only one character is specified, JES2 uses the installation-defined default. If more than two characters are specified, JES2 terminates the job.
- For JES3, if an incorrect number of characters is coded, JES3 terminates the job.

## Example of the DLM Parameter

```
//DD1 DD *,DLM=AA
      .
      data
      .
AA
```

The DLM parameter assigns the characters AA as the delimiter for the data defined in the input stream by DD statement DD1. For JES2, the characters // would also serve as valid delimiters since a DD \* statement was used. JES3 accepts only the characters specified for the DLM parameter as a terminator for DD \* or DD DATA.

## DD: DSID

### DSID Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the DSID parameter to specify the data set identifier of an input or output data set on a diskette of the 3540 Diskette Input/Output Unit.

An input data set is read from a 3540 diskette by a diskette reader program, and an output data set is written on a 3540 diskette by a diskette writer, which is an external writer. Neither JES2 or JES3 can read or write diskette data sets.

To create an output data set on a 3540 diskette, the DD statement must contain:

- A DSID parameter.
- A SYSOUT parameter that specifies the output class that the diskette writer processes and the name of the diskette writer.

Also, a system command, from the operator or in the input stream, must start the diskette writer before this DD statement is processed.

**References:** For more information about associated data sets, see "Associated Data Sets (3540 Diskette)" on page 7-6 and *OS/VS2 IBM 3540 Programmer's Reference*. External writers are described in *SPL: Job Management*.

**Syntax:**

<pre>DSID={id      }       {(id,[V])}</pre>
You can omit the parentheses if you code only an id.

### Subparameter Definition

**id**

Specifies the data set identifier. The id is 1 to 8 characters. The characters must be alphanumeric, national, a hyphen, or a left bracket. The first character must be alphabetic or national.

**V**

Indicates that the data set label must have been previously verified on a 3741 Data Station/Workstation. This subparameter is required only on a SYSIN DD statement.

## Relationship to Other Parameters

Do not code the following parameters with the DSID parameter.

BURST	FLASH
CHARS	MODIFY
DDNAME	MSVGP
DYNAM	QNAME

**For 3540 Diskette Input/Output Units:** A DSID parameter on a DD \*, DD DATA, or sysout DD statement is ignored except when detected by a diskette reader as a request for an associated data set. See *IBM 3540 Programmer's Reference*.

On a DD \* or DD DATA statement processed by a diskette reader, you can specify DSID, VOLUME=SER, DCB=BUFNO, and DCB=LRECL to indicate that a diskette data set is to be merged into the input stream following the DD statement.

DD

## Example of the DSID Parameter

```
//JOB1      JOB      , ,MSGLEVEL=(1,1)
//STEP      EXEC     PGM=AION
//SYSIN     DD       *,DSID=(ABLE,V),VOLUME=SER=123456,
//          DCB=LRECL=80
//SYSPRINT  DD       SYSOUT=E,DCB=LRECL=128,DSID=BAKER
```

In this example, the SYSIN DD statement indicates that the input is on diskette 123456 in data set ABLE and must have been verified. The output will be written on a diskette in data set BAKER.

## DD: DSNNAME

### DSNNAME Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the DSNNAME parameter to specify the name of a data set. For a new data set, the specified name is assigned to the data set; for an existing data set, the system uses the name to locate the data set.

**References:** For more information on indexed sequential data sets and generation data groups, see "Specifying the DSNNAME Parameter" on page 7-2 and Chapter 8, "Guide to Special Data Sets." Partitioned data sets are described in *Data Management Services Guide*.

**Syntax:**

```
          {dsname                }
          {dsname(member-name)   }
          {dsname(generation-number) }
{DSNAME}={dsname(area-name)     }
{DSN   } {&&dsname                }
          {&&dsname(member-name)  }
          {&&dsname(area-name)    }
          {*.ddname              }
          {*.stepname.ddname     }
          {*.stepname.procstepname.ddname}
          {NULLFILE              }
```

- You can abbreviate DSNNAME as DSN.
- If the data set name begins with a blank character, the system assigns the data set a temporary data set name.
- The system ignores blank characters at the end of a data set name.

**Special Characters:** When a data set name contains special characters not significant to the system, other than periods or hyphens, enclose it in apostrophes. For example, DSNNAME='DS/29'.

Code each apostrophe that is part of the data set name as two consecutive apostrophes. For example, code DAYS'END as DSNNAME='DAYS''END'.

The following special characters are significant to the system. Do not enclose them in apostrophes.

- Ampersands used to identify temporary data sets.
- Parentheses enclosing the member name of a partitioned data set, the area name of an indexed sequential data set, or the generation number of a generation data set.
- The asterisk used in a backward reference.

On a DD statement in a cataloged or in-stream procedure, if the data set name is a symbolic parameter, do not enclose it in apostrophes. If it is enclosed in apostrophes, the system performs correct substitution only if the symbolic parameter enclosed in apostrophes is preceded by a symbolic parameter not enclosed in apostrophes.

The data set name should not contain the 44 special characters (X'04') created by the 12-4-9 multiple punch or any operation that converts the value of characters to X'04'.



## Subparameter Definition

### Permanent Data Sets

Assign a permanent data set either an unqualified or qualified name:

**Unqualified Name:** 1 to 8 alphanumeric or national characters, a hyphen, or a plus zero (+0). The first character must be alphabetic or national.

**Qualified Name:** multiple names joined by periods. Each name is coded like an unqualified name. The maximum length of a qualified data set name is:

- 44 characters, including periods.
- For a generation data group, 35 characters, including periods.
- For an output tape data set, 17 characters, including periods. If longer than 17 characters, only the rightmost 17 characters are written to the tape header label. For more information, see *Tape Labels*.

#### **dsname**

Specifies a data set name.

#### **dsname(member-name)**

Specifies a permanent partitioned data set name and the name of a member within that data set.

#### **dsname(generation-number)**

Specifies the name of a generation data group (GDG) and the generation number (zero or a signed integer) of a generation data set within the GDG.

To retrieve all generations of a generation data group, omit the generation number.

#### **dsname(area-name)**

Specifies the name of a permanent indexed sequential data set and an area of the data set. The area-name is INDEX, PRIME, or OVFLOW.

If you define an indexed sequential data set on only one DD statement, omit the area name or code it as PRIME. For example, DSNNAME = dsname or DSNNAME = dsname(PRIME).

To retrieve an indexed sequential data set, omit the area name.

## DD: DSNNAME

### Temporary Data Sets

A temporary data set is created and deleted within a job. When defining a temporary data set, you can code the DSNNAME parameter or omit it; if omitted, the system will generate a name for the data set.

When coded, the data set name for a temporary data set consists two ampersands (&&) followed by 1 to 8 alphanumeric or national characters, a hyphen, or a plus zero (+0). The first character following the ampersands must be alphabetic or national.

The system generates a qualified name for the temporary data set. The name begins with SYS and includes the job name, the data set name from the DSNNAME parameter, if coded, and other identifying characters. If several jobs enter the system at the same time and contain DD statements with the same temporary data set name or with no data set name, the qualified names generated by the system will not be unique.

*Note:* A single ampersand before a data set name in a cataloged or in-stream procedure signifies a symbolic parameter. However, if no value is assigned to the name on either the EXEC statement that calls the procedure or on a PROC statement in the procedure, the system treats the name as a temporary data set name.

#### **&&dsname**

Specifies the name of a temporary data set.

#### **&&dsname(member-name)**

Specifies the name of a temporary partitioned data set and a member within that data set.

#### **&&dsname(area-name)**

Specifies the name of a temporary indexed sequential data set and an area of the data set. The area name is INDEX, PRIME, or OVFLOW.

If you define an indexed sequential data set on only one DD statement, omit the area name or code it as PRIME. For example, DSNNAME = &&dsname or DSNNAME = &&dsname(PRIME).

To retrieve an indexed sequential data set, omit the area name.

### Backward References

A backward reference is a reference to an earlier DD statement in the job or in a cataloged or in-stream procedure called by a job step. A backward reference can be coded in the DSNNAME parameter to copy a data set name from another DD statement.

When copying the data set name, the system also copies the following from the DD statement:

- Whether or not the data set is a PDS.
- Whether or not the data set is a temporary data set.

**\*.ddname**

Asks the system to copy the data set name from earlier DD statement ddname.

**\*.stepname.ddname**

Asks the system to copy the data set name from DD statement, ddname, in an earlier step, stepname, in the same job.

**\*.stepname.proclistname.ddname**

Asks the system to copy the data set name from a DD statement in a cataloged or in-stream procedure. Stepname is the name of the job step that calls the procedure, proclistname is the name of the procedure step that contains the DD statement, and ddname is the name of the DD statement.

**Dummy Data Sets****NULLFILE**

Specifies a dummy data set. NULLFILE has the same effect as coding the DD DUMMY parameter. See "Defining a Dummy Data Set" on page 8-8.

**Relationship to Other Parameters**

Do not code the following parameters with the DSNAME parameter.

*	DATA	FLASH SYSOUT
BURST	DDNAME	MODIFY
CHARS	DLM	OUTPUT
COPIES	DYNAM	QNAME

**With DD AMP Parameter:** When you code an AMP parameter for a VSAM data set, do not code a DSNAME:

- That contains parentheses, a minus (hyphen), or a plus (+) sign.
- That is in the form for ISAM.
- That is in the form for PAM (partitioned access method).
- That names a generation data group.

**Examples of the DSNAME Parameter**

```
//DD1 DD DSNAME=ALPHA,DISP=(,KEEP),
// UNIT=3420,VOLUME=SER=389984
```

DD statement DD1 defines a new data set and names it ALPHA. DD statements in later job steps or jobs may retrieve this data set by specifying ALPHA in the DSNAME parameter, unit information in the UNIT parameter, and volume information in the VOLUME parameter.

```
//DD2 DD DSNAME=LIB1(PROG12),DISP=(OLD,KEEP),UNIT=3350,
// VOLUME=SER=882234
```

DD statement DD2 retrieves member PROG12 from the partitioned data set named LIB1.

## DD: DSNNAME

```
//DD3 DD DSNNAME=&&WORK,UNIT=3420
```

DD statement DD3 defines a temporary data set. Because the data set is deleted at the end of the job step, the DSNNAME parameter can be omitted. The following example shows why a temporary data set should be named.

---

```
//STEP1 EXEC PGM=CREATE
//DD4 DD DSNNAME=&&ISDATA(PRIME),DISP=(,PASS),UNIT=(3350,2),
// SPACE=(CYL,(10,,2),,CONTIG),VOLUME=SER=33489,
// DCB=DSORG=IS
//STEP2 EXEC PGM=OPER
//DD5 DD DSNNAME=*.STEP1.DD4,DISP=(OLD,DELETE)
```

DD statement DD4 in STEP1 defines a temporary indexed sequential data set named ISDATA. This DD statement defines all of the areas of an indexed sequential data set. DD statement DD5 in STEP2 retrieves the data set by referring to the earlier DD statement that defines the data set. Since the temporary data set is passed when it is defined in STEP1, STEP2 can retrieve the data set.

## The DUMMY Parameter

**Parameter Type:** Positional, optional

**Purpose:** Use the DUMMY parameter to specify that:

- No device or external storage space is to be allocated to the data set.
- No disposition processing is to be performed on the data set.
- For BSAM and QSAM, no input or output operations are to be performed on the data set.

One use of the DUMMY parameter is in testing a program. When testing is finished and you want input or output operations performed on the data set, replace the DD DUMMY statement with a DD statement that fully defines the data set.

Another use of the DUMMY parameter is in a cataloged or in-stream procedure. Code on the DD DUMMY statement all the required parameters. When the procedure is called, code on the DD statement that overrides the DD DUMMY statement a DSNNAME parameter that matches the DSNNAME parameter on the DD DUMMY statement.

**References:** For more information on the DUMMY parameter, see “Defining a Dummy Data Set” on page 8-8.

**Syntax:**

```
//ddname DD DUMMY[,parameter]...
```

All parameters coded on a DD DUMMY statement must be syntactically correct. The system checks their syntax.

### Parameters on DD DUMMY Statements

- Code the DUMMY parameter by itself or follow it with all the parameters you would normally code when defining a data set, except the DDNAME parameter.
- Code the DCB parameter as you normally would. If the program does not supply all the data control block information, make sure that the DCB parameter supplies the missing information.
- Code AMP=AMORG if you are using VSAM.
- If you code either VOLUME=REF=dsname or DCB=dsname with DUMMY, the referenced dsname must be cataloged or passed; otherwise the job is terminated.
- Because no I/O is performed to the dummy data set, the system ignores the UNIT, SPACE, and DISP parameters, if coded.

## DD: DUMMY

### Relationship to Other Parameters

Do not code the following parameters with the DUMMY parameter.

*	DLM
DATA	DYNAM
DDNAME	QNAME

### Relationship to Other Control Statements

**Backward References:** If a later DD statement in a job refers to a DD DUMMY statement when requesting unit affinity (UNIT=AFF=ddname) or volume affinity (VOLUME=REF=\*.stepname.ddname), the system assigns a dummy status to the later DD statement.

**Overriding a Procedure DD Statement:** Coding DUMMY on a DD statement that overrides a DD statement in a procedure does not nullify symbolic parameters on the overridden DD statement. You must assign values to, or nullify, symbolic parameters on the overridden DD statement as described in "Assigning Values to and Nullifying Symbolic Parameters" on page 2-16.

The DSNAME parameter on the overriding DD statement must not specify NULLFILE.

If the overriding DD statement contains a SUBSYS parameter, the system nullifies a DUMMY parameter on the overridden DD statement in the procedure.

**Data Sets Concatenated to Dummy Data Sets:** The system treats data sets concatenated to a DUMMY data set as dummy data sets in that I/O operations are bypassed. However, the system performs disposition processing and allocates devices and storage for any concatenated data sets.

### Relationship to Access Methods

Use one of the following access methods with the DUMMY parameter:

- Basic sequential access method (BSAM).
- Virtual storage access method (VSAM).
- Queued sequential access method (QSAM).
- BDAM load mode (BSAM with MACRF=WL in the data control block).

If you use any other access method, the job is terminated.

### Examples of the DUMMY Parameter

```
//OUTDD1 DD DUMMY,DSNAME=X.X.Z,UNIT=3350,  
//          SPACE=(TRK,(10,2)),DISP=(,CATLG)
```

DD statement OUTDD1 defines a dummy data set. The other parameters coded on the statement are checked for syntax but not used.

---

```
//IN1 DD DUMMY,DCB=(BLKSIZE=800,LRECL=400,RECFM=FB)
```

DD statement IN1 defines a dummy data set. The DCB parameter supplies data control block information not supplied in the program. Without it, the step might be abnormally terminated.

---

```
//IN2 DD DUMMY,DSNAME=ELLN,DISP=OLD,VOLUME=SER=11257,UNIT=3350
```

When calling a cataloged procedure that contains DD statement IN2 in procedure step STEP4, you can nullify the effects of the DUMMY parameter by coding:

```
//STEP4.IN2 DD DSNAME=ELLN
```

---

```
//TAB DD DSNAME=APP.LEV12,DISP=OLD
```

If you call a cataloged procedure that contains DD statement TAB in procedure step STEP1, you can make this DD statement define a dummy data set by coding:

```
//STEP1.TAB DD DUMMY
```

---

```
//MSGs DD SYSOUT=A
```

If you call a cataloged procedure that contains the DD statement MSGS in procedure step LOCK, you can make this DD statement define a dummy data set by coding:

```
//LOCK.MSGS DD DUMMY
```

## DD: DYNAM

### DYNAM Parameter

**Parameter Type:** Positional, optional

**Purpose:** Use the DYNAM parameter to specify that the system can hold a resource in anticipation of reuse. Even when DYNAM is not coded, the system normally holds resources in anticipation of reuse. The DYNAM parameter provides compatibility with older systems.

A DD DYNAM statement is a DUMMY request.

**References:** For further information, see “Dynamically Allocating and Deallocating Data Sets” on page 4-12.

**Syntax:**

```
//ddname DD DYNAM
```

### Relationship to Other Parameters

Do not code any parameters with the DYNAM parameter.

Do not code DYNAM on a DD statement with a ddname that is meaningful to the system; for example, JOBLIB, SYSCHK.

### Relationship to Other Control Statements

- Do not refer to a DD DYNAM statement in a DDNAME parameter.
- To nullify the DYNAM parameter on a DD statement in a cataloged or in-stream procedure, code a SYSOUT or DSNNAME parameter in the overriding DD statement. DSNNAME=NULLFILE does not nullify a DYNAM parameter.
- Do not make a backward reference to a DD DYNAM statement.
- Do not code the DYNAM parameter on the first DD statement for a concatenation.

### Example of the DYNAM Parameter

```
//INPUT DD DYNAM
```

This DD statement increases by one the control value for dynamically allocated resources held for reuse.



## FCB Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the FCB parameter to specify:

- The forms control buffer (FCB) image JES is to use to guide printing of the output data set by a 3211 Printer, 3203 Printer Model 5, 3800 Printing Subsystem, or 4248 Printer, or by a printer supported by systems network architecture (SNA) remote job entry (RJE).
- The carriage control tape JES uses to control printing of the output data set by a 1403 printer or by a printer supported by SNA RJE.
- The data-protection image JES uses to control output by a 3525 Card Punch.

The FCB image specifies how many lines are to be printed per inch and the length of the form. JES loads the image into the printer's forms control buffer. The FCB image is stored in SYS1.IMAGELIB. IBM provides three standard FCB images:

- STD1, which specifies 6 lines per inch on an 8.5-inch-long form.
- STD2, which specifies 6 lines per inch on an 11-inch-long form.
- STD3, which in a JES3 system specifies 8 lines per inch for a dump.

**References:** For more information on the FCB parameter, see "Requesting Forms Control" on page 7-60. For more information on the forms control buffer, see *SPL: Data Management, Programming Support for the IBM 3505 Card Reader and IBM 3525 Card Punch*, or *IBM 3800 Printing Subsystem Programmer's Guide*.

**Syntax:**

```
FCB={ fcb-name          }
      {( fcb-name[ ,ALIGN| ,VERIFY ] }
```

- You can omit the parentheses if you code only the fcb-name.
- Code the fcb-name as STD1 or STD2 only to request the IBM-supplied images.
- Code the fcb-name as STD3 only for a high-density dump in a JES3 system.

### Subparameter Definition

**fcf-name**

Identifies the FCB image. The name is 1 to 4 alphanumeric or national characters and is the last characters of a SYS1.IMAGELIB member name:

- FCB2xxxx member for a 3211, 3203 model 5, or printer supported by SNA.
- FCB3xxxx member for a 3800.
- FCB4xxxx member for a 4248.

## DD: FCB

### ALIGN

Requests that the system ask the operator to check the alignment of the printer forms before the data set is printed.

*Note:*

- ALIGN is ignored for a sysout data set.
- ALIGN is ignored for a data set printed on a 3800. The 3800 does not use the ALIGN subparameter.

### VERIFY

Requests that the system ask the operator to verify that the image displayed on the printer is for the desired FCB image. The operator can also take this opportunity to align the printer forms.

*Note:* VERIFY is ignored for a sysout data set.

## Defaults

If you do not code the FCB parameter, the system checks the FCB image in the printer's forms control buffer; if it is a default image, as indicated by its first byte, JES uses it. If it is not a default image, JES loads the FCB image that is the installation default specified at JES initialization.

## Overrides

An FCB parameter on a sysout DD statement overrides an OUTPUT JCL FCB parameter.

## Relationship to Other Parameters

Do not code the following parameters with the FCB parameter.

*	DLM
AMP	PROTECT
DATA	QNAME
DDNAME	

Do not code the following DCB subparameters with the FCB parameter.

CYLOFL	INTVL
FRID	RKP

For output to the 3525, do not code the SYSOUT parameter and the FCB parameter; the system ignores the FCB parameter.

## Relationship to Other Control Statements

You can also code the FCB parameter on the following:

- The OUTPUT JCL statement. See “FCB Parameter” on page 12-77.
- The JES2 /\*OUTPUT statement. See “/\*OUTPUT Statement” on page 16-13.
- The JES3 /\*FORMAT PR statement. See “/\*FORMAT PR Statement” on page 17-9.

## Defining an FCB Image for a Work Station

When a work station uses a peripheral data set information record (PDIR), the FCB image is defined in the work station. The DD statement FCB fcb-name subparameter must match the FCB name defined in the PDIR work station.

When a work station does not use a PDIR, add an FCB member to SYS1.IMAGELIB. At setup time, JES3 translates the FCB into a set vertical format (SVF).

## Requesting a High-Density Dump in a JES3 System

You can request a high-density dump on the 3800 in a JES3 system through two parameters on the DD statement for the dump data set or on an OUTPUT JCL statement referenced by the dump DD statement:

- FCB=STD3. This parameter produces dump output at 8 lines per inch.
- CHARS=DUMP. This parameter produces 204-character print lines.

You can code one or both of these parameters. You can place both on the same statement or one on each statement.

## Examples of the FCB Parameter

```
//DD1 DD UNIT=3211,FCB=(IMG1,VERIFY)
```

In this example, the DD statement defines an output data set to be printed by a 3211. The FCB parameter requests that the data set be printed under control of the FCB image IMG1 in SYS1.IMAGELIB. Because VERIFY is coded, the system displays the FCB image on the printer before printing the data set.

```
//DD2 DD SYSOUT=A,FCB=IMG2
```

This sysout DD statement specifies output class A. If output class A routes output to a printer having the forms control buffer feature, JES loads the FCB image IMG2 into the forms control buffer. If the printer does not have the forms control buffer feature, the operator receives a message to mount the carriage control tape IMG2 on the printer.

```
//OUTDDS DD UNIT=3211,FCB=(6,ALIGN)
```

In this example, the DD statement defines an output data set to be printed by a 3211. The FCB parameter requests that the data set be printed under control of the FCB image named 6. Because ALIGN is coded, the system issues a message to the operator requesting that the alignment of the printer forms be checked before the data set is printed.

## DD: FCB

```
//PUNCH DD UNIT=3525,FCB=DP2
```

In this example, the DD statement requests output on a 3525. Therefore, the FCB parameter defines the data protection image to be used for the 3525.

---

```
//SYSUDUMP DD SYSOUT=A,FCB=STD3
```

In this example, the DD statement requests that the 3800 print a dump at 8 lines per inch.

## FLASH Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the FLASH parameter to identify the forms overlay to be used in printing the output data set on a 3800 Printing Subsystem and, optionally, to specify the number of copies on which the forms overlay is to be printed.

**Note:** FLASH is valid only for a data set printed on a 3800.

**References:** For more information on the FLASH parameter, see “Requesting Forms Overlay” on page 7-62. For information on forms overlays, see the *Forms Design Reference Guide for the IBM 3800 Printing Subsystem*.

**Syntax:**

```
FLASH={ (overlay-name [ ,count] ) }
       { NONE }
```

The count subparameter is optional. If you omit it, you can omit the parentheses. However, if you omit it, you must not code it as a null; for example, FLASH=(ABCD,) is invalid.

DD

### Subparameter Definition

**overlay-name**

Identifies the forms overlay frame that the operator is to insert into the printer before printing begins. The name is 1 to 4 alphanumeric or national characters.

**count**

Specifies the number, 0 through 255, of copies that JES is to flash with the overlay, beginning with the first copy printed. Code a count of 0 to flash **all** copies.

**NONE**

Suppresses flashing for this data set.

### Defaults

If you do not code a FLASH parameter or specify an installation default at JES2 or JES3 initialization, forms are not flashed.

If you specify an overlay-name without specifying a count or with a count of 0, all copies are flashed. That is, the default for count is 255.

## DD: FLASH

### Overrides

A FLASH parameter on a sysout DD statement overrides an OUTPUT JCL FLASH parameter.

*Note:* A null first subparameter is invalid in a FLASH parameter on a DD statement, but is permitted on an OUTPUT JCL statement.

### Relationship to Other Parameters

Do not code the following parameters with the FLASH parameter.

*	DLM	MSVGP
AMP	DSID	PROTECT
DATA	DSNAME	QNAME
DDNAME	DYNAM	VOLUME
DISP	LABEL	

**Relationship to COPIES Parameter:** If this DD statement or a referenced OUTPUT JCL statement also contains a COPIES parameter, JES prints with the forms overlay the number of copies specified in one of the following:

- COPIES=nnn, if the FLASH count is larger than nnn. For example, if COPIES=10 and FLASH=(LTHD,12) JES prints 10 copies, all with the forms overlay.
- The sum of the group-values specified in the COPIES parameter, if the FLASH count is larger than the sum. For example, if COPIES=(,(2,3,4)) and FLASH=(LTHD,12) JES prints nine copies in groups, all with the forms overlay.
- The count subparameter in the FLASH parameter, if the FLASH count is smaller than nnn or the sum from the COPIES parameter. For example, if COPIES=10 and FLASH=(LTHD,7) JES prints seven copies with the forms overlay and three copies without.

### Relationship to Other Control Statements

FLASH can also be coded on the following:

- The OUTPUT JCL statement. See "FLASH Parameter" on page 14-28.
- The JES3 `//*FORMAT PR` statement. See "`//*FORMAT PR Statement`" on page 17-9.
- The JES2 `/*OUTPUT` statement. See "`/*OUTPUT Statement`" on page 16-13.

### Verification of Forms Overlay Frame

Before printing starts, JES does not verify that the operator inserted the correct forms overlay frame for flashing.

## Printing without Flashing

To print without flashing, specify one of the following:

- FLASH = NONE on the DD or OUTPUT JCL statement.
- Omit the FLASH parameter on all of the statements for the data set and on all JES initialization statements.
- For a sysout data set, omit the FLASH parameter on the DD statement and specify FLASH = (,0) on a referenced OUTPUT JCL statement.

## Example of the FLASH Parameter

```
//DD1 DD  SYSOUT=A,COPIES=10,FLASH=(ABCD,5)
```

In this example, JES issues a message to the operator requesting that the forms-overlay frame named ABCD be inserted into the printer. Then JES prints the first five copies of the data set with the forms-overlay and the last five copies without.

DD

## DD: FREE

### FREE Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the FREE parameter to specify when the system is to deallocate the resources used for the data set defined by the DD statement. These resources can be devices, volumes, or exclusive use of a data set. You can specify unallocation at the end of the job step or when the data set is closed.

Use FREE=CLOSE on a sysout DD statement to make JES print the system output data set before the job is finished.

**Syntax:**

```
FREE={END|CLOSE}
```

### Subparameter Definition

#### END

Requests that the system deallocate the data set at the end of the step.

#### CLOSE

Requests that the system deallocate the data set when it is closed.

### Defaults

If no FREE parameter is specified, the default is END. Also, if the FREE parameter is incorrectly coded, the system substitutes END and issues a warning message.

### Overrides

FREE=CLOSE is ignored:

- For a data set that is a member of a concatenated group.
- When a task in the step abnormally terminates.
- When the data set is referenced by another DD statement in the same or subsequent step.

### Relationship to Other Parameters

Do not code the following parameters with the FREE parameter.

*	DLM
DATA	DYNAM
DDNAME	QNAME

If the DD statement specifies FREE=END and a DISP subparameter of PASS, the data set is not deallocated until the end of the job or until used for a later DD statement with a disposition of other than PASS.



Do not specify FREE=CLOSE on a DD statement with a ddname of JOBCAT, JOBLIB, STEPCAT, or STEPLIB; CLOSE is ignored.

### Relationship to Other Control Statements

If a DD statement requests unit affinity in a UNIT=AFF parameter or volume affinity in a VOLUME=REF parameter with an earlier DD statement, do not code FREE=CLOSE on the earlier statement.

If you code FREE=CLOSE on a sysout DD statement that references an OUTPUT JCL statement containing a GROUPID parameter, JES2 will not group the data sets into one output group. Instead, JES2 produces one copy of the system output data set for each OUTPUT JCL statement that the DD statement references.

### Relationship to the CLOSE Macro Instruction

DD

When FREE=CLOSE is specified for a data set that is opened and closed more than once during a job step:

- The data set is deallocated after it is closed if the assembler CLOSE macro instruction specifies DISP, REWIND, or FREE. Then if the data set is reopened after the system has deallocated it, the job step abnormally terminates, unless the data set is dynamically allocated in the interval.
- The data set is not deallocated until the end of the job step if the assembler CLOSE macro instruction specifies LEAVE or REREAD. Then the data set can be reopened.

### Examples of the FREE Parameter

```
//EA33 DD SYSOUT=D,FREE=CLOSE
```

In this example, the FREE=CLOSE parameter makes JES deallocate this output class D data set when it is closed, rather than at the end of the job step. JES schedules the data set for printing.

---

```
//EA33 DD DSNAME=SYBIL,DISP=OLD,FREE=CLOSE
```

In this example, the FREE=CLOSE parameter makes JES deallocate the data set, dequeue it, and make it available to other jobs as soon as it is closed.

---

## DD: FREE

```
//STEP1 EXEC PGM=ABLE1
//DD1 DD DSNAME=A,DISP=(,PASS),FREE=END
//STEP2 EXEC PGM=ABLE2
//DD2 DD DSNAME=A,DISP=(OLD,CATLG),FREE=END
```

In this example, data set A is passed by STEP1 to STEP2. FREE=END on DD statement DD1 is ignored because the disposition is PASS. FREE=END on DD statement DD2 causes data set A to be deallocated at the end of STEP2, when it is also cataloged.

---

```
//STEP1 EXEC PGM=BAKER1
//DD DD DSNAME=A,DISP=(NEW,PASS),FREE=END
//STEP2 EXEC PGM=BAKER2
```

In this example, data set A is a new data set. Because PASS is specified, FREE=END is ignored and the data set remains allocated.

## HOLD Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the HOLD parameter to tell the system to hold a system output data set until it is released by the system operator. When the data set is ready for processing, notify the system operator to release it via a TSO NOTIFY parameter, a JES2 /\*MESSAGE statement, or a JES3 /\*OPERATOR statement.

A TSO user can specify HOLD= YES in order to retrieve a sysout data set and display it on a terminal.

**Note:** HOLD is supported only for sysout data sets. If HOLD appears on a DD statement that does not contain a SYSOUT parameter, it is ignored.

**References:** For more information on the HOLD parameter, see “Delaying the Writing of an Output Data Set” on page 7-55.

**Syntax:**

```
HOLD= { [ YES | Y ] | [ NO | N ] }
```

### Subparameter Definition

#### YES

Requests that the system hold the system output data set until the data set is released by the system operator. This subparameter can also be coded as Y.

#### NO

Requests that system perform the installation-defined processing for the sysout data set's output class. This subparameter can also be coded as N.

### Defaults

If no HOLD parameter is specified, the default is NO. If the HOLD parameter is incorrectly coded, the system assumes the default of NO and issues a warning message; the job continues.

### Overrides

HOLD=NO is overridden by the deallocation verb of dynamic allocation or the TSO FREE command.

## DD: HOLD

### Relationship to Other Parameters

Do not code the following parameters with the HOLD parameter.

*	DLM
DATA	DYNAM
DDNAME	QNAME

### Relationship to Other Control Statements

Code a NOTIFY parameter on the JOB statement to ask the system to send a message to your TSO userid when job processing is complete. For information on the JOB NOTIFY parameter, see "NOTIFY Parameter" on page 10-18.

JES2 users can use the /\*NOTIFY control statement to direct job notification messages and to override a JOB NOTIFY parameter. For information on the JES2 /\*NOTIFY statement, see "/\*NOTIFY Statement" on page 16-11.

### Example of the HOLD Parameter

```
//JOB01 JOB , 'HAROLD DUQUETTE',MSGLEVEL=1
//STEP1 EXEC PGM=MJCOSCO
//DD1 DD SYSOUT=B,DEST=RMT6,HOLD=YES
```

Sysout data set DD1 from JOB01 is held on a queue until the TSO user at RMT6 asks the system operator to release the data set.

## LABEL Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the LABEL parameter to specify for a tape data set:

- The relative position of the data set on the volume.
- The type and contents of the label or labels for the data set.
- If a password is required to access the data set.
- If the system is to open the data set only for input or output.
- The expiration data or retention period for the data set.

**References:** For more information on the LABEL parameter, see “Specifying the LABEL Parameter” on page 7-7. For details on tape labels, see *Tape Labels*. For details on direct access labels, see *Data Management Services Guide*. For information on protecting a data set with a password, see *SPL: Data Management*.

**Syntax:**

<pre> (                               [ ,SL ]                               ) (                               [ ,SUL ]                              ) (                               [ ,AL ]                               ) (                               [ ,AUL ] [ ,PASSWORD ] [ ,IN ] [ ,RETPD=nnnn ] ) LABEL=( [data-set-sequence-number] [ ,NSL ] [ ,NOPWREAD ] [ ,OUT ] [ ,EXPDT=yyddd ] ) (                               [ ,NL ] [ ,           ] [ ,           ] ) (                               [ ,BLP ]                               ) (                               [ ,LTM ]                               ) (                               [ ,           ]                               ) </pre>
<p>The first four subparameters are positional; the last subparameter is keyword. If you omit any positional subparameters but code a following positional subparameter, indicate each omitted subparameter by a comma.</p> <p>If you specify only the data-set-sequence-number or only the retention period or only the expiration date, you can omit the parentheses. For example, code LABEL = data-set-sequence-number or LABEL = RETPD = nnnn or LABEL = EXPDT = yyddd.</p>

### Subparameter Definition

#### Data-Set-Sequence-Number

##### data-set-sequence-number

Identifies the relative position of a data set on a tape volume. The data-set-sequence-number is 1 through 4 decimal digits. Omit this subparameter or code 0 or 1 to indicate the first data set on the tape volume.

Omit this subparameter for the following:

- Cataloged data sets. The system obtains the data-set-sequence-number from the catalog.

## DD: LABEL

- A DD DSNNAME parameter that requests all members of a generation data group (GDG). The system retrieves the data-set-sequence-number from the catalog.
- A data set passed from a preceding step. The system obtains the data-set-sequence-number from the passing step.

### Label Types

The system does not retain label type information for cataloged data sets; if the label type is not coded in the LABEL parameter for a cataloged data set, the system assumes SL.

For a data set on a direct access device, the system obtains the label type from the DD statement; the label type is not obtained from any other source referred to in the DD statement. Data sets on direct access devices always have standard labels, but can optionally have user labels also.

#### SL

Indicates that a data set has IBM standard labels.

Code only SL or SUL:

- For data sets on direct access devices.
- When referencing an earlier tape volume DD statement. If you specify any other label type, the system copies the label type from the referenced DD statement, overriding the label type on the referencing DD statement.

#### SUL

Indicates that a data set has both IBM standard and user labels.

Code only SL or SUL:

- For data sets on direct access devices.
- When referencing an earlier tape volume DD statement. If you specify any other label type, the system copies the label type from the referenced DD statement, overriding the label type on the referencing DD statement.

Do not code SUL for partitioned or indexed sequential data sets.

#### AL

Indicates that a tape data set has ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels.

*Note:* Do not specify AL or AUL in the LABEL parameter of a SYSCKEOV DD statement.

If you specify AL for a tape generation data set for output, the ending .GnnnnVnn (where n=0 to 9) **will not** appear as part of the file identifier (data set name field) of the HDR1 label. Instead, the data is placed in the generation and version number fields of the HDR1 label.

**AUL**

Indicates that a tape data set has user labels and ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels.

*Note:* Do not specify AL or AUL in the LABEL parameter of a SYSCKEOV DD statement.

**NSL**

Indicates that a tape data set has nonstandard labels.

**NL**

Indicates that a tape data set has no labels.

When retrieving two or more data sets from several NL or BLP tape volumes, concatenate the DD statements and repeat the LABEL parameter on each DD statement.

If you are processing ASCII data on unlabeled tapes, the data control block must specify OPTCD=Q.

**BLP**

Requests that the system bypass label processing for a tape data set.

If the installation did not specify the BLP feature in the reader cataloged procedure, BLP has the same effect as NL.

If you code BLP and the tape volume has labels, a tapemark delimits the data set. To let the system position a tape with labels to the proper data set, code the data-set-sequence-number subparameter; the number must reflect all labels and data sets that precede the desired data set.

Do not specify BLP when the DD DSNAME parameter requests all members of a generation data group (GDG); the system obtains the data-set-sequence-number from the catalog. Therefore, coding BLP might result in incorrect tape positioning.

When retrieving two or more data sets from several NL or BLP tape volumes, concatenate the DD statements and repeat the LABEL parameter on each DD statement.

**LTM**

Indicates that the data set has a leading tapemark.

**Password Protection**

Password protecting data sets requires the following:

- Data set names no longer than 17 characters. MVS retains in the tape label only the rightmost 17 characters of the data set name. Consequently, longer names could be identical in password checks.
- Volumes with IBM standard labels or ISO/ANSI/FIPS Version 3 labels.
- A password assigned in the PASSWORD data set. If a password is not assigned, the system will abnormally terminate a job step when it attempts to open the data set for output, if NOPWREAD is coded, or for input or output, if PASSWORD is coded.

## DD: LABEL

To create a password-protected data set following an existing password-protected data set, you must supply the password of the existing data set. The security indicator must be the same in both the existing and the new data set.

To password-protect a data set on a tape volume containing other data sets, you must password-protect all the data sets on the volume and the security indicators must be the same for all data sets.

To password-protect an existing data set using **PASSWORD** or **NOPWREAD**, open the data set for output the first time it is used during the job step.

### **PASSWORD**

Indicates that a data set cannot be read, changed, deleted, or written to unless the system operator or TSO user supplies the correct password.

### **NOPWREAD**

Indicates that a data set cannot be changed, deleted, or written to unless the system operator or TSO user supplies the correct password. No password is necessary for reading the data set.

## **Input or Output Processing**

### **IN**

Indicates that a BSAM data set opened for **INOUT** or a BDAM data set opened for **UPDAT** is to be read only. The **IN** subparameter overrides the processing option in the assembler **OPEN** macro instruction. Any attempt by the processing program to write in the data set makes the system give control to the error analysis (**SYNAD**) routine.

### **OUT**

Indicates that a BSAM data set opened for **OUTIN** or **OUTINX** is to be written in only. The **OUT** subparameter overrides the processing option in the assembler **OPEN** macro instruction. Any attempt by the processing program to read the data set makes the system give control to the error analysis (**SYNAD**) routine.

## **Expiration Date for Data Set**

If the **DD** statement contains **DISP=(NEW,DELETE)** or the **DISP** parameter is omitted to default to **NEW** and **DELETE**, the system deletes the data set when the step terminates normally or abnormally, even though an **RETPD** period or **EXPDT** date is also specified.

Do not specify or imply **RETPD** or **EXPDT** for a temporary data set.

Expiration dates for successive files on an ISO/ANSI/FIPS Version 3 volume must be coded in **descending** order. For example, a data set with an expiration date of December 7, 1984 (**LABEL = EXPDT = 84342**) should be followed by a data set with an expiration date of December 6, 1984 (**LABEL = EXPDT = 84341**) or earlier.

*Note:* Do not try to protect valuable data with the **RETPD** and **EXPDT** subparameters. Instead, use password protection or the Resource Access Control Facility for data protection.

### **RETPD = nnnn**

Specifies the retention period, in days, for the data set. The **nnnn** is 1 through 4 decimal digits. After **nnnn** days, the data set can be deleted or written over by another data set.



The system adds nnnn to the current date to produce an expiration date. If the calculated date is after January 1, 2000, the system will retain the data set only to January 1, 2000. The calculated expiration date uses 365-day years, ignoring leap years.

**EXPDT = yyddd**

Specifies an expiration date for the data set. The yy is a two-digit year number and the ddd is a three-digit day number from 001 through 366. For example, code February 2, 1986 as EXPDT = 86033.

**Defaults**

- If no data-set-sequence-number subparameter is specified or if the number is coded as 0 or 1, the default is the first data set on the tape volume, unless the data set is passed or cataloged.
- If no label type subparameter is specified, the default is only IBM standard labels (SL).

**Relationship to Other Parameters**

Do not code the following parameters with the LABEL parameter.

*	DATA	FLASH
BURST	DDNAME	MODIFY
CHARS	DLM	OUTPUT
COPIES	DYNAM	QNAME
		SYSOUT

Do not specify the LABEL parameter with the FUNC subparameter of the DCB parameter. The results are unpredictable.

ISO/ANSI/FIPS Version 3 tape data sets can be protected by use of the ACCODE parameter. See "ACCODE Parameter" on page 12-9.

Do not specify AL or AUL in the LABEL parameter of a SYSCKEOV DD statement.

**Deleting a Data Set Before its Expiration Date**

To delete a data set before the expiration date or retention period has passed, use one of the following:

- For data sets cataloged in a VSAM or ICF catalog, use the DELETE command, as described in *Access Method Services Reference*.
- For data sets not cataloged in a VSAM or ICF catalog, use the IEHPROGM utility, as described in *Utilities*.
- For the data set control block (DSCB), use the SCRATCH macro with the OVRD parameter, as described in *SPL: Data Management*. Deletion of the DSCB makes the space occupied by the data set available for reallocation.

## DD: LABEL

### Translation

If the installation specified `ASCII=INCLUDE` during system generation, then `AL` or `AUL` in the `LABEL` parameter requests translation. You can also request translation by specifying `OPTCD=Q` in the data control block. If the tape is not labeled, `LABEL=(,NL)`, you *must* specify `OPTCD=Q` for translation to occur.

### Examples of the LABEL Parameter

```
//DD1 DD DSNAME=HERBI,DISP=(NEW,KEEP),UNIT=TAPE,  
//      VOLUME=SER=T2,LABEL=(3,NSL,RETPD=188)
```

This DD statement defines a new data set. The `LABEL` parameter tells the system:

- This data set is to be the third data set on the tape volume.
- This tape volume has nonstandard labels.
- This data set is to be kept for 188 days.

---

```
//DD2 DD DSNAME=A.B.C,DISP=(,CATLG,DELETE),UNIT=3400-5,  
//      LABEL=(,NL)
```

This DD statement defines a new data set, requests that the system catalog it, and indicates that the data set has no labels. Each time this data set is used by a program, the DD statement must include `LABEL=(,NL)`.

---

```
//DD3 DD DSNAME=SPECS,UNIT=3400-5,VOLUME=SER=10222,  
//      DISP=OLD,LABEL=4
```

This DD statement indicates an existing data set. The `LABEL` parameter indicates that the data set is fourth on the tape volume.

---

```
//STEP1 EXEC PGM=FIV  
//DDX DD DSNAME=CLEAR,DISP=(OLD,PASS),UNIT=3400-5,  
//      VOLUME=SER=1257,LABEL=(,NSL)  
//STEP2 EXEC PGM=BOS  
//DDY DD DSNAME=*.STEP1.DDX,DISP=OLD,LABEL=(,NSL)
```

DD statement `DDX` in `STEP1` indicates an existing data set with nonstandard labels and requests that the system pass the data set. DD statement `DDY` in `STEP2` receives the data set. `DDY` does not contain unit and volume information, because the system obtains this information through the backward reference in the `DSNAME` parameter. `DDY` contains the label type, because the system does not obtain the label type through the backward reference.

## MODIFY Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the MODIFY parameter to specify a copy-modification module that tells JES how to print the output data set on a 3800 Printing Subsystem. The module can specify the following:

- Legends.
- Column headings.
- Where and on which copies the data is to be printed.

The module is defined and stored in SYS1.IMAGELIB using the IEBIMAGE utility program.

**Note:** MODIFY is supported only for the 3800 Printing Subsystem Model 1 and 2 and the 3800 Printing Subsystem Model 3 in compatibility mode.

DD

**References:** For more information on the MODIFY parameter, see “Requesting Copy Modification” on page 7-58, and on the copy modification module and the IEBIMAGE utility program, see the *IBM 3800 Printing Subsystem Programmer’s Guide*.

### Syntax:

```
MODIFY={module-name      }
        {(module-name[,trc])}
```

- You must code the module-name.
- The trc subparameter is optional. If you omit it, you can omit the parentheses. However, if you omit it, you must not code it as a null; for example, MODIFY=(TAB1,) is invalid.

### Subparameter Definition.

#### module-name

Identifies a copy-modification module in SYS1.IMAGELIB. The module-name is 1 through 4 alphanumeric or national characters.

#### trc

Identifies which table-name in the CHARS parameter is to be used. This **table reference character** is 0 for the first table-name specified, 1 for the second, 2 for the third, or 3 for the fourth. The CHARS parameter is on the following, in override order:

1. This DD statement.
2. A referenced OUTPUT JCL statement.
3. A statement in the SYS1.IMAGELIB member specified on the OUTPUT JCL PAGEDEF parameter.

## DD: MODIFY

4. A statement in the SYS1.IMAGELIB member obtained by default.
5. A JES3 initialization statement.

### Defaults

If no MODIFY parameter is specified, JES3 uses an installation default specified at initialization. JES2 provides no installation default at initialization.

If you do not specify **trc** or if the **trc** value is greater than the number of table-names in the CHARS parameter, JES2 uses the first table named in the CHARS parameter and JES3 uses the default character arrangement table.

### Overrides

A MODIFY parameter on a sysout DD statement overrides an OUTPUT JCL MODIFY parameter.

*Note:* A null first subparameter is invalid in a MODIFY parameter on a DD statement, but is permitted on an OUTPUT JCL statement.

### Relationship to Other Parameters

Do not code the following parameters with the MODIFY parameter.

*	DLM	MSVGP
AMP	DSID	PROTECT
DATA	DSNAME	QNAME
DDNAME	DYNAM	VOLUME
DISP	LABEL	

### Relationship to other Control Statements

MODIFY can also be coded on the following:

- The OUTPUT JCL statement. See "MODIFY Parameter" on page 14-41.
- The JES3 **//\*FORMAT PR** statement. See "**//\*FORMAT PR Statement**" on page 17-9.
- The JES2 **/\*OUTPUT** statement. See "**/\*OUTPUT Statement**" on page 16-13.

The second character of each logical record can be a TRC code, so that each record can be printed in a different font. This way of specifying fonts is indicated by the OUTPUT JCL TRC parameter.

**Example of the MODIFY Parameter**

```
//DD1 DD UNIT=3800,MODIFY=(A,0),CHARS=(GS15,GS10)
```

In this example, the **MODIFY** parameter requests that the data in the copy-modification module named **A** replace variable data in the data set to be printed by the 3800. Module **A** defines which positions are to be replaced and which copies are to be modified. The second subparameter in **MODIFY** specifies that the first character arrangement table in the **CHARS** parameter, **GS15**, be used.

## DD: MSVGP

### MSVGP Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the MSVGP parameter to place the data set in a group of mass storage volumes on a mass storage system (MSS) device. MSVGP applies only to a nonspecific volume request, which is a DD statement for a new data set that can be assigned to any volume or volumes in a group.

**References:** For more information on using mass storage volumes, see "Specifying Data Sets for Mass Storage Systems (MSS)" on page 7-37, and on defining mass storage groups, see *Mass Storage System (MSS) Services General Information*.

**Syntax:**

```
MSVGP={ ( id [ , ddname ] ) }  
      { SYSGROUP }
```

You can omit the parentheses if you code only the id subparameter or SYSGROUP.

### Subparameter Definition

**id**

Identifies a group of mass storage volumes. The id is 1 through 8 alphanumeric or national characters and must be previously defined by the installation using a mass storage system service.

**ddname**

Requests that the data set for this DD statement is to be allocated to volume(s) other than the volume(s) occupied by the data set for DD statement ddname. DD statement ddname must appear earlier in the job step.

If volume separation within the group is not possible, the system terminates the job.

The ddname refers to only the first data set (1) when the named DD statement starts a concatenation or (2) when the DSNAMES parameter of the named DD statement requests all members of a generation data group (GDG).

**SYSGROUP**

Identifies a default group of mass storage volumes. Code MSVGP=SYSGROUP to make sure that a nonspecific request is allocated to SYSGROUP.

## Relationship to Other Parameters

Do not code the following parameters with the MSVGP parameter.

*	DDNAME	FLASH
BURST	DLM	MODIFY
CHARS	DSID	OUTPUT
COPIES	DYNAM	QNAME
DATA		SYSOUT

**SPACE Parameter:** Code the SPACE parameter for the following reasons:

- To allocate noncontiguous primary space. Contiguous space is the MSVGP default.
- For nonspecific requests for BPAM and ISAM data sets.

Do not code the ABSTR, MXIG, and ALX subparameters in the SPACE parameter.

**VOLUME Parameter:** Do not code VOLUME=SER with the MSVGP parameter; VOLUME=SER is a specific volume request, while MSVGP can be used only for nonspecific volume requests. Other VOLUME subparameters can be specified with MSVGP.

VOLUME=PRIVATE with MSVGP is redundant, because MSVGP causes allocation to a private volume.

**UNIT and VOLUME Counts:** The unit count in the UNIT parameter must be less than the volume count in the VOLUME parameter to guarantee allocation of a nonsharable unit.

## Allocation when MSVGP is Not Coded

When a DD statement defines a new, nonspecific data set and does not contain an MSVGP parameter, the system allocates as follows:

- Places a permanent data set on a mounted 3330V Disk Storage volume, if one exists. If one does not exist, a volume is selected from SYSGROUP; in this case, the DD statement must contain a SPACE parameter or the job is terminated.
- Places a temporary data set on a mounted 3330V Disk Storage public volume, if one exists. If one does not exist or it does not have enough space, a volume is selected from SYSGROUP.

## DD: MSVGP

### Examples of the MSVGP Parameter

```
//DD1 DD DSNAME=ACCOUNT,DISP=(NEW,CATLG),UNIT=3330V,  
//      MSVGP=AB$1234@,VOLUME=(,,3)
```

A new, cataloged data set is to be created on one, two, or three mass storage volumes in the group called AB\$1234@. The installation previously defined this group and assigned at least three volumes to it. If this service provided a space default of SPACE=(CYL,(200,100)), the system selects from the group a volume with at least 200 cylinders available.

During step execution, if more than 200 cylinders are required, and if 100 more cylinders are not available on the mounted volume, the system asks the operator to demount the volume. The system selects from group AB\$1234@ a volume with at least 100 cylinders available and asks the operator to mount the volume. The volume count of three allows the data set to extend over up to three volumes. If more are required, the step abnormally terminates.

---

```
//DD1 DD DSNAME=MASTRIN,DISP=OLD  
//DD2 DD DSNAME=MASTROUT,UNIT=3330V,DISP=(,CATLG),  
//      MSVGP=(AB$1234@,DD1)
```

DD1 requests an existing cataloged data set. DD2 defines a new data set that will be allocated to a volume in mass storage group AB1234@.

Because DD1 is specified as the ddname subparameter of MSVGP on DD2, the system allocates the DD2 data set, MASTROUT, to different volumes than the volumes for the DD1 data set, MASTRIN.



## OUTLIM Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the OUTLIM parameter to limit the number of logical records in the system output data set defined by this DD statement. When the limit is reached, the system exits to the SYSOUT limit exit routine. If the installation supplies a user-written routine, the routine can determine whether to terminate the job or increase the limit. If the installation does not supply a routine, the system terminates the job.

If the installation supplies a routine, it must be included in the SYS1.LPALIB library.

**Note:** OUTLIM is valid only on a DD statement with a SYSOUT parameter.

**References:** For more information on the OUTLIM parameter, see “Limiting Output Records” on page 7-55, and on the SYSOUT limit exit routine, see *SPL: System Management Facilities*.

**Syntax:**

```
OUTLIM=number
```

### Subparameter Definition

**number**

Specifies the maximum number of logical records. The number is 1 through 8 decimal digits from 1 through 16777215.

### Default

(1) If no OUTLIM parameter is specified or OUTLIM=0 is coded and (2) if output is not limited by JES control statements, JES3 uses an installation default specified at initialization; JES2 provides no installation default at initialization.

### Relationship to Other Parameters

OUTLIM can be coded only when SYSOUT is also coded.

Do not code the following parameters with the OUTLIM parameter.

*	DISP	MSVGP
AMP	DLM	PROTECT
CHKPT	DSNAME	QNAME
DATA	DYNAM	SUBSYS
DDNAME	LABEL	

Do not code the OUTLIM parameter with the DCB subparameters CPRI or THRESH; these subparameters can alter the OUTLIM value.

**On Dump DD Statements:** JES3 ignores an OUTLIM parameter on a SYSABEND or SYSUDUMP DD statement.

# DD: OUTLIM

## Relationship to Other Control Statements

Output can also be limited by the following:

- The LINES, BYTES, or PAGES parameters of the JES2 /\*JOBPARM statement. See “/\*JOBPARM Statement” on page 16-4.
- The LINES or CARDS parameters of the JES3 /\*MAIN statement. See “/\*MAIN Statement” on page 17-23.

## Example of the OUTLIM Parameter

```
//OUTDD DD SYSOUT=F,OUTLIM=1000
```

The limit for the number of logical records is 1000.

## OUTPUT Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the OUTPUT parameter with the SYSOUT parameter to associate a system output data set explicitly with an OUTPUT JCL statement. JES processes the system output data set using the options from this DD statement combined with the options from the referenced OUTPUT JCL statement.

When the OUTPUT parameter references more than one OUTPUT JCL statement, the system produces separate output for each OUTPUT JCL statement.

**Note:** Code the OUTPUT parameter only on a DD statement with a SYSOUT parameter. Otherwise, the system checks the OUTPUT parameter for syntax then ignores it.

**References:** For information on the OUTPUT JCL statement, see Chapter 14, “Coding the OUTPUT JCL Statement.”

### Syntax:

```

      {(*.name[,*.name]...)}
OUTPUT={(*.stepname.name[,*.stepname.name]...)}
      {(*.stepname.procstepname.name[,*.stepname.procstepname.name]...)}

```

- A reference is one of the following:

```

*.name
*.stepname.name
*.stepname.procstepname.name

```

- You can omit the parentheses if you code only one reference.
- You must not code a null in an OUTPUT parameter. For example, OUTPUT=(,\*.name) is invalid.
- You can reference a maximum of 128 OUTPUT JCL statements on one OUTPUT parameter.
- You can code references in any combination. For example, the following are valid:

```

//EXA DD SYSOUT=A,OUTPUT=( *.name,*.name,*.stepname.name)
//EXB DD SYSOUT=A,OUTPUT=( *.stepname.name,
//      *.stepname.procstepname.name,*.name)

```

- You can code the references to OUTPUT JCL statements in any order.

# DD: OUTPUT

## Subparameter Definition

### \*.name

Refers to the earlier OUTPUT JCL statement with **name** in its name field. The system searches for the OUTPUT JCL statement first in the same step, then before the first EXEC statement of the job.

### \*.stepname.name

Refers to the earlier OUTPUT JCL statement, **name**, in this step or an earlier step, **stepname**, in the same job.

### \*.stepname.procstepname.name

Refers to an OUTPUT JCL statement in a cataloged or in-stream procedure. **Stepname** is the name of this job step or an earlier job step that calls the procedure, **procstepname** is the name of the procedure step that contains the OUTPUT JCL statement, and **name** is the name field of the OUTPUT JCL statement.

## Defaults

If no OUTPUT parameter is specified on a sysout DD statement, JES obtains processing options for the system output data set in the following order:

1. From all OUTPUT JCL statements containing **DEFAULT=YES** in the same step.
2. From all OUTPUT JCL statements containing **DEFAULT=YES** before the first EXEC statement in the job, provided that the step contains no OUTPUT JCL statements with **DEFAULT=YES**.
3. Only from the sysout DD statement, provided that neither the step nor job contains any OUTPUT JCL statements with **DEFAULT=YES**.

Each OUTPUT JCL statement represents an output processing request that JES is to use in processing the sysout data set.

## Overrides

When an OUTPUT JCL statement is used with the sysout DD statement to specify processing, JES handles parameters as follows:

- If a parameter appears on the DD statement, JES uses the parameter.
- If a parameter appears only on the OUTPUT JCL statement, JES uses the parameter.
- If the same parameter appears on both statements, JES uses the DD parameter.

JES uses the whole overriding parameter, ignoring the whole overridden parameter. If a subparameter is left off the overriding parameter, the system does not pick up that subparameter from the overridden parameter. For example:

```
//EXAMP2 OUTPUT  DEFAULT=YES,FLASH=(,3)
//FVZ2   DD      SYSOUT=F,OUTPUT=*.EXAMP2,FLASH=(GF12)
```

Only GF12 is used. The system ignores all of the FLASH parameter on the OUTPUT JCL statement, including the second parameter.

## Relationship to Other Subparameters

**Null Subparameters:** A null first subparameter is invalid in a FLASH or MODIFY parameter on a DD statement, but is permitted on an OUTPUT JCL statement. For example, MODIFY=(,3) is valid only on an OUTPUT JCL statement.

**SYSOUT Code-name Subparameter:** You cannot reference a JES2 /\*OUTPUT statement using the code-name subparameter of the SYSOUT parameter if either of the following is also coded:

- The OUTPUT parameter on the same DD statement.
- An OUTPUT JCL statement containing DEFAULT=YES in the same step or before the EXEC statement of the job, when the DD statement does not contain an OUTPUT parameter.

**DEFAULT Parameter on OUTPUT JCL Statement:** When you code DEFAULT=YES on an OUTPUT JCL statement, you can still refer to the OUTPUT JCL statement in the OUTPUT parameter of a sysout DD statement.

DD

## Location in the JCL

All referenced OUTPUT JCL statements must precede the DD statement that refer to them.

## No Match for OUTPUT Name

If the system finds no match for the name coded in the OUTPUT parameter, the system issues a JCL error message and fails the job.

## Processing Options in Multiple References

Processing options for a system output data set come from one sysout DD statement and one OUTPUT JCL statement. Processing options are **not** cumulative across a group of OUTPUT JCL statements.

When the OUTPUT parameter on a sysout DD statement explicitly refers to more than one OUTPUT JCL statement, each combination of the sysout DD statement and one of the referenced OUTPUT JCL statements produces one set of printed or punched output.

The same rule applies when one sysout DD statement implicitly references several default OUTPUT JCL statements.

## Examples of the OUTPUT Parameter

```
//OUT DD SYSOUT=A,OUTPUT=*.OUTA
```

This OUTPUT parameter specifies that the system is to use the processing options specified on OUTPUT JCL statement OUTA. OUTPUT JCL statement OUTA is located either in this job step before this DD statement or before the first EXEC statement in this job.

## DD: OUTPUT

```
//OUT2 DD SYSOUT=A,OUTPUT=(*.OUTA,*.STEP1.OUTC),DEST=HQ
```

This OUTPUT parameter specifies that the system is to produce two sets of output for the system output data set, as follows:

- Output according to the combined processing options on OUTPUT JCL statement OUTA and DD statement OUT2. OUTA is earlier in this step or is before the first EXEC statement of the job.
- Output according to the combined processing options on OUTPUT JCL statement OUTC and DD statement OUT2. OUTC is in the step STEP1, which can be this step or an earlier step in this job.

---

```
//STEP5 EXEC PGM=WRITER  
//OUT3 DD SYSOUT=A,OUTPUT=(*.STEP1.OUTB,*.STEP2.OUTD,  
// *.STEP4.PSTEP1.OUTY,*.OUTE,*.STEP3.PSTEP2.OUTX)
```

This OUTPUT parameter specifies that the system is to produce five separate sets of output for the system output data set, as follows:

1. For this DD statement and OUTPUT JCL statement OUTB, which is in earlier STEP1.
2. For this DD statement and OUTPUT JCL statement OUTD, which is in earlier STEP2.
3. For this DD statement and OUTPUT JCL statement OUTY, which is in procedure step PSTEP1 of the procedure called by earlier STEP4.
4. For this DD statement and OUTPUT JCL statement OUTE, which is before the first EXEC statement of this job.
5. For this DD statement and OUTPUT JCL statement OUTX, which is in procedure step PSTEP2 of the procedure called by earlier STEP3.

Note that the references to OUTPUT JCL statements are in no particular order.

---

```

//EXAMP      JOB      MSGCLASS=A
//OUT1       OUTPUT   DEFAULT=YES,DEST=COMPLEX7,FORMS=BILLING,
//           CHARS=(AOA,AOB),COPIES=2
//           //
//OUT2       OUTPUT   DEFAULT=YES,DEST=COMPLEX3
//OUT3       OUTPUT   DEST=COMPLEX1
//STEP1      EXEC     PGM=ORDERS
//OUT4       OUTPUT   DEFAULT=YES,DEST=COMPLEX9
//R1         DD       SYSOUT=A,OUTPUT=* .OUT3
//R2         DD       SYSOUT=A
//STEP2      EXEC     PGM=BILLING
//B1         DD       SYSOUT=A
//B2         DD       SYSOUT=A

```

This job requests that the system produce nine sets of output: eight sets of job output and one set for the system-managed output data set.

**Set 1**

In STEP1, DD statement R1 explicitly references OUTPUT JCL statement OUT3. Therefore, the system produces one set of output at COMPLEX1 for DD statement R1 combined with OUTPUT JCL statement OUT3.

**Set 2**

In STEP1, DD statement R2 implicitly references OUTPUT JCL statement OUT4 for both of the following reasons:

- DD statement R2 does not contain an OUTPUT parameter.
- STEP1 contains an OUTPUT JCL statement with DEFAULT = YES.

Therefore, the system produces one set of output at COMPLEX9 for DD statement R2 combined with OUTPUT JCL statement OUT4.

**Sets 3 through 8**

In STEP2, DD statements B1 and B2 implicitly reference OUTPUT JCL statements OUT1 and OUT2 for all of the following reasons:

- DD statements B1 and B2 do not contain OUTPUT parameters.
- STEP2 does not contain an OUTPUT JCL statement with DEFAULT = YES.
- DEFAULT = YES is specified on OUTPUT JCL statements OUT1 and OUT2.

Therefore, the system produces three sets of output each for DD statements B1 and B2:

**Sets 3 and 4** at COMPLEX7 for DD statement B1 combined with OUTPUT JCL statement OUT1.

**Set 5** at COMPLEX3 for DD statement B1 combined with OUTPUT JCL statement OUT2.

**Sets 6 and 7** at COMPLEX7 for DD statement B2 combined with OUTPUT JCL statement OUT1.

**Set 8** at COMPLEX3 for DD statement B2 combined with OUTPUT JCL statement OUT2.

# DD: OUTPUT

## Set 9

The system-managed output data set is processed locally because of the MSGCLASS parameter on the JOB statement.



## PROTECT Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the PROTECT parameter to tell the Resource Access Control Facility (RACF) to create a discrete profile to protect:

- A data set on direct access.
- A tape volume.

Use the PROTECT parameter only if RACF is installed and active.

**References:** For more information on discrete profiles and RACF, see *Resource Access Control Facility (RACF) Security Administrator's Guide*.

**Syntax:**

```
PROTECT=YES
```

### Subparameter Definition

**YES**

Requests RACF to create a discrete profile to protect a direct access data set or a tape volume. This parameter can also be coded as Y.

### Relationship to Other Parameters

Do not code the following parameters with the PROTECT parameter.

*	DLM	QNAME
BURST	DYNAM	SYSOUT
CHARS	FCB	TERM
DATA	FLASH	UCS
DDNAME	MODIFY	
	OUTPUT	

### Requirements for Protecting a Tape Volume

A DD statement that contains a PROTECT parameter to establish RACF protection for a tape volume must:

- Request one of the volume's existing data sets in the DSNAME parameter.
- Specify or imply VOLUME=PRIVATE.
- Specify DISP=OLD. The status must not be MOD treated as OLD.

## DD: PROTECT

- Specify a label type in the LABEL parameter of:
  - SL or SUL for IBM standard labels.
  - NSL for nonstandard labels. In this case, the NSL installation exit routine must issue a RACFDEF macro instruction. See *SPL: Supervisor* for a description of RACFDEF.
  - AI or AUL for ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 tape labels.

To establish protection, the tape data set must be:

- The first on the volume, except for NSL tapes.
- Located on the first volume of a multivolume data set, except for NSL tapes. Thus, the volume-sequence-number of the VOLUME parameter must be 1.
- Opened for OUTPUT or OUTIN by the program.

### Requirements for Protecting a Direct Access Data Set

A DD statement that contains a PROTECT parameter to establish RACF protection for a direct access data set must:

- Specify a permanent data set in the DSNAME parameter.
- Specify a status of DISP=NEW or MOD treated as NEW. RACF can establish protection only when the data set is being created.
- Specify a normal termination disposition of KEEP, PASS, CATALG, or UNCATLG and an abnormal termination disposition of KEEP, CATALG, or UNCATLG. A disposition of DELETE is not allowed, to prevent unauthorized persons from deleting protected data sets.

### Examples of the PROTECT Parameter

```
//DASD DD UNIT=3330,VOLUME=SER=333000,DSNAME=USER37.MYDATA,  
// DISP=(,CATLG),SPACE=(TRK,1),PROTECT=YES
```

This DD statement requests RACF protection for the new direct access data set USER37.MYDATA.

---

```
//TAPEDATA DD UNIT=3400-5,DISP=(OLD,UNCATLG),  
// LABEL=(,SL),VOLUME=SER=T49850,DSNAME=TAPEDS,  
// PROTECT=YES
```

This DD statement requests RACF protection for tape volume T49850. Because a specific tape volume is requested, it automatically has the PRIVATE attribute. The volume has a standard label, and TAPEDS must be the first data set on the tape.

## QNAME Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the QNAME parameter to indicate that this DD statement defines a data set of telecommunications access method (TCAM) messages. The QNAME parameter refers to a TPROCESS macro instruction that defines a destination queue for the messages. Optionally, the QNAME parameter can also name a TCAM job to process the messages.

**References:** For information about TCAM and the TPROCESS macro instruction, see *TCAM Programmer's Guide*.

**Syntax:**

```
QNAME=procname[.tcamname]
```

### Subparameter Definition

#### procname

Identifies a TPROCESS macro instruction. The procname is 1 through 8 alphanumeric or national characters; the first character must be alphabetic or national. This procname must be identical to the procname in the name field of the TPROCESS macro instruction.

#### tcamname

Names a TCAM job or started task. The tcamname is 1 through 8 alphanumeric or national characters; the first character must be alphabetic or national.

The job can be a task started by an operator START command.

### Relationship to Other Parameters

The **only** DD parameter that you can code with the QNAME parameter is the DCB parameter.

The only DCB subparameters that you can code with the QNAME parameter are: BLKSIZE, BUFL, LRECL, OPTCD, and RECFM.

### Example of the QNAME Parameter

```
//DYZ DD QNAME=FIRST,DCB=(RECFM=FB,LRECL=80,BLKSIZE=320)
```

This DD statement defines a data set of TCAM messages. **FIRST** is the name of the TPROCESS macro instruction that specifies the destination queue to which the messages are routed. The DCB parameter supplies information not supplied in the program's DCB macro instruction for the data control block.

```
//DXD DD QNAME=SECOND.TCAM01
```

This DD statement defines a data set of TCAM messages. **SECOND** is the name of the TPROCESS macro instruction that specifies the destination queue to which the messages are routed. TCAM program TCAM01 will process the messages.

## DD: SPACE

### SPACE Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the SPACE parameter to request space for a new data set on a direct access volume. You can request space in two ways:

- Tell the system how much space you want and let the system assign specific tracks.
- Tell the system the specific tracks to be allocated to the data set.

Letting the system assign the specific tracks is most frequently used. You specify only that space is to be measured in tracks, cylinders, or blocks and how many of those tracks, cylinders, or blocks are required.

The SPACE parameter has no meaning for tape volumes; however, if you assign a data set to a device class that contains both direct access devices and tape devices, for example, UNIT=SYSSQ, you should code the SPACE parameter.

**References:** For more information on the SPACE parameter, see "Requesting Space for Non-VSAM Data Sets" on page 7-39.

**Syntax:**

For system assignment of space:

```
SPACE=({TRK,      } [ ,CONTIG ] )
({CYL,      } (primary-qty[,second-qty][,directory)][,RLSE][,MXIG ] [,ROUND])
({blklgth,}( [ ,      ][,index ])[,      ][,ALX ] )
```

To request specific tracks:

```
SPACE=(ABSTR,(primary-qty,address[,directory]))
( [ ,index ] )
```

- You can omit the parentheses around the primary quantity if you do not code secondary, directory, or index quantities. For example, SPACE=(TRK,20,,CONTIG).
- All the subparameters are positional. Code a comma to indicate an omitted subparameter if any others follow. Thus:
  - If you code primary and directory or index quantities and omit a secondary quantity, code a comma to indicate the omission. For example, SPACE=(TRK,(20,,2)).
  - If you omit RLSE but code a following subparameter, code a comma to indicate the omission. For example, SPACE=(TRK,(20,10),,CONTIG).
  - If you omit CONTIG, MXIG, or ALX and ROUND follows, code a comma to indicate the omission. For example, SPACE=(400,30,,,ROUND).

## Subparameter Definition

### System Assignment of Space

#### TRK

Requests that space be allocated in tracks.

#### CYL

Requests that space be allocated in cylinders.

#### blklgth

Specifies the average block length of the data. The system computes how many tracks to allocate. The blklgth is a decimal number from 1 through 65535.

#### primary-qty

Specifies one of the following:

- For TRK, the number of tracks to be allocated.
- For CYL, the number of cylinders to be allocated.
- For a block length, the number of data blocks in the data set.

**Note: When you specify TRK or CYL for a partitioned data set, the space for the directory is specified twice: once as part of the primary quantity and a second time separately as the directory quantity.**

One volume must have enough available space for the primary quantity. If you request a particular volume and it does not have enough space available for your request, the system terminates the job step. Allow for track overflow when computing track requirements.

#### second-qty

Specifies how many additional tracks or cylinders are to be allocated if more space is needed. The system can allocate up to 16 extents for a data set on a volume. An extent is space that is not contiguous to other space allocated to the data set. The extents for a data set include the primary quantity space and user-label space.

*Note:* BDAM data sets cannot be extended.

The system computes the number of tracks for the secondary quantity based on the DCB BLKSIZE subparameter, the DCB macro instruction, or the SPACE block length subparameter. The system does not allocate additional space until it is needed.

When you specify a secondary quantity and the data set requires additional space, the system allocates the specified quantity:

1. In contiguous tracks or cylinders, if available.
2. If not, in up to five extents.

Each time the data set requires more space, the system allocates the secondary quantity. This space is allocated on the same volume as the primary quantity until one of the following occurs:

- The volume does not have enough space available for the secondary quantity.
- 16 extents have been allocated to the data set.

## DD: SPACE

Then, the system allocates the secondary quantity on another volume. If the requested volumes have no more available space and if at least one volume is demountable, the system asks the operator to mount scratch (nonspecific) volumes until the secondary allocation is complete. If none of the volumes are demountable, the system abnormally terminates the job step.

### **directory**

Specifies the number of 256-byte records needed in the directory of a partitioned data set.

*Note:* When creating a partitioned data set, you must request space for a directory.

### **index**

For the index of an indexed sequential data set, specifies one of the following:

- For TRK, the number of tracks needed. The number of tracks must equal one or more cylinders.
- For CYL, the number of cylinders needed.

### **RLSE**

Requests that space allocated to an output data set, but not used, is to be released when the data set is closed. Unused space is released only if the data set is open for output and the last operation was a write.

If you specify RLSE and an abnormal termination occurs, the system does not release unused space even though the data set is open.

The system ignores a request to release unused space when a data set is closed if:

- Another job is sharing the data set.
- Another task in the same job is processing an OPEN, CLOSE, EOVS, or FEOV request for the data set.
- Another data control block is open for the data set.

The RLSE subparameter is ignored when TYPE = T is coded in the CLOSE macro instruction.

### **CONTIG**

Requests that space allocated to the data set must be contiguous. This subparameter affects only primary space allocation.

If CONTIG is specified and contiguous space is not available, the system terminates the job step.

### **MXIG**

Requests that space allocated to the data set must be (1) the largest area of available contiguous space on the volume and (2) equal to or greater than the primary quantity. This subparameter affects only primary space allocation.

*Note:* Do not code a MXIG subparameter for an indexed sequential data set.

**ALX**

Requests that up to five separate areas of contiguous space are to be allocated to the data set and each area must be equal to or greater than the primary quantity. This subparameter affects only primary space allocation.

*Note:* Do not code an ALX subparameter for an indexed sequential data set.

**ROUND**

When the first subparameter specifies the average block length, requests that space allocated to the data set must be equal to an integral number of cylinders.

**Request for Specific Tracks****ABSTR**

Requests that the data set be allocated at the specified location on the volume.

**primary-qty**

Specifies the number of tracks to be allocated to the data set.

The volume must have enough available space for the primary quantity. If it does not, the system terminates the job step.

**address**

Specifies the track number of the first track to be allocated. Count the first track of the first cylinder on the volume as 0. Count through the tracks on each cylinder until you reach the track on which you want the data set to start.

*Note:* Do not request track 0.

**directory**

Specifies the number of 256-byte records needed in the directory of a partitioned data set.

*Note:* When creating a partitioned data set, you must request space for a directory.

**index**

Specifies the number of tracks needed for the index of an indexed sequential data set. The number of tracks must equal one or more cylinders.

**Relationship to Other Parameters**

Do not code the following parameters with the SPACE parameter.

*	DLM
AMP	DYNAM
DATA	QNAME
DDNAME	

**With DCB KEYLEN for Block Requests:** If space is requested in blocks and the blocks have keys, code the DCB subparameter KEYLEN on the DD statement and specify the key length.

## DD: SPACE

### SPACE for New Data Sets on Mass Storage Volumes

- The SPACE parameter must be coded for new data sets when VOLUME=SER is coded. It is optional when MSVGP is coded. If you code neither VOLUME=SER nor MSVGP, SPACE must be coded even if you code VOLUME=PRIVATE.
- Contiguous space is the MSVGP default. If you want a noncontiguous primary space allocation, you must specify the SPACE parameter.

### Examples of the SPACE Parameter

```
//DD1 DD DSNAME=&&TEMP,UNIT=MIXED,SPACE=(CYL,10)
```

The DD statement defines a temporary data set. The UNIT parameter requests any available tape or direct access volume; MIXED is a name for a group of tape and direct access devices. If a tape volume is assigned, the SPACE parameter is ignored; if a direct access volume is assigned, the SPACE parameter is used to allocate space to the data set. The SPACE parameter specifies only the required subparameters: the type of units and a primary quantity. It requests that the system allocate 10 cylinders.

---

```
//DD2 DD DSNAME=PDS12,DISP=(,KEEP),UNIT=3350,  
// VOLUME=SER=25143,SPACE=(CYL,(10,,10),,CONTIG)
```

The DD statement defines a new partitioned data set. The system allocates 10 cylinders to the data set, of which ten 256-byte records are for a directory. Since the CONTIG subparameter is coded, the system allocates 10 contiguous cylinders on the volume.

---

```
//REQUEST1 DD DSNAME=EXM,DISP=NEW,UNIT=3330,VOLUME=SER=606674,  
// SPACE=(1024,75),DCB=KEYLEN=8
```

This DD statement requests space in block lengths. The average block length of the data is 1024 bytes. 75 blocks of data are expected as output. Each block is preceded by a key eight bytes long. The system computes how many tracks are needed, depending on the device requested in the UNIT parameter.

---

```
//REQUEST2 DD DSNAME=PET,DISP=NEW,UNIT=3330,VOLUME=SER=606674,  
// SPACE=(1024,(75)),DCB=KEYLEN=8,SPACE=(ABSTR,(1,1))
```

In this example, the SPACE parameter asks the system to allocate one track, beginning on the second track of the volume. The first track is 0.



## SUBSYS Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the SUBSYS parameter to request a subsystem to process this data set and, optionally, to specify parameters defined by the subsystem.

In a loosely-coupled multiprocessing environment, the requested subsystem must be defined on all processors that could interpret this DD statement.

**References:** For more information on the SUBSYS parameter and subsystem-defined parameters, refer to the documentation for the requested subsystem.

### Syntax:

```
SUBSYS=( subsystem-name [ , subsystem-subparameter ] ... )
```

**Single Subparameter:** You can omit the parentheses if you code only the subsystem-name.

**Number of Subparameters:** Code up through 254 subsystem-subparameters, if needed.

**Multiple Subparameters:** When the parameter contains more than one the subsystem-name, separate the subparameters by commas and enclose the subparameter list in parentheses. For example, SUBSYS=(XYZ,1724,DT25).

**Positional Subparameters:** If you omit a subparameter that the subsystem considers positional, code a comma in its place.

**Special Characters:** When a subparameter contains special characters, enclose the subparameter in apostrophes. For example, SUBSYS=(XYZ,1724,'KEY=Y').

Code each apostrophe that is part of a subparameter as two consecutive apostrophes. For example, code O'Day as SUBSYS=(XYX,1724,'NAME=O'DAY').

**Continuation onto Another Statement:** Enclose the subparameter list in only one set of parentheses. End each statement with a comma after a complete subparameter. For example:

```
//DS1 DD DSNAME=DATA1,SUBSYS=(XYZ,1724,'KEY=Y',
// DT25,'NAME=O'DAY')
```

## Subparameter Definition

### subsystem-name

Identifies the subsystem. The subsystem name is 1 through 4 alphanumeric or national characters; the first characters must be alphabetic or national. The subsystem must be available in the installation.

## DD: SUBSYS

### subsystem-subparameter

Specifies information needed by the subsystem. A subparameter consists of alphanumeric, national, or special characters.

### Relationship to Other Parameters

Do not code the following DD parameters with the SUBSYS parameter.

*	DYNAM
DATA	OUTPUT
DDNAME	QNAME
DLM	SYSOUT

The specified subsystem can define other parameters that must not be coded with the SUBSYS parameter.

**Ignored but Permitted DD Parameters:** If the following DD parameters are specified, they are checked for syntax and ignored:

COPIES	OUTLIM
DEST	SPACE
FCB	UNIT

**DISP Parameter:** The system checks the DISP status subparameter for syntax, but always indicates a status of MOD to the subsystem. If the DISP normal or abnormal termination subparameter is CATLG or UNCATLG, the system allocates the appropriate catalog to the subsystem.

**DUMMY Parameter:** If DUMMY is specified with SUBSYS, the subsystem checks the syntax of the subsystem subparameters. If they are acceptable, the system treats the data set as a dummy data set.

**When This Statement Overrides a Procedure Statement:** If SUBSYS appears on a DD statement that overrides a DD statement in a cataloged or in-stream procedure, the following occurs:

- The system ignores a UNIT parameter, if specified, on the overridden DD statement.
- The system nullifies a DUMMY parameter, if specified, on the overridden DD statement.

### Examples of the SUBSYS Parameter

```
//DD1 DD DSNAME=ANYDS,DISP=OLD,SUBSYS=ABC
```

The DD statement asks subsystem ABC to process data set ANYDS.

---

```
//DD1 DD DSNAME=ANYDS,DISP=OLD,SUBSYS=(XYZ2,  
// 'KEYWORD=DATA VALUE1')
```

The DD statement asks subsystem XYZ2 to process data set ANYDS. The system passes the subparameter KEYWORD=DATA VALUE1 to the subsystem. The parameter is enclosed in apostrophes because it contains an equal sign and a blank, which are special characters.

---

```
//DD1 DD DSNAME=ANYDS,DISP=OLD,SUBSYS=(XYZ2,IKJ2,  
//      'NAME='MODULE1'',DATE=4/11/86')
```

The DD statement asks subsystem XYZ2 to process the data set ANYDS. The system passes three subparameters to the subsystem: IKJ2, NAME='MODULE1' and DATE=4/11/86. Note that the character string MODULE1 is passed to the subsystem enclosed in apostrophes.

## SYSOUT Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the SYSOUT parameter to define this data set as a system output data set, also called a sysout data set. The SYSOUT parameter also:

- Assigns this sysout data set to an output class.
- Optionally requests an external writer to process the sysout data set rather than JES. An external writer is an IBM- or installation-written program.
- Optionally identifies the forms on which the data set is to be printed or punched.
- Optionally refers to a JES2 /\*OUTPUT statement for processing parameters.

**Note:** If a system output data set has the same class as the JOB statement MSGCLASS parameter, the job log appears on the same output listing as this output data set.

**References:** For more information on output classes, see "Assigning System Output Data Sets to Output Classes" on page 7-51, and on external writers, see *SPL: Job Management*.

### Syntax:

```
        {class-name                }  
        {                            }  
SYSOUT={{( {class-name} [,writer-name] [,form-name] )}  
        {{( {,                } [,                ] [,code-name] )}  
        {                            }  
        {*                          }
```

- You can omit the parentheses if you code only a class-name.
- All of the subparameters are positional. Code a comma to indicate an omitted subparameter if any others follow. Thus:
  - If you omit the class-name, code a comma to indicate the omission. For example, SYSOUT=(,XWTR,FM26).  
  
Because the class-name is not optional, its omission must be indicated by a comma whether other subparameters follow or not. For example, code a null class-name as SYSOUT=(,).
  - If you omit a writer-name but code a form-name or code-name, code a comma to indicate the omission. For example, SYSOUT=(A,,FM26).

## Subparameter Definition

### class-name

Identifies the output class for the data set. class-name is one character: A through Z or 0 through 9. The attributes of each output class are defined during JES initialization; specify the class with the desired attributes.

\*

Requests the output class in the MSGCLASS parameter on the JOB statement.

### writer-name

Identifies an external writer. The name is 1 to 8 alphanumeric characters.

Two names are reserved for JES: INTRDR for JES2 and STDWTR for JES3. Code INTRDR to specify that JES2 is to treat this data set as an input job stream. For more information, see "Specifying the Internal Reader" on page 7-52.

### form-name

Identifies the print or punch forms. form-name is 1 to 4 alphanumeric or national characters.

### code-name

Identifies a JES2 /\*OUTPUT statement from which JES2 is to obtain processing characteristics. The code-name is the 1 to 4 alphanumeric or national characters appearing as the code parameter on the JES2 /\*OUTPUT statement.

#### Note:

- code-name is supported only on JES2 systems.
- Do not specify the code-name subparameter when the job or job step contains an OUTPUT JCL statement.

## Defaults

If no writer-name subparameter is specified on this DD statement or a referenced OUTPUT JCL statement, the installation's job entry subsystem processes the sysout data set.

If no form-name subparameter is specified on this DD statement or a referenced OUTPUT JCL statement, JES uses an installation default specified at initialization.

## Overrides

The class-name subparameter of the DD statement SYSOUT parameter overrides an OUTPUT JCL CLASS parameter. On the DD statement, you must code a null class-name in order to use the OUTPUT JCL CLASS parameter; for example:

```
//OUTDS DD SYSOUT=(,)
```

The writer-name subparameter of the DD statement SYSOUT parameter overrides an OUTPUT JCL WRITER parameter.

## DD: SYSOUT

The form-name subparameter of the DD statement SYSOUT parameter overrides an OUTPUT JCL FORMS parameter. Note that the SYSOUT form-name subparameter can be only four characters maximum while both the OUTPUT JCL FORMS form-name and the JES initialization default form names can be eight characters maximum.

### Relationship to Other Parameters

Do not code the following DD parameters with the SYSOUT parameter.

*	DISP	MSVGP
AMP	DLM	PROTECT
CHKPT	DSNAME	QNAME
DATA	DYNAM	SUBSYS
DDNAME	LABEL	

**Parameters on Procedure DD Statements that are Overridden:** When an overriding DD statement contains a SYSOUT parameter, the system ignores a UNIT parameter on the overridden DD statement in the cataloged or in-stream procedure.

If the overridden DD statement contains a DSNAME parameter, the system issues a JCL warning message.

**SYSOUT and DEST Subparameters:** Do not code the SYSOUT writer-name subparameter when coding a DEST userid subparameter. These subparameters are mutually exclusive. You can code:

```
//VALID1 DD SYSOUT=D,DEST=(node,userid)
//VALID2 DD SYSOUT=(D,program-name),DEST=node
```

**With DCB Subparameters:** JES2 ignores DCB=PRTSP=2 on a DD statement that also contains a SYSOUT parameter.

### Relationship to Other Control Statements

SYSOUT cannot specify a code-name subparameter in a job or job step that contains an OUTPUT JCL statement; in this case, JES2 treats the third subparameter as a form-name, instead of a reference to a JES2 /\*OUTPUT statement.

### Starting an External Writer when Requested

A system command, from the operator or in the input stream, must start an external writer before the system process a JCL statement that specifies it.

### Backward References

Do not refer to a earlier DD statement that contains a SYSOUT parameter.

## Held Classes in a JES2 System

If SYSOUT specifies a class-name that is defined to JES2 as a held class, the JOB MSGCLASS parameter should specify (1) the same class-name as the SYSOUT parameter or (2) another class that is also defined to JES2 as a held class. For information on the JOB MSGCLASS parameter, see “MSGCLASS Parameter” on page 10-14.

## Significance of Output Classes

To print this output data set and the messages from your job on the same output listing, code one of the following:

- The same output class in the DD SYSOUT parameter as in the JOB MSGCLASS parameter.
- DD SYSOUT=\* to default to the JOB MSGCLASS output class.
- A null class-name, DD SYSOUT=(,) to default to one of the following:
  1. The CLASS parameter in an explicitly or implicitly referenced OUTPUT JCL statement. In this case, the OUTPUT JCL CLASS parameter should specify the same output class as the JOB MSGCLASS parameter.
  2. The JOB MSGCLASS output class, if no OUTPUT JCL statement is referenced or if the referenced OUTPUT JCL statement contains CLASS=\*

## Examples of the SYSOUT Parameter

```
//DD1 DD SYSOUT=P
```

In this example, the DD statement specifies that JES is to write the system output data set to the device handling class P output.

---

```
//JOB50 JOB , 'C. BROWN', MSGCLASS=C
//STEP1 EXEC PGM=SET
//DDX DD SYSOUT=C
```

In this example, DD statement DDX specifies that JES is to write the system output data set to the device handling class C output. Because the SYSOUT parameter and the MSGCLASS parameter specify the same class, the messages from this job and the system output data set can be written to the same device.

---

```
//DD5 DD SYSOUT=(F, , 2PRT)
```

In this example, the DD statement specifies that JES is to write the system output data set to the device handling class F output. The data set is to be printed or punched on forms named 2PRT.

## DD: SYSOUT

```
//JOB51 JOB ACCT123,MAEBIRD,MSGCLASS=B  
//STEP1 EXEC PGM=RPTWTR  
//OUT1 OUTPUT CLASS=*  
//REPORT DD SYSOUT=(,),OUTPUT=*,OUT1
```

In this example, JES processes the sysout data set defined in DD statement REPORT in output class B. Because SYSOUT specifies a null class-name, the CLASS parameter in the explicitly referenced OUTPUT JCL statement is used. That CLASS parameter specifies the MSGCLASS output class.



## TERM Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the TERM parameter to indicate to the system that a data set is coming from or going to a terminal for a TSO user.

**Syntax:**

```
TERM=TS
```

### Subparameter Definition

TS

Indicates to the system one of the following:

- That the input or output data set is coming from or going to a TSO user, when TERM = TS appears in a foreground job submitted by a TSO user.
- That the data set should be treated as an in-stream DD \* data set, when TERM = TS appears on a DD statement in a batch job.

### Relationship to Other Parameters

Do not code the following DD parameters with the TERM parameter.

*	DLM
AMP	DYNAM
DATA	PROTECT
DDNAME	QNAME

Code only the DCB and SYSOUT parameters with the TERM parameter. The system ignores any other DD parameters.

### Location in the JCL

In a foreground TSO job, a DD statement containing TERM = TS and a SYSOUT parameter begins an in-stream data set.

In a batch job, a DD statement containing TERM = TS begins an in-stream data set.

When concatenating DD statements, the DD statement that contains TERMS = TS must be the last DD statement in a job step.

## DD: TERM

### Examples of the TERM Parameter

```
//DD1 DD TERM=TS, SYSOUT=C
```

In a foreground job submitted from a TSO terminal, this DD statement defines an in-stream data set coming from the TSO terminal. In a batch job, TERM=TS is ignored.

---

```
//DD3 DD UNIT=3400-5, DISP=(MOD, PASS), TERM=TS, LABEL=(, NL),  
//      DCB=(LRECL=80, BLKSIZE=80)
```

In a foreground job, the system ignores all of the parameters in this example except TERM and DCB. In a batch job, the system ignores only the TERM parameter.

## UCS Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the UCS parameter to identify:

- The universal character set (UCS) JES is to use in printing the output data set.
- A print train (print chain or print band) JES is to use in printing the output data set on an impact printer.
- A character-arrangement table for a data set printed on a 3800 Printing Subsystem in a JES2 system. In this use, the UCS parameter acts like a CHARS parameter.

The UCS image specifies the special character set to be used. JES loads the image into the printer's buffer. The UCS image is stored in SYS1.IMAGELIB. IBM provides the special character set codes in Figure 12-1.

**References:** For more information on the UCS parameter, see "Requesting a Special Character Set Using the UCS Feature" on page 7-59 and *SPL: Data Management*.

**Syntax:**

```
UCS={character-set-code
      {(character-set-code[,FOLD|,][,VERIFY])}}
```

- You can omit the parentheses if you code only a character-set-code.
- All of the subparameters are positional. If you omit FOLD but code VERIFY, code a comma to indicate the omission. For example, UCS=(AN,,VERIFY).

### Subparameter Definition

#### character-set-code

Identifies a universal character set. The character-set-code is 1 to 4 alphanumeric characters. See Figure 12-1 for IBM standard special character set codes.

#### FOLD

Requests that the chain or train for the universal character set is loaded in fold mode. Fold mode is described in *IBM 2821 Control Unit*. Fold mode is most often used when upper- and lower-case data is to be printed only in uppercase.

#### VERIFY

Requests that, before the data set is printed, the operator verify visually that the character set image is for the correct chain or train. The character set image is displayed on the printer before the data set is printed.

## DD: UCS

1403	3203 Model 5	3211	Characteristics
AN	AN	A11	Arrangement A, standard EBCDIC character set, 48 characters
HN	HN	H11	Arrangement H, EBCDIC character set for FORTRAN and COBOL, 48 characters
		G11	ASCII character set
PCAN	PCAN		Preferred alphanumeric character set, arrangement A
PCHN	PCHN		Preferred alphanumeric character set, arrangement H
PN	PN	P11	PL/I alphanumeric character set
QN	QN		PL/I preferred alphanumeric character set for scientific applications
QNC	QNC		PL/I preferred alphanumeric character set for commercial applications
RN	RN		Preferred character set for commercial applications of FORTRAN and COBOL
SN	SN		Preferred character set for text printing
TN	TN	T11	Character set for text printing, 120 characters
XN			High-speed alphanumeric character set for 1403, Model 2
YN			High-speed preferred alphanumeric character set for 1403, Model N1

*Note:* Where three values exist (for the 1403, 3211, and 3203 Model 5 printers), code any one of them. JES selects the set corresponding to the device on which the data set is printed.

Not all of these character sets may be available at your installation. Also, an installation can design character sets to meet special needs and assign a unique code to them. Follow installation procedures for using character sets.

Figure 12-1. Special Character Sets for the 1403, 3203 Model 5, and 3211 Printers

## Defaults

If you do not code the UCS parameter, the system checks the UCS image in the printer's buffer; if it is a default image, as indicated by its first byte, JES uses it. If it is not a default image, JES loads the UCS image that is the installation default specified at JES initialization.

On an impact printer, if the chain or train does not contain a valid character set, JES asks the operator to specify a character set and to mount the corresponding chain or train.

## Overrides

For printing on a printer with the UCS feature, the UCS parameter on a sysout DD statement overrides an OUTPUT JCL UCS parameter. For printing on a 3800, a CHARS parameter on the sysout DD statement or the OUTPUT JCL statement overrides all UCS parameters.

For a data set scheduled to the Print Services Facility (PSF), the PSF uses the following parameters, in override order, to select the font list:

1. Font list in the SYS1.IMAGELIB member specified by an OUTPUT JCL PAGEDDEF parameter.
2. DD CHARS parameter.
3. OUTPUT JCL CHARS parameter.
4. DD UCS parameter.
5. OUTPUT JCL UCS parameter.

6. JES installation default for the device.
7. Font list on the PAGEDEF parameter in the PSF cataloged procedure.

See “PAGEDEF Parameter” on page 14-43 for more information.

## Relationship to Other Parameters

Do not code the following DD parameters with the UCS parameter.

*	DLM
AMP	DYNAM
DATA	PROTECT
DDNAME	QNAME

Do not code the UCS parameter with the DCB subparameters RKP and CYLOFL.

Do not code the FOLD and VERIFY subparameters on the same statement with a SYSOUT parameter; the system ignores the FOLD and VERIFY subparameters.

## Using Special Character Sets

To use a special character set, SYS1.IMAGELIB must contain an image of the character set, and the chain or train for the character set must be available. IBM provides standard special character sets, and the installation may provide user-designed special character sets.

## Examples of the UCS Parameter

```
//DD1 DD UNIT=1403,UCS=(YN,,VERIFY)
```

In this example, the DD statement requests a 1403 Printer. The UCS parameter requests the chain or train for special character set code YN. Because VERIFY is coded, the system will display the character set image on the printer before the data set is printed.

```
//DD2 DD SYSOUT=G,UCS=PN
```

In this example, the DD statement requests the device for output class G. If the device is a printer with the UCS feature, the system will ask the operator to mount the chain or train for UCS code PN, if it is not already mounted. If the device is a 3800 Printing Subsystem, the system will use the UCS subparameter to select the font list. Otherwise, the system ignores the the UCS parameter.

## DD: UNIT

### UNIT Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the UNIT parameter to ask the system to place the data set on:

- A specific device.
- A certain type or group of devices.
- The same device as another data set.

The UNIT parameter can also tell the system how many devices to assign and request that the operator should defer mounting the volume until the data set is opened.

**References:** For more information on the UNIT parameter, see "Requesting Units and Volumes" on page 7-24.

**Syntax:**

```
{ ([device-address][,unit-count] ) }
{UNIT=( [device-type ] [,P ] [,DEFER] ) }
{ ([group-name ] [, ] ) }
{ }
{UNIT=AFF=ddname }
```

- You can omit the parentheses if you code only the first subparameter.
- All of the subparameters are positional. If you omit unit-count or P but code DEFER, code a comma to indicate the omission. For example, UNIT=(3420,,DEFER).

### Subparameter Definition

**device-address**

Identifies a particular device. device-address is the 3-character hexadecimal address for the device.

Do not identify a device by its address unless it is absolutely necessary. Specifying a device address limits device assignment; it will delay a job if another job is using the device.

**device-type**

Requests a device by its generic name, which is an IBM-supplied name that identifies a device by its machine type and model. For example, UNIT=3350. IBM device types are listed in *SPL: System Generation Reference*. Obtain a list of device types from your installation.

Following are rules for certain devices:

- For a 3330 Disk Storage Model 11, code UNIT = 3330-1.
- If your installation has 3340 Direct Access Storage Facilities both with and without the Fixed Head feature, do not code the device type in the UNIT parameter. Instead, code the device address or a group-name.
- For mass storage volumes, code UNIT = 3330V.

#### group-name

Requests a group of devices by an esoteric name, which the installation assigned to the device(s) during system generation in the UNITNAME system generation macro or which IBM assigned to the devices. The group-name is 1 through 8 alphanumeric characters.

A group-name can identify a single device or a group of devices. If it represents a group, the group can consist of devices of the same type or of different types. For example, a group can contain both direct access and tape devices.

When you code a group name, the system assigns any available device from the group. If a group consists of only one device, the system assigns that device. If the group consists of more than one device type, the units requested are allocated from the same device type. For example, if GPDA contains 3330 Disk Storage and 3350 Direct Access Storage devices, a request for two units would be allocated to two 3330s or to two 3350s.

If a data set that was created using the group-name subparameter is to be extended, the system allocates additional devices of the same type as the original devices. However, the additional devices may not necessarily be from the same group.

IBM assigned group-names include SYSALLDA, which contains all direct access devices defined to the system.

#### unit-count

Requests the number of devices the system is to assign to the data set. The unit-count is a decimal number from 1 through 59.

You can receive more devices than the unit-count specifies if:

- You specify VOLUME = REF and/or a permanently resident or reserved volume.
- The VOLUME parameter indicates more devices. The system uses the greatest of the following: the unit-count, the volume-count of the VOLUME parameter, and the number of serial numbers in the VOLUME = SER subparameter.

Code a unit count if the data set needs more than one device when the DD statement receives a passed data set or refers in a VOLUME = REF subparameter to a cataloged data set or earlier DD statement for volume and unit information. Otherwise, the system assigns one device.

If two DD statements in a step request the same volume and either DD statement requests any other volume(s), the system assigns an additional device. For more information on determining the number of devices for a request, see *SPL: Job Management*.

## DD: UNIT

### P

Asks the system to allocate the same number of devices as requested in the **VOLUME** volume-count or **SER** subparameter, whichever is higher. Thus, all volumes for the data set are mounted in parallel.

### DEFER

Asks the system to assign the data set to device(s) but requests that the volume(s) not be mounted until the data set is opened. To defer mounting, **DEFER** must be specified or implied for all **DD** statements that reference the volume.

If you request deferred mounting of a volume and the data set on that volume is never opened by the processing program, the volume is never mounted during the job step.

Following are restrictions on the use of **DEFER**:

- The system ignores **DEFER** for a new data set on direct access.
- Do not code **DEFER** on a **SYSCKEOV** **DD** statement.
- Do not request deferred mounting for volumes belonging to a mass storage volume group (**MSVGP**) when the step contains new data set requests for the same mass storage volume group. The delay can cause volume conflicts within the job or between jobs and so cause poor performance.

### AFF = ddname

Requests that the system allocate different data sets residing on different, removable volumes to the same device during execution of the step. This request is called **unit affinity**. The **ddname** is the **ddname** of an earlier **DD** statement in the same step.

Use **unit affinity** to reduce the number of devices used in a job step. Request that an existing data set be assigned to the same device(s) as another existing data set.

Following are restrictions on the use of **unit affinity**:

- Do not code **DISP=NEW** with **UNIT=AFF=ddname** if the referenced data set for **DD** statement **ddname** resides on a direct access device. If coded, the system terminates the job. If the referenced data set can be allocated to either tape or direct access devices, the system allocates both requests to tape devices.
- Do not code **UNIT=AFF** on a **DD \*** or **DD DATA** statement. The system ignores **UNIT=AFF** and defaults the device name to **SYSALLDA**. **SYSALLDA** is the system-defined group-name for all direct access devices defined to the system.
- Do not code **UNIT=AFF=ddname** when **DD** statement **ddname** contains **FREE=CLOSE**.

If the referenced **DD** statement is not earlier in the same step, the system treats the **DD** statement containing **UNIT=AFF** as a **DD DUMMY** statement.



## Overrides

If you code `SYSOUT` and `UNIT` on the same statement, the `SYSOUT` specification overrides the `UNIT` specification.

The system also obtains device information when the system obtains volume serial information from:

- A `VOLUME = REF = dsname` reference to an earlier data set.
- A `VOLUME = REF = ddname` reference to an earlier `DD` statement.
- The volume(s) for a passed data set.
- The catalog for a cataloged data set.

However, you can override the retrieved device information if the device you specify is a subset of the retrieved device information. For example, if the retrieved unit grouping is 3350, and the specified unit subparameter is 3350A (a subset of 3350), then the system allocates from the devices contained in 3350A.

DD

## Relationship to Other Parameters

Do not code the following `DD` parameters with the `UNIT` parameter.

*	DLM
DATA	DYNAM
DDNAME	QNAME

Do not code the `UNIT DEFER` subparameter on a `SYSCKEOV` `DD` statement.

***UNIT = AFF and Other Parameters:*** Do not code `DISP = NEW` with `UNIT = AFF = ddname` if the referenced data set for `DD` statement `ddname` resides on a direct access device. If coded, the system terminates the job. If the referenced data set can be allocated to either tape or direct access devices, the system allocates both requests to tape devices.

Do not code `UNIT = AFF` on a `DD *` or `DD DATA` statement. The system ignores `UNIT = AFF` and defaults the device to `SYSALLDA`.

Do not code `UNIT = AFF = ddname` when `DD` statement `ddname` contains `FREE = CLOSE`.

## Location in the JCL

When a `DD` statement contains a `UNIT = AFF = ddname` parameter, the `DD` statement `ddname` must appear earlier in the job step. For example:

```
//STEP EXEC PGM=TKM
//DD1 DD DDNAME=DD5
//DD2 DD DSNAME=A,DISP=OLD
//DD3 DD DSNAME=C,DISP=SHR,UNIT=AFF=DD1
//DD5 DD DSNAME=B,DISP=SHR
```

## DD: UNIT

### Examples of the UNIT Parameter

```
//STEP2 EXEC PGM=POINT
//DDX DD DSNAME=EST,DISP=MOD,VOLUME=SER=(42569,42570),
// UNIT=(3330,2)
//DDY DD DSNAME=ERAS,DISP=OLD,UNIT=3330-1
//DDZ DD DSNAME=RECK,DISP=OLD,
// VOLUME=SER=(40653,13262),UNIT=AFF=DDX
```

DD statement DDX requests two 3330 Disk Storage. DD statement DDZ requests the same two devices as DDX. Note that the operator will have to change volumes on the two 3330 devices during execution of the job step.

DD statement DDY requests one 3330 Disk Storage Model 11.

---

```
//DD1 DD DSNAME=AAG3,DISP=(,KEEP),
// VOLUME=SER=13230,UNIT=3400-5
```

This DD statement defines a new data set and requests that the system assign any 3420 Magnetic Tape Unit that can operate in 800 BPI NRZI nine-track format.

---

```
//DD2 DD DSNAME=X.Y.Z,DISP=OLD,UNIT=(,2)
```

This DD statement defines a cataloged data set and requests that the system assign two devices to the data set. The system obtains the device type from the catalog.

---

```
//DD3 DD DSNAME=COLLECT,DISP=OLD,
// VOLUME=SER=1095,UNIT=(3330,,DEFER)
```

This DD statement defines an existing data set that resides on a direct access volume and requests that the system assign a 3330 Disk Storage. Because DEFER is coded, the volume will not be mounted until the data set is opened.

---

```
//STEP4 DD DSNAME=FALL,DISP=OLD,UNIT=237
```

For this data set, the system retrieves the volume and device type from the catalog. The UNIT parameter, by specifying device 237, overrides the catalog device type; however, device 237 must be the same type of device as stated in the catalog.

## VOLUME Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the VOLUME parameter to identify the volume or volumes on which a data set resides or will reside. You can request:

- A private volume
- Retention of the volume
- A specific volume by serial number
- The same volume that another data set uses

**References:** For more information on the VOLUME parameter, see “Requesting Units and Volumes” on page 7-24.

**Syntax:**

DD

```
{VOLUME}=( [PRIVATE] [,RETAIN] [,volume-sequence-number] [,volume-count]
{VOL   }      [,      ][,      ]

          [SER=(serial-number[,serial-number]...)] )
          [REF=dsname                               ] )
          [,][REF=*.ddname                           ] )
          [REF=*.stepname.ddname                     ] )
          [REF=*.stepname.procstepname.ddname       ] )
          [REF=*.procstepname.ddname                 ] )
```

**Single Subparameter:** You can omit the parentheses if you code only PRIVATE or only a keyword subparameter. For example, VOLUME=PRIVATE or VOLUME=SER=222001 or VOLUME=REF=DS1.

**Positional Subparameters:** The first four subparameters are positional. The last subparameter, SER or REF, is a keyword subparameter and must follow all positional subparameters. Code a comma to indicate an omitted positional subparameter as follows:

- If you omit PRIVATE, do not code a comma in its place. For example, VOLUME=(RETAIN,2,3,SER=(222001,222002,222003)).
- Code a comma when RETAIN is omitted and the volume sequence number or the volume count subparameter follows. For example, VOLUME=(PRIVATE,,2,3,SER=(222001,222002,222003)).
- Code a comma when the volume sequence number is omitted and the volume count subparameter follows. For example, VOLUME=(RETAIN,,3,SER=(222001,222002,222003)) and VOLUME=(PRIVATE,,,3,SER=(222001,222002,222003)) and VOLUME=(,,3,SER=(222001,222002,222003)).
- Code a comma when the volume count is omitted, at least one other subparameter precedes it, and a keyword subparameter follows. For example, VOLUME=(RETAIN,2,,SER=(222001,222002,222003)).

## DD: VOLUME

**Single SER Subparameter:** You can omit the parentheses in the SER subparameter if you code only one serial number. For example, `VOLUME = SER = 222001`.

**Special Characters** When a serial number in the SER subparameter contains special characters, other than hyphens, enclose it in apostrophes. For example, `VOLUME = SER = (222001,222-02,'222/03')`.

When the dsname in the REF subparameter contains special characters, other than the periods used in a qualified name, enclose it in apostrophes. For example, `VOLUME = REF = 'DS/284'`.

Code each apostrophe that is part of the serial number or data set name as two consecutive apostrophes. For example, `VOLUME = SER = 'O''HARE'` or `VOLUME = REF = 'DS''371'`.

### Subparameter Definition

#### PRIVATE

Requests a private volume. Private means that:

- The system is not to allocate an output data set to the volume unless the volume is specifically requested, such as in a `VOLUME = SER` subparameter.
- If tape, the volume is to be demounted after the data set is closed, unless `RETAIN` is also coded or the `DD DISP` parameter specifies `PASS`.
- If a demountable direct access volume, the volume is to be demounted after the data set is closed.

#### RETAIN

For a private tape volume, `RETAIN` requests that this volume is not to be demounted or rewound after the data set is closed or at the end of the step. For a public tape volume, `RETAIN` requests that this volume is to be retained at the device if it is demounted during the job.

*Note:* `RETAIN` is supported only by the basic control program and by JES2. If coded on a `DD` statement processed by JES3, it is ignored. However, the comma to indicate its omission is always necessary.

`RETAIN` has no effect on the handling of direct access volumes.

Coding `RETAIN` does not ensure that the operator will not unload the volume or that the system will not deallocate and demount it for another job. Either can occur when the device on which the volume is mounted is not allocated to the job step containing the `DD` statement that specified `RETAIN` or, for unlabeled tapes, when the volume requires verification.

#### volume-sequence-number

Identifies the volume of an existing multivolume data set to be used to begin processing the data set. The volume sequence number is a decimal number from 1 through 255; the first volume is identified as 1.

The volume sequence number must be less than or equal to the number of volumes on which the data set exists.

For new data sets, the system ignores the volume sequence number.

Normally, you code a volume sequence number when you do not specify volume serial numbers on the DD statement, for example, when you are retrieving a cataloged data set or you are referring to an earlier DD statement or data set. If you code both a volume sequence number and a volume serial number, the system begins processing the data set with the volume that corresponds to the volume sequence number.

If you code a volume sequence number and request specific volumes, the volume sequence number must not exceed the number of volume serial numbers or the job fails.

**volume-count**

Specifies the maximum number of volumes that an output data set requires. The volume count is a decimal number from 1 through 255.

The total volume count for all DD statements in one job step cannot exceed 4095.

Code the volume count for a new data set that will reside on 6 or more volumes. If a volume count of 1 through 5 is specified, the maximum number allowed is 5; if a count of 6 through 20 is specified, the maximum number allowed is 20; for a volume count greater than 20, the maximum number allowed is a multiple of 15 plus 5, up to a maximum of 255.

If the volume count is greater than the number of specified volume serial numbers, the system assigns other volumes to the remaining devices. If the volume count is smaller than the number of specified volume serial numbers, the system ignores the volume count.

If a data set might require more volumes than the number of specified volume serial numbers, specify in the volume count subparameter the total number of volumes that might be used. By requesting more volumes in the volume count subparameter, you can ensure that the data set can be written on more volumes if it exceeds the requested volumes.

If the request is for a nonspecific, nonprivate volume on a direct access device, the system ignores the volume count and allocates the number of volumes in the UNIT unit count subparameter.

If the request is for a nonspecific, private volume, the system treats it like a specific request if more than one volume is needed and allocates the number of volumes in the volume count subparameter.

For more information on determining the number of volumes per request, see *SPL: Job Management*.

**SER = (serial-number[,serial-number]...)**

Identifies by serial number the volume(s) on which the data set resides or will reside. A volume serial number is 1 through 6 alphanumeric, national, or special characters; enclose a serial number that contains special characters, other than hyphens, in apostrophes. If the number is shorter than 6 characters, it is padded with trailing blanks.

You can code a maximum of 255 volume serial numbers on a DD statement.

Do not specify duplicate volume serial numbers in a SER parameter. Each volume must have a unique volume serial number, regardless of whether it is a tape and disk volume.

## DD: VOLUME

Do not code a volume serial number as SCRTCH, PRIVAT, or Lnnnnn (L with five numbers); these are used in messages to ask the operator to mount a volume.

When using some typewriter heads or printer chains, a volume serial number may be unrecognizable if you code certain special characters.

**REF = dsname**

**REF = \*.ddname**

**REF = \*.stepname.ddname**

**REF = \*.stepname.procstepname.ddname**

**REF = \*.procstepname.ddname**

Tells the system to obtain volume serial numbers from another data set or an earlier DD statement.

**Referenced Data Set Not Opened:** When REF refers to a DD statement in a previous step and the data set was not opened, the system allocates a device that has the widest range of eligibility to meet both DD statement requests. Thus, the system might allocate a device for which the referring data set is not eligible. To prevent this problem for tape data sets, always code the DCB DEN subparameter on a DD statement that you plan to reference.

**References to Multivolume Tape Data Sets:** When REF refers to a data set that resides on more than one tape volume, the system allocates only the last volume. If this job step extends the the data set to more volumes, this new volume information is not available to following DD statements.

**References to Multivolume Direct Access Data Sets:** When REF refers to a data set that resides on more than one direct access volume, the system allocates all of the volumes.

**References to DD Statements with UNIT Group Names:** When REF refers to a DD statement containing a UNIT group-name subparameter, the system allocates a device of the same type actually used for the referenced data set, but not necessarily a device in the referenced group-name.

**References to VSAM Data Sets:** When REF refers to a multivolume VSAM data set, the system allocates a device of the same type as the first device type used for the referenced VSAM data set.

**Do Not Refer to In-Stream Data Sets:** Do not refer to a DD \*, DD DATA, or DD SYSOUT statement. The system ignores the reference and defaults the device name to SYSALLDA, which is the system-defined group name for all direct access devices defined to the system.

**References to DUMMY Data Sets:** If ddname refers to a DD DUMMY statement, the data set for this DD statement is also assigned a dummy status.

**Other DD Parameter Picked up from Referenced Statement:** When REF is coded, the system also copies the LABEL label type subparameter from the referenced DD statement.

**dsname**

Names a cataloged or passed data set. The system assigns this data set to the same volumes containing the cataloged or passed data set. The dsname cannot be a generation data group (GDG) name or a GDG member.

When the dsname contains special characters, other than the periods used in a qualified name, enclose it in apostrophes.

**\*.ddname**

Asks the system to obtain the volume serial numbers from earlier DD statement ddname in the same job step.

**\*.stepname.ddname**

Asks the system to obtain the volume serial numbers from DD statement, ddname, in an earlier step, stepname, in the same job.

**\*.stepname.procstepname.ddname**

Asks the system to obtain the volume serial numbers from a DD statement in a cataloged on in-stream procedure. Stepname is the name of the job step that calls the procedure, procstepname is the name of the procedure step that contains the DD statement, and ddname is the name of the DD statement.

**\*.procstepname.ddname**

Asks the system to obtain the volume serial numbers from a DD statement in a cataloged or in-stream procedure called by the current job step. Procstepname is the name of the procedure step that contains the DD statement and ddname is the name of the DD statement.

## Overrides

The volume sequence number overrides a DISP=MOD parameter. Thus, instead of starting at the end of the data set on the last volume, according to the MOD subparameter, processing of the data set begins with the volume indicated by the volume sequence number.

## Relationship to Other Parameters

Do not code the following parameters with the VOLUME parameter.

BURST	DDNAME	MODIFY	QNAME
CHARS	DYNAM	MSVGP	SYSOUT
COPIES	FLASH	OUTPUT	

**Other DD Parameter Picked up from Referenced Statement:** When REF is coded, the system also copies the LABEL label type subparameter from the referenced DD statement.

### *With Mass Storage Volumes For New Data Sets*

- The SPACE parameter is required when VOLUME = SER is coded.
- To guarantee allocation to SYSGROUP for a nonspecific request, specify VOLUME = PRIVATE or MSVGP = SYSGROUP.

**For 3540 Diskette Input/Output Units:** The VOLUME = SER, DCB = BUFNO, and DSID parameters on a DD \* or DD DATA statement are ignored except when they are detected by a diskette reader as a request for an associated data set. See *IBM 3540 Programmer's Reference*.

## DD: VOLUME

### VOLUME Information for a Checkpoint/Restart Data Set

When a checkpoint data set is not cataloged, code on the SYSCHK DD statement the `VOLUME=SER` parameter to specify the serial number for the volume on which the checkpoint entry is written.

If a checkpoint data set is cataloged, you can omit the `VOLUME` parameter unless the checkpoint entry is on a tape volume other than the first volume of the data set; then, code either a volume sequence number or the volume serial number. If you code the volume serial number, you must also code the `UNIT` parameter.

### VOLUME Parameter in a JES3 System

When you do not code a volume serial number, code `PRIVATE` if you want JES3 to manage the allocation. Otherwise, MVS manages the allocation.

`RETAIN` is ignored in a JES3 system.

### VOLUME Parameter for Optical Readers

For optical readers, if no volume serial number is specified, the system assumes `VOLUME=SER=OCRINP`.

### Examples of the VOLUME Parameter

```
//DD1 DD DSNAME=DATA3,UNIT=3350,DISP=OLD,  
//      VOLUME=(PRIVATE,SER=548863)
```

The DD statement requests an existing data set, which resides on the direct access volume, serial number 548863. Since `PRIVATE` is coded, the system will not assign to the volume another data set for which a nonspecific volume request is made and will demount the volume at the end of the job.

---

```
//DD2 DD DSNAME=QUET,DISP=(MOD,KEEP),UNIT=(3400-5,2),  
//      VOLUME=(, , ,4,SER=(96341,96342))
```

The DD statement requests an existing data set, which resides on two volumes, serial numbers 96341 and 96342. The `VOLUME` volume count subparameter requests four volumes, if required. Thus, if more space is required, the system can assign a third and fourth volume.

---

```
//DD3 DD DSNAME=QOUT,DISP=NEW,UNIT=3400-5
```

The DD statement defines a temporary data set. By omission of the `VOLUME` parameter, the statement requests that the system assign a suitable volume to the data set.

---



```
//OUTDD DD DSN=TEST.TWO,DISP=(NEW,CATLG),
//        VOLUME=(, , 3,SER=(333001,333002,333003)),
//        SPACE=(TRK,(9,10)),UNIT=(3330,P)
//NEXT DD DSN=TEST.TWO,DISP=(OLD,DELETE)
```

DD statement OUTDD creates a multivolume data set and catalogs it. If the data set does not require three volumes, it will reside on fewer volumes. DD statement NEXT then deletes the data set.

If the data set resides on fewer volumes than the number of volumes on which it is cataloged, the following messages appear in the job log when the system deletes the data set:

```
IEF285I TEST.TWO DELETED
IEF285I VOL SER NOS=333001,333003.
IEF283I TEST.TWO NOT DELETED 8
IEF283I VOL SER NOS=333002 1.
IEF283I TEST.TWO UNCATALOGED
IEF283I VOL SER NOS=333001,333002,333003.
```

If the data set resides on all specified volumes, the following messages appear in the job log when the system deletes the data set:

```
IEF285I TEST.TWO DELETED
IEF285I VOL SER NOS=333001,333002,333003.
```



## Chapter 13. Coding Special DD Statements

Use special DD statements to specify private catalogs, private libraries, and data sets for storage dumps and checkpoints.

These special statements are arranged alphabetically in the following pages. For more information on these statements, see “Creating and Using Private and Temporary Libraries” on page 8-1 and “Requesting an Abnormal Termination Dump” on page 8-6.

**Syntax:**

```
//ddname DD keyword-parameter[,keyword-parameter]... comments
```

Spec DD

The special data sets are identified by the following ddnames:

JOBCAT  
JOBLIB  
STEPCAT  
STEPLIB  
SYSABEND  
SYSCHK  
SYSCKEOV  
SYSMDUMP  
SYSUDUMP

Code these ddnames only when you want the special data sets.

# JOB CAT DD

## JOB CAT DD Statement

**Purpose:** Use the JOB CAT DD statement to define a private VSAM or ICF user catalog for the duration of a job. The system searches the private user catalog for data sets before it searches the master catalog or a private catalog associated with the first qualifier of a data set's name.

You *cannot* specify OS CVOLs as JOB CAT. Access to an OS CVOL is possible only with a CVOL pointer in the master catalog.

**References:** For more information on VSAM data sets, see *VSAM Programmer's Guide*.

**Syntax:**

```
//JOB CAT DD DISP={OLD|SHR},DSNAME=user-catalog-name[,parameter]...
```

### Parameters on JOB CAT DD Statements

Do not specify any unit or volume information. The system obtains the location of the private catalog from the master catalog.

### Relationship to STEPCAT DD Statement

A JOB CAT DD statement applies to any job step for which you do not specify a STEPCAT DD statement.

### Relationship to Other Control Statements

**Concatenating Job Catalogs:** To specify more than one user catalog for a job:

- Code a JOB CAT DD statement.
- Immediately follow this statement with DD statements that define other private catalogs. Omit a ddname from these subsequent DD statements.

### Location in the JCL

- Place the JOB CAT DD statement *after* the JOB statement and *before* the first EXEC statement.
- Place a JOBLIB DD statement before a JOB CAT DD statement.

## Example of the JOBCAT DD Statement

```
//EXAMPLE JOB WILLIAMS,MSGLEVEL=1
//JOBLIB DD DSNAME=USER.LIB,DISP=SHR
//JOBCAT DD DSNAME=LYLE,DISP=SHR
// EXEC PGM=SCAN
```

In this example, the JOBCAT DD statement specifies a private catalog. The JOBCAT DD statement follows the JOBLIB DD statement.

## JOBLIB DD

### JOBLIB DD Statement

**Purpose:** Use the JOBLIB DD statement to define a private library that the system is to search for the program named in each EXEC statement PGM parameter. If the system does not find the program in the private library, only then does the system search the system libraries.

**References:** For more information on private libraries, see "Creating and Using Private and Temporary Libraries" on page 8-1.

**Syntax:**

```
//JOBLIB DD parameter[,parameter]...
```

### Parameters on JOBLIB DD Statements

**When the Library is Cataloged:**

- Code the DSNNAME parameter.
- Code the DISP parameter. The status subparameter must be OLD or SHR. The disposition subparameters should indicate what you want done with the private library after its use in the job step.
- Do not code VOLUME = SER.

**When the Library is Not Cataloged:**

- Code the DSNNAME parameter unless the disposition is DISP = (NEW,PASS).
- Code the DISP parameter. The DISP parameter must be one of the following:

```
DISP = (NEW,PASS)  
DISP = (NEW,CATLG)  
DISP = (OLD,PASS)  
DISP = (SHR,PASS)
```

- Code the UNIT parameter.
- Code the VOLUME parameter, if the status of the data set is OLD or SHR.
- Code the SPACE parameter, if the status of the data set is NEW.

**Other Parameters:** Code the DCB parameter if complete data control block information is not contained in the data set label.

## Relationship to Other Control Statements

**Relationship to Later DD Statements:** Code `VOLUME=REF=*.JOBLIB` to obtain volume and unit information when a later DD statement defines a data set that the system is to place on the same volume as the private library.

To refer to the private library in a later DD statement, code `DSNAME=*.JOBLIB,VOLUME=REF=*.JOBLIB` (or `VOLUME=SER=serial number,UNIT=unit information`), and the `DISP` parameter, `DISP=(OLD,PASS)`.

**Concatenating Step Libraries:** To specify more than one private library for a step:

- Code a STEPLIB DD statement.
- Immediately follow this statement with DD statements that define other private libraries. Omit a ddname from these subsequent DD statements.

**Overriding a JOBLIB:** If you want the system to ignore the JOBLIB for a particular job step and the step does not require another private library, define the system library on a STEPLIB DD statement. For example:

```
//STEPLIB DD DSNAME=SYS1.LINKLIB,DISP=SHR
```

For this particular job step, the system will search `SYS1.LINKLIB` as specified on the STEPLIB DD statement for the program requested in the EXEC statement. The system will not search the JOBLIB.

**EXEC Statement COND Parameter:** If `COND=ONLY` is specified on the EXEC statement of a job step and a JOBLIB DD statement is being used, the system does not pass the unit and volume information to any succeeding steps, and the system must search the catalog for the JOBLIB data set's unit and volume information.

## Location in the JCL

- The JOBLIB DD statement must immediately follow the JOB statement and any JES statements. There must be no intervening EXEC or other DD statements between the JOBLIB DD statement and the JOB statement.
- Do not include a JOBLIB DD statement in a cataloged procedure.

## Relationship of a JOBLIB to a STEPLIB

Use a STEPLIB DD statement to define a private library for one job step in a job. If you include a STEPLIB DD statement for a job step and a JOBLIB DD statement for the entire job, the system first searches the step library and then the system library for the requested program. The system ignores the job library for the step that has a STEPLIB DD statement.

# JOBLIB DD

## Examples of the JOBLIB DD Statement

```
//PAYROLL JOB JONES,CLASS=C
//JOBLIB DD DSNAME=PRIVATE.LIB4,DISP=(OLD,PASS)
//STEP1 EXEC PGM=SCAN
//STEP2 EXEC PGM=UPDATE
//DD1 DD DSNAME=*.JOBLIB,DISP=(OLD,PASS)
```

The private library requested on the JOBLIB DD statement is cataloged. The system passes catalog information to subsequent job steps. The system searches for the programs SCAN and UPDATE first in PRIVATE.LIB4, then in SYS1.LINKLIB. DD statement DD1 refers to the private library requested in the JOBLIB DD statement.

---

```
//PAYROLL JOB FOWLER,CLASS=L
//JOBLIB DD DSNAME=PRIV.DEPT58,DISP=(OLD,PASS),
// UNIT=3350,VOLUME=SER=D58PVL
//STEP EXEC PGM=DAY
//STEP2 EXEC PGM=BENEFITS
//DD1 DD DSNAME=*.JOBLIB,VOLUME=REF=*.JOBLIB,
// DISP=(OLD,PASS)
```

The private library requested on the JOBLIB DD statement is not cataloged; therefore, unit and volume information is specified. The system searches for the programs DAY and BENEFITS first in PRIV.DEPT58, then in SYS1.LINKLIB. DD statement DD1 refers to the private library requested in the JOBLIB DD statement.

---

```
//TYPE JOB MSGLEVEL=(1,1)
//JOBLIB DD DSNAME=GROUP8.LEVEL5,DISP=(NEW,CATLG),
// UNIT=3350,VOLUME=SER=148562,
// SPACE=(CYL,(50,3,4))
//STEP1 EXEC PGM=DISC
//DDA DD DSNAME=GROUP8.LEVEL5(RATE),DISP=MOD,
// VOLUME=REF=*.JOBLIB
//STEP2 EXEC PGM=RATE
```

The private library requested on the JOBLIB DD statement does not exist yet; therefore, the JOBLIB DD statement contains all the parameters required to define the library. The library is created in STEP1, when DD statement DDA defines the new member RATE for the library. Therefore, the system searches SYS1.LINKLIB for the program named DISC. In STEP2, the system searches for the program RATE first in GROUP8.LEVEL5.

---

```
//PAYROLL JOB BIRDSALL,TIME=1440
//JOBLIB DD DSNAME=KRG.LIB12,DISP=(OLD,PASS)
// DD DSNAME=GROUP31.TEST,DISP=(OLD,PASS)
// DD DSNAME=PGMSLIB,UNIT=3350,
// DISP=(OLD,PASS),VOLUME=SER=34568
```

The three DD statements concatenate the three private libraries. The system searches the libraries for each program in this order:

```
KRG.LIB12
GROUP31.TEST
PGMSLIB
SYS1.LINKLIB
```



## STEPCAT DD Statement

**Purpose:** Use the STEPCAT DD statement to define a private VSAM or ICF user catalog for the duration of a job step. The system searches the private catalog for data sets before it searches the master catalog or a private catalog associated with the first qualifier of a data set's name.

You *cannot* specify OS CVOLs as STEPCAT. Access to an OS CVOL is possible only with a special CVOL pointer in the master catalog.

**References:** For more information on VSAM data sets, see *VSAM Programmer's Guide*.

**Syntax:**

```
//STEPCAT DD DISP={OLD|SHR},DSNAME=user-catalog-name[,parameter]...
```

### Parameters on STEPCAT DD Statements

Spec DD

Do not specify any unit or volume information. The system obtains the location of the user catalog from the master catalog.

### Relationship to Other Control Statements

**Concatenating Step Catalogs:** To specify more than one user catalog for a step:

- Code a STEPCAT DD statement.
- Immediately follow this statement with DD statements that define other private catalogs. Omit a ddname from these subsequent DD statements.

**Overriding a JOBCAT:** To override a JOBCAT private catalog with the master catalog for a particular job step, code the following in the job step:

```
//STEPCAT DD DISP=OLD,DSNAME=master-catalog-name
```

### Location in the JCL

Place a STEPCAT DD statement in any position among the DD statements for a step.

### Example of the STEPCAT DD Statement

```
//          EXEC PROC=SNZ12
//STEPCAT DD  DSNAME=BETTGER,DISP=SHR
```

The STEPCAT DD statement *immediately* following an EXEC statement specifies a private catalog that the system uses for this job step only.

## STEPLIB DD

### STEPLIB DD Statement

**Purpose:** Use the STEPLIB DD statement to define a private library that the system is to search for the program named in the EXEC statement PGM parameter. If the system does not find the program in the private library, only then does the system search the system libraries.

Subsequent job steps in the same job may refer to or receive a private library defined on a STEPLIB DD statement. Also, you can place a STEPLIB DD statement in a cataloged procedure.

**References:** For more information on using private libraries, see “Creating and Using Private and Temporary Libraries” on page 8-1.

**Syntax:**

```
//STEPLIB DD parameter[,parameter]...
```

### Parameters on STEPLIB DD Statements

**When the Library is Cataloged:**

- Code the DSNAME parameter.
- Code the DISP parameter. The status subparameter must be either OLD or SHR. The disposition subparameters should indicate what you want done with the private library after its use in the job step.
- Do not code VOLUME = SER.

**When the Library is Passed from a Previous Step:**

- In the passing job step, code a DISP disposition subparameter of PASS when a step library is to be used by subsequent steps in the job.
- In a receiving step, code the DISP parameter. The status subparameter must be OLD. The disposition subparameters should indicate what you want done with the private library after its use in the job step.

**When the Library is Neither Cataloged Nor Passed:**

- Code the DSNAME parameter.
- Code the DISP parameter. The status subparameter must be OLD or SHR. The disposition subparameters should indicate what you want done with the private library after its use in the job step.
- Code the UNIT parameter.
- Code the VOLUME parameter.

**Other Parameters:** Code the DCB parameter if complete data control block information is not contained in the data set label.

## Relationship to Other Control Statements

**Concatenating Step Libraries:** To specify more than one private library for a step: of DD statements that define different data sets:

- Code a STEPLIB DD statement.
- Immediately follow this statement with DD statements that define other private libraries. Omit a ddname from these subsequent DD statements.

**Overriding a JOBLIB:** If you want the system to ignore the JOBLIB for a particular job step and the step does not require another private library, define the system library on a STEPLIB DD statement. For example:

```
//STEPLIB DD DSN=SYS1.LINKLIB,DISP=SHR
```

For this particular job step, the system will first search SYS1.LINKLIB as specified on the STEPLIB DD statement for the program requested in the EXEC statement. The system will not search the JOBLIB.

Spec Df

**References to a Previous Step Library:** To refer to a previously defined step library:

- In the DSN parameter, code either the name of the step library or a backward reference of the form \*.stepname.STEPLIB. If the step library was defined in a cataloged procedure, the backward reference must include the procedure step name: \*.stepname.procstepname.STEPLIB.
- Code the DISP parameter. The status subparameter must be OLD. The disposition subparameters should indicate what you want done with the private library after its use in the job step.

## Location in the JCL

Place a STEPLIB DD statement in any position among the DD statements for a step.

## Relationship of a STEPLIB to a JOBLIB

Use a JOBLIB DD statement to define a private library that the system is to use for an entire job. If you include a JOBLIB DD statement for the job and a STEPLIB DD statement for an individual job step, the system first searches the step library and then the system library for the program requested in the EXEC statement. The system ignores the JOBLIB library for that step.

# STEPLIB DD

## Examples of the STEPLIB DD Statement

```
//PAYROLL JOB BROWN,MSGLEVEL=1
//STEP1 EXEC PROC=LAB14
//STEP2 EXEC PGM=SPKCH
//STEPLIB DD DSN=PRIV.LIB5,DISP=(OLD,KEEP)
//STEP3 EXEC PGM=TIL80
//STEPLIB DD DSN=PRIV.LIB12,DISP=(OLD,KEEP)
```

The system searches PRIV.LIB5 for the program SPKCH and PRIV.LIB12 for TIL80. The system catalogs both private libraries.

---

```
//PAYROLL JOB BAKER,MSGLEVEL=1
//JOB LIB DD DSN=LIB5.GROUP4,DISP=(OLD,PASS)
//STEP1 EXEC PROC=SNZ12
//STEP2 EXEC PGM=SNAP10
//STEPLIB DD DSN=LIBRARYP,DISP=(OLD,PASS),
// UNIT=3350,VOLUME=SER=55566
//STEP3 EXEC PGM=A1530
//STEP4 EXEC PGM=SNAP11
//STEPLIB DD DSN=*.STEP2.STEPLIB,
// DISP=(OLD,KEEP)
```

The system searches LIBRARYP for program SNAP10; LIBRARYP is passed to subsequent steps of this job. The STEPLIB DD statement in STEP4 refers to the LIBRARYP library defined in STEP2; the system searches LIBRARYP for SNAP11. Since a JOBLIB DD statement is included, the system searches for programs SNZ12 and A1530 first in LIB5.GROUP4, then in SYS1.LINKLIB.

---

```
//PAYROLL JOB THORNTON,MSGLEVEL=1
//JOB LIB DD DSN=LIB5.GROUP4,DISP=(OLD,PASS)
//STEP1 EXEC PGM=SUM
//STEPLIB DD DSN=SYS1.LINKLIB,DISP=OLD
//STEP2 EXEC PGM=VARY
//STEP3 EXEC PGM=CALC
//STEPLIB DD DSN=PRIV.WORK,DISP=(OLD,PASS)
// DD DSN=LIBRARYA,DISP=(OLD,KEEP),
// UNIT=3350,VOLUME=SER=44455
// DD DSN=LIB.DEPT88,DISP=(OLD,KEEP)
//STEP4 EXEC PGM=SHORE
```

For STEP2 and STEP4, the system searches the private library named LIB5.GROUP4 defined in the JOBLIB DD statement first for programs VARY and SHORE. For STEP1, the system searches SYS1.LINKLIB first for program SUM, because the STEPLIB DD statement names the system library.

A concatenation of private libraries is defined in STEP3. The system searches for the program named CALC in this order: PRIV.WORK, LIBRARYA, LIB.DEPT88, SYS1.LINKLIB. If a later job step refers to the STEPLIB DD statement in STEP3, the system will search for the program in the private library named PRIV.WORK and, if it is not found there, in SYS1.LINKLIB; the concatenated libraries are not searched.

## SYSABEND, SYSMDUMP, and SYSUDUMP DD Statements

**Purpose:** Use a SYSABEND, SYSMDUMP, or SYSUDUMP DD statement in a job step to direct the system to produce a dump. The system produces the requested dump:

- If the step abnormally terminates.
- If the step ABENDs but system recovery procedures enable the step to terminate normally.

The DD statements for the requested dumps are:

### SYSABEND DD statement

Produces a formatted dump that includes only the processing program storage area.

### SYSMDUMP DD statement

Produces an unformatted, machine-readable dump that includes the system nucleus and the processing program storage area.

### SYSUDUMP DD statement

Produces a formatted dump that includes the processing program storage area.

For SYSABEND and SYSUDUMP dumps, the installation determines which of the storage areas addressable by the processing program are to be included in the dump. For more information, see *SPL: Initialization and Tuning Guide*

**References:** For information on how to interpret dumps, see *Debugging Handbook* and *Diagnostic Techniques*.

### Syntax:

```
//SYSABEND DD parameter[,parameter]...
//SYSMDUMP DD parameter[,parameter]...
//SYSUDUMP DD parameter[,parameter]...
```

### Location in the JCL

Do not place in the same job step two DD statements with the same dump ddname.

### Storing a Dump

If you wish to store a dump instead of having it printed, code the following parameters on the dump DD statement:

- The DSNAME parameter.
- The UNIT parameter.
- The VOLUME parameter.
- The DISP parameter. The data set's status is NEW. Because you want to store the data set, make the data set's abnormal termination disposition KEEP or CATLG.

# SYSABEND, SYSMDUMP, SYSUDUMP DD

- The SPACE parameter, if the dump is written on direct access.

The SYSMDUMP DD statement must specify a magnetic tape unit, a direct access device, or a sysout data set. If the job or step is running in nonpageable virtual storage (ADDRSPC=REAL on the JOB or EXEC statement), the SYSMDUMP DD statement must specify a virtual I/O (VIO) data set.

To write more than one SYSMDUMP dump in the same data set on tape, specify the following:

- DSNAME=SYS1.SYSMDPxx where xx is 00 through FF. SYSMDPxx is a preallocated data set that you must initialize with an end-of-file (EOF) mark on the first record.
- DISP=SHR

You can ask the system to write additional dumps only if you off-load any previous dump and write an EOF mark at the beginning of the SYS1.SYSMDPxx data set. To accomplish this, your installation must install an exit routine for message IEA993. For information on this exit routine, see *SPL: Supervisor*.

## Printing a Dump

To print a dump for either a SYSABEND or SYSUDUMP DD statement, code one of the following on the DD statement for the output data set:

- The UNIT parameter.
- The SYSOUT parameter with the output class subparameter.

To print a dump for a SYSMDUMP DD statement, use one of the following programs:

### AMDPRDMP service aid

This program is described in *SPL: Service Aids*. When using AMDPRDMP, the SYSMDUMP DD statement must allocate the dump data set to a magnetic tape or a direct access device.

If the job or job step is running in nonpageable virtual storage (ADDRSPC=REAL on the JOB or EXEC statement), the SYSMDUMP DD statement must specify a virtual I/O (VIO) data set.

### IPCS

This program is described in *Interactive Problem Control System (IPCS) System Information*. When using IPCS, the data set disposition affects the collection of events. See "Requesting an Abnormal Termination Dump" on page 8-6.

If you print the dump on a 3800 Printing Subsystem, code CHARS=DUMP for a dump with 204 characters per line and FCB=STD3 for 8 lines per inch.

# SYSABEND, SYSMDUMP, SYSUDUMP DD

## Overriding Dump DD Statements

To override a dump DD statement in a cataloged or in-stream procedure, use a DD statement with a ddname that is different from the ddname of the dump DD statement in the procedure.

## Duplicate Dump Requests

You can code more than one dump request in a job step using DD statements that have different ddnames. When you do this, the system uses the last dump DD statement it encounters.

When the system finds dump DD statements with duplicate ddnames, processing is as follows:

- In a JES2 system, the job fails with message IEC912I.
- In a JES3 system:
  - If both DD statements request JES3- or jointly-managed devices, the job is cancelled during JES3 interpretation.
  - If only one or neither statement requests JES3- or jointly-managed devices, the job fails with message IEC912I.

Spec DD

## Examples of the SYSABEND, SYSMDUMP, and SYSUDUMP DD Statements

```
//STEP2 EXEC PGM=A
//SYSUDUMP DD SYSOUT=A
```

The SYSUDUMP DD statement specifies that you want the dump routed to system output class A.

---

```
//SYSMDUMP DD DSNAME=DUMP,DISP=(NEW,KEEP),
// UNIT=3400-6,VOLUME=SER=147958
```

The SYSMDUMP DD statement specifies that the dump is to be stored on a tape. Because the LABEL parameter is not coded, the tape must have IBM standard labels.

---

```
//STEP1 EXEC PGM=PROGRAM1
//SYSABEND DD DSNAME=DUMP,UNIT=3350,DISP=(,PASS,KEEP),
// VOLUME=SER=1234,SPACE=(TRK,(40,20))
//STEP2 EXEC PGM=PROGRAM2
//SYSABEND DD DSNAME=*.STEP1.SYSABEND,DISP=(OLD,DELETE,KEEP)
```

Both SYSABEND DD statements specify that the dump is to be stored. The space request in STEP1 is ample and will not inhibit dumping due to insufficient space. If STEP1 does not abnormally terminate but STEP2 does, the system writes the dump for STEP2 in the space allocated in STEP1. In both steps, an abnormal termination disposition of KEEP is specified so that the dump is stored if either of the steps abnormally terminates. If both of the steps successfully execute, the second DISP subparameter, DELETE, in STEP2 instructs the system to delete the data set and free the space acquired for dumping.

---

## SYSABEND, SYSMDUMP, SYSUDUMP DD

```
//STEP1 EXEC PGM=WWK
//SYSUDUMP DD DSNAME=DUMP,UNIT=3350,DISP=(,DELETE,
// KEEP),VOLUME=SER=54366,SPACE=(1680,(160,80))
//STEP2 EXEC PGM=PRINT,COND=ONLY
//IN DD DSNAME=*.STEP1.SYSUDUMP,DISP=(OLD,DELETE),
// VOLUME=REF=*.STEP1.SYSUDUMP
```

STEP1 specifies that the dump is to be stored if the step abnormally terminates. Because the EXEC COND=ONLY parameter is specified in STEP2, STEP2 is executed only if STEP1 abnormally terminates. STEP2 executes a program that prints the dump.

---

```
//STEP EXEC PGM=EXSYSM
//SYSMDUMP DD UNIT=3330,VOLUME=SER=123456,SPACE=(CYL,(0,1)),
// DISP=(NEW,DELETE,KEEP),DSNAME=MDUMP
```

The SYSMDUMP DD statement allocates dump data set MDUMP to a direct access device.

---

```
//STEP EXEC PGM=EXSYSMDP
//SYSMDUMP DD DSNAME=SYS1.SYSMDP00,DISP=SHR
```

The SYSMDUMP is written in data set SYS1.SYSMDP00.

*Note:* When you specify a DSNAME of SYS1.SYSMDPxx and DISP=SHR, the first SYSMDUMP produced is retained on the data set. This first SYSMDUMP must be off-loaded and an end-of-file mark written at the beginning of the SYS1.SYSMDPxx data set before subsequent dumps can be written.

---

```
//STEPA EXEC PGM=EXSYSM2,ADDRSPC=REAL
//SYSMDUMP DD UNIT=SYSDA,SPACE=(CYL,(0,1)),
// DISP=(NEW,PASS)
//STEPB EXEC PGM=AMDPRDMP,COND=ONLY
//SYSUT1 DD DSNAME=*.STEPA.SYSMDUMP,DISP=OLD
//PRINTER DD SYSOUT=A
//SYSIN DD *
        FORMAT
        LOGDATA
        END
/*
```

In STEPA, the SYSMDUMP DD statement directs output to a VIO data set (1) by specifying a VIO-eligible device group (SYSDA) and (2) by not assigning a data set name to make the data set temporary and eligible for VIO. STEPB is executed only if STEPA abnormally terminates. In STEPB, the dump output is printed by the AMDPRDMP service aid on a device assigned to output class A.



## SYSCHK DD Statement

**Purpose:** Use the SYSCHK DD statement to define a checkpoint data set that the system is to write during execution of a processing program. Use this statement again when the step is restarted from a checkpoint written in the data set.

**Note:** If restart is to begin at a step, as indicated by the RD parameter on the EXEC statement, do not use a SYSCHK DD statement.

**References:** See “Restrictions on Use of SYSCHK DD Statement and DD Statement RESTART Parameter” on page v. For detailed information about the checkpoint/restart facilities, see *Checkpoint/Restart*.

### Syntax:

```
//SYSCHK DD parameter[,parameter]...
```

## Parameters on SYSCHK DD Statements

Spec D

### *When the Checkpoint Data Set is Cataloged:*

- Code the DSNAME parameter.
- Code the DISP parameter to specify a status of OLD and a disposition of KEEP.
- Code the UNIT parameter.
- Code the VOLUME parameter. If the checkpoint entry is on a tape volume other than the first volume of the checkpoint data set, code the volume serial number or volume sequence number to identify the correct volume. The serial number of the volume on which a checkpoint entry was written appears in the console message issued after the checkpoint entry is written. If you code the volume serial number, you must also code the UNIT parameter, because the system will not look in the catalog for unit information.

### *When the Checkpoint Data Set is Not Cataloged:*

- Code the DSNAME parameter. If the checkpoint data set is partitioned, do not code a member-name in the DSNAME parameter.
- Code the DISP parameter to specify a status of OLD and a disposition of KEEP.
- Code the UNIT parameter.
- Code the VOLUME parameter. The serial number of the volume on which a checkpoint entry was written appears in the console message issued after the checkpoint entry is written.

# SYSCHK DD

## *Other Parameters:*

- Code the LABEL parameter if the checkpoint data set does not have standard labels.
- Code DCB=TRTCH=C if the checkpoint data set is on 7-track magnetic tape with nonstandard labels or no labels.
- If the volume containing the checkpoint data set is to be mounted on a JES3-managed device, do not code the DEFER subparameter of the UNIT parameter on the SYSCHK DD statement.

## *When Allocating a Checkpoint Data Set:*

- Code a SPACE parameter, but do not request secondary space.
  - The **primary space** request must be large enough to hold all checkpoints. Although your program or the system can write checkpoints in secondary space, the system **cannot** perform a restart from checkpoints in secondary space.
  - If you do **not** request **secondary space** and the primary space fills up, the job abnormally terminates. You can successfully restart the job at the last checkpoint; however, when the processing program or system writes the next checkpoint the job abnormally terminates again.
  - If you **do** request **secondary space** and the primary space fills up, the processing program or the system writes one invalid checkpoint followed by successful checkpoints. An attempt to restart from one of the checkpoints following the invalid checkpoint results in abnormal termination.
- Code the RLSE subparameter of the SPACE parameter only if the processing program opens the checkpoint data set and the checkpoint data set remains open until the end of the program. If you specify RLSE, the system releases unused space after the first CLOSE macro instruction.

### Do **not** code the RLSE subparameter:

- If the processing program opens the checkpoint data set before writing each checkpoint and closes the checkpoint data set after writing each checkpoint. The system releases all unused space while closing the data set after the first checkpoint, leaving no space for additional checkpoints.
- If the system opens the checkpoint data set. The system opens and closes the checkpoint data set before it writes the first checkpoint. With RLSE specified, the system would release all space before the first checkpoint could be written.
- Code the **CONTIG** subparameter of the SPACE parameter to request contiguous space. The system otherwise provides additional primary space using extents. If the extents are **not** contiguous, any checkpoints in these extents cannot be used for a successful restart.

## Relationship to Other Control Statements

Code the RESTART parameter on the JOB statement; without it, the system ignores the SYSCHK DD statement.

## Location in the JCL

- When writing checkpoints, place the SYSCHK DD statement after any JOBLIB DD statements, if coded; otherwise, after the JOB statement.
- When restarting a job from a checkpoint, place the SYSCHK DD statement immediately before the first EXEC statement of the resubmitted job.

## Examples of the SYSCHK DD Statement

```
//JOB1   JOB   RESTART=(STEP3,CK3)
//SYSCHK DD  DSNAME=CHLIB,UNIT=3350,
//          DISP=OLD,VOLUME=SER=456789
//STEP1   EXEC PGM=A
```

The checkpoint data set defined on the SYSCHK DD statement is not cataloged.

```
//JOB2   JOB   RESTART=(STEP2,NOTE2)
//JOBLIB DD DSNAME=PRIV.LIB3,DISP=(OLD,PASS)
//SYSCHK DD DSNAME=CHECKPTS,DISP=(OLD,KEEP),
//          UNIT=3400-6,VOLUME=SER=438291
//STEP1   EXEC PGM=B
```

The checkpoint data set defined on the SYSCHK DD statement is not cataloged. Note that the SYSCHK DD statement follows the JOBLIB DD statement.

## SYSCKEOV DD

### SYSCKEOV DD Statement

**Purpose:** Use the SYSCKEOV DD statement to define a checkpoint data set for checkpoint records from the checkpoint at end-of-volume (EOV) facility. The checkpoint at EOV facility is invoked by a DD CHKPT parameter.

**Note:** You must place a SYSCKEOV DD statement in a job step if you have coded the CHKPT parameter on any DD statements in the step.

**References:** For information on the DD CHKPT parameter, see "CHKPT Parameter" on page 12-21. For information on checkpoint/restart facilities, see *Checkpoint/Restart*.

**Syntax:**

```
//SYSCKEOV DD parameter[,parameter]...
```

### Parameters on SYSCKEOV DD Statements

- Do not code the CHKPT=EOV parameter on the SYSCKEOV DD statement.
- Do not specify the DCB parameter. All DCB information is provided by the checkpoint at EOV facility.
- Do not code the DEFER subparameter of the UNIT parameter.
- If you code the LABEL parameter, you must specify LABEL=(,SL) for IBM standard labels.
- The DISP=MOD parameter is recommended to reduce loss of checkpoint data in case of a system failure during checkpointing.
- The SYSCKEOV DD statement must define a BSAM data set.
- If the SYSCKEOV data set resides on a direct access storage device, that device cannot be shared with another processor.

**When Allocating a Checkpoint Data Set:**

- Code a SPACE parameter, but do not request secondary space. The **primary space** request must be large enough to hold all checkpoints; if not, the job abnormally terminates.
- Do not code the RLSE subparameter of the SPACE parameter.
- Code the CONTIG subparameter of the SPACE parameter to request contiguous space. The system otherwise provides additional primary space using extents.

## Coding SYSCKEOV for VSAM Data Sets

The AMP parameter for VSAM data sets has some restrictions and options. Refer to *Checkpoint/Restart* for a discussion of these restrictions and options.

## Example of the SYSCKEOV DD Statement

```
//SYSCKEOV DD DSNAME=CKPTDS,UNIT=TAPE,DISP=MOD
```

This statement defines a checkpoint data set for checkpoint at EOVS records.



## Chapter 14. Coding the OUTPUT JCL Statement

Figure 14-1 shows job-level and step-level OUTPUT JCL statements in the input stream.

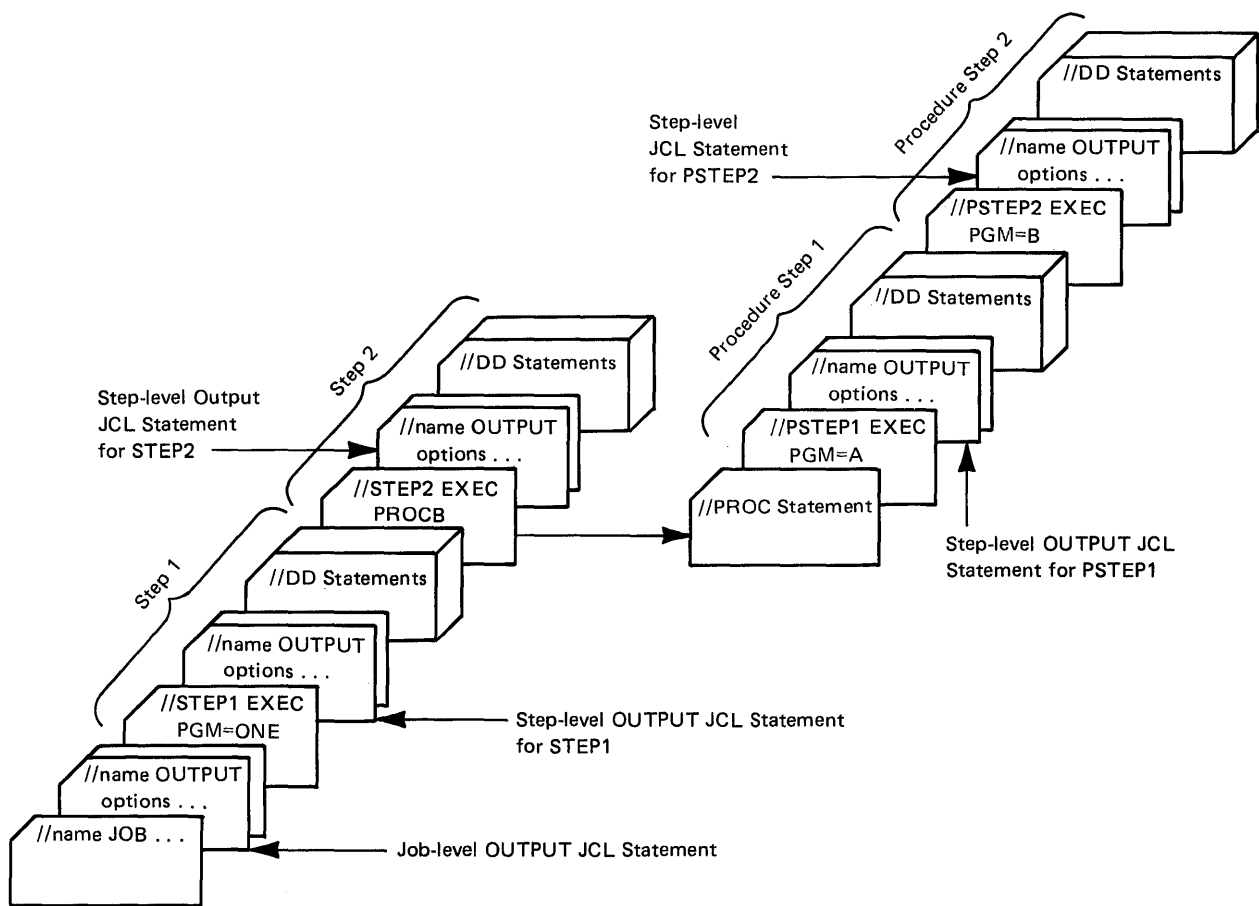


Figure 14-1. Using job- and step-level OUTPUT JCL statements

# OUTPUT JCL

**Purpose:** Use the OUTPUT JCL statement to specify:

- The characteristics of and/or processing options for a specific sysout data set.
- In JES2 installations only, the grouping of sysout data sets for processing by a printer or punch.
- Default processing options for output data sets.
- A destination for an output data set.

The parameters you can specify for sysout data set processing are arranged alphabetically in the following pages.

**References:** For more information on coding OUTPUT JCL statement parameters, see "Processing System Output Data Sets Using the OUTPUT JCL Statement" on page 7-44. For information about the JES initialization parameters that provide installation defaults, see *SPL: JES2 Initialization and Tuning* and *SPL: JES3 Initialization and Tuning*.

**Syntax:**

```
//name      OUTPUT parameter[,parameter]...  comments
```

The OUTPUT JCL statement consists of the characters // in columns 1 and 2 and four fields: name, operation (OUTPUT), parameter, and comments.

## Name Field

Code a name in the name field of every OUTPUT JCL statement, as follows:

- Each OUTPUT JCL name must be unique within the job.
- The name must begin in column 3.
- The name is 1 through 8 alphanumeric or national characters.
- The first character must be alphabetic or national.
- The name must be followed by at least one blank.

## Parameter Field

The OUTPUT JCL statement contains only keyword parameters. All parameters are optional. However, do not leave the parameter field blank. You can code any of the keyword parameters in any order in the parameter field. The following list indicates:

- Parameters supported only in a JES2 system. These parameters cannot be used in a JES3 system.
- Parameters supported only in a JES3 system. These parameters cannot be used in a JES2 system.



Do not use OUTPUT JCL statement parameter keywords as symbolic parameters, names, or labels.

Parameter	JES2 Only, Not JES3	JES3 Only, Not JES2
BURST		
CHARS		
CKPTLINE		
CKPTPAGE		
CKPTSEC		
CLASS		
COMPACT		
CONTROL		
COPIES		
DEFAULT		
DEST		
FCB		
FLASH		
FORMDEF		
FORMS		
GROUPID	X	
INDEX	X	
JESDS		
LINDEX	X	
LINECT	X	
MODIFY		
PAGEDEF		
PIMSG		
PRMODE		
PRTY		
THRESHLD		X
TRC		
UCS		
WRITER		

OUTPUT

**Default OUTPUT JCL Statement:** An OUTPUT JCL statement that contains a DEFAULT=YES parameter is called a default OUTPUT JCL statement.

## Comments Field

The comments field follows the parameter field after at least one intervening blank.

## Location in the JCL

You must place an OUTPUT JCL statement in the input stream before any sysout DD statement that refers to it.

**References by Sysout DD Statements:** An OUTPUT JCL statement can be referenced by a sysout DD statement in two ways:

- **Explicitly.** The sysout DD statement contains an OUTPUT parameter that specifies the name of the OUTPUT JCL statement.

# OUTPUT JCL

- **Implicitly.** The sysout DD statement does not contain an OUTPUT parameter. Implicit references are to default OUTPUT JCL statements and require that the job or step contain one or more default OUTPUT JCL statements before the sysout DD statement.

*Note:* If the sysout DD statement does not contain an OUTPUT parameter and the job or step does not contain a default OUTPUT JCL statement, processing of the sysout data set is controlled only by the DD statement, a JES2 /\*OUTPUT statement or JES3 //FORMAT statement, and appropriate installation defaults.

**Job-Level OUTPUT JCL Statements:** This statement appears after the JOB statement and before the first EXEC statement.

**Step-Level OUTPUT JCL Statements:** This statement appears in a step, that is, anywhere after the first EXEC statement in a job.

**Location of Default OUTPUT JCL Statements:** Where you place default OUTPUT JCL statements determines which statements a sysout DD statement refers to in an implicit reference, as follows:

- A sysout DD statement implicitly references all step-level default OUTPUT JCL statements in the same step.
- A sysout DD statement implicitly references all job-level default OUTPUT JCL statements when the step containing the DD statement does not contain any step-level default OUTPUT JCL statements.

You can place more than one job- or step-level default OUTPUT JCL statement in a job or step.

**OUTPUT JCL Statement with JESDS Parameter:** Place an OUTPUT JCL statement with a JESDS parameter after the JOB statement and before the first EXEC statement.

## Overrides

- Parameters on a sysout DD statement override corresponding parameters on an OUTPUT JCL statement.
- Parameters that appear only on the sysout DD statement or only on the OUTPUT JCL statement are used by JES in processing the data set.

## Relationship to the JES2 /\*OUTPUT Statement

JES2 ignores a JES2 /\*OUTPUT statement when either of the following appears in the same job or step:

- A default OUTPUT JCL statement implicitly referenced by a sysout DD statement.
- An OUTPUT JCL statement explicitly referenced by the OUTPUT parameter of a sysout DD statement.

In this case, JES2 uses the form-name subparameter of the DD SYSOUT parameter as a form name, and not as a reference to a JES2 /\*OUTPUT statement.

## Relationship to the JES3 **/\*\*FORMAT** Statement

- When a sysout DD statement implicitly or explicitly references an OUTPUT JCL statement, JES3 ignores any default JES3 **/\*\*FORMAT** statements in the job. A default **/\*\*FORMAT** statement contains a DDNAME=, parameter.
- When a JES3 **/\*\*FORMAT** statement contains a DDNAME parameter that explicitly references a sysout DD statement, JES3 ignores any default OUTPUT JCL statements in the job.
- JES3 uses the processing options from both a JES3 **/\*\*FORMAT** statement and an OUTPUT JCL statement in a job when (1) the **/\*\*FORMAT** statement DDNAME parameter names a sysout DD statement and (2) the sysout DD statement's OUTPUT parameter names an OUTPUT JCL statement. Two separate sets of output are created from the data set defined by the sysout DD statement:
  - One processed according to the options on the JES3 **/\*\*FORMAT** statement.
  - One processed according to the options on the OUTPUT JCL statement.

# OUTPUT JCL: BURST

## BURST Parameter

*Parameter Type:* Keyword, optional

*Purpose:* Use the BURST parameter to specify that 3800 Printing Subsystem output is to go to:

- The burster-trimmer-stacker, to be burst into separate sheets.
- The continuous forms stacker, to be left in continuous fanfold.

If the specified stacker is different from the last stacker used, or if a stacker was not previously requested, JES issues a message to the operator to thread the paper into the required stacker.

*Note:* BURST is valid only for a data set printed on a 3800 equipped with a burster-timmer-stacker.

*Syntax:*

```
BURST={ [YES|Y] }  
      { [NO|N] }
```

## Subparameter Definition

### YES

Requests that the printed output is to be burst into separate sheets. This subparameter can also be coded as Y.

### NO

Requests that the printed output is to be in a continuous fanfold. This subparameter can also be coded as N.

## Defaults

If you do not code a BURST parameter and the output data set is printed on a 3800 that has a burster-timmer-stacker, JES uses an installation default specified at initialization.

## Overrides

A BURST parameter on the sysout DD statement overrides the OUTPUT JCL BURST parameter.

## Example of the BURST Parameter

```
//OUTDS1 OUTPUT BURST=YES
```

In this example, the output from the 3800 will be burst into separate sheets.

## CHARS Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the CHARS parameter to specify the name of one or more character-arrangement tables for printing the data set on a 3800 Printing Subsystem.

**Note:**

- CHARS is valid only for a data set printed on a 3800.
- STD and DUMP are valid only on a JES3 system.

**References:** For more information on character-arrangement tables, see “Requesting Character Arrangements with a 3800 Printer” on page 7-60 and the *IBM 3800 Printing Subsystem Programmer’s Guide*. Refer to *System Generation Reference* for information on how to choose during system generation particular groups, other than the Basic group, which is always available.

**Syntax:**

```

        {table-name                }
CHARS={ (table-name[,table-name]...)}
        {STD                       }
        {DUMP                       }
        { (DUMP[,table-name]...)    }
    
```

- You can omit the parentheses if you code only one table-name.
- Null positions in the CHARS parameter are invalid. For example, you **cannot** code CHARS=(,table-name) or CHARS=(table-name,,table-name).

OUTPUT

## Subparameter Definition

### table-name

Names a character-arrangement table. Each table-name is 1 to 4 alphanumeric or national characters. Code from one to four names.

### STD

Specifies the standard character arrangement table. JES3 uses the standard table specified at initialization.

**Note:** STD is supported only on JES3 systems.

# OUTPUT JCL: CHARS

## DUMP

Requests a high-density dump of 204-character print lines from a 3800. If more than one table-name is coded, DUMP must be first.

*Note:*

- DUMP is supported only on JES3 systems.
- DUMP is valid only on the OUTPUT JCL statement referenced in a SYSABEND, SYSM DUMP, or SYSUDUMP DD statement that specifies a SYSOUT data set for the dump.

## Defaults

If you do not code the OUTPUT JCL CHARS parameter, JES uses the following, in order:

1. The DD CHARS parameter.
2. The DD UCS parameter value, if coded.
3. The OUTPUT JCL UCS parameter value, if coded.

If no character-arrangement table is specified on the DD or OUTPUT JCL statements, JES uses an installation default specified at initialization.

## Overrides

A CHARS parameter on the sysout DD statement overrides the OUTPUT JCL CHARS parameter.

For a data set scheduled to the Print Services Facility (PSF), the PSF uses the following parameters, in override order, to select the font list:

1. Font list in the SYS1.IMAGELIB member specified by an OUTPUT JCL PAGEDEF parameter.
2. DD CHARS parameter.
3. OUTPUT JCL CHARS parameter.
4. DD UCS parameter.
5. OUTPUT JCL UCS parameter.
6. JES installation default for the device.
7. Font list on the PAGEDEF parameter in the PSF cataloged procedure.

See "PAGEDEF Parameter" on page 14-43 for more information.

## Requesting a High-Density Dump in a JES3 System

You can request a high-density dump on the 3800 in a JES3 system through two parameters on the DD statement for the dump data set or on an OUTPUT JCL statement referenced by the dump DD statement:

- **FCB=STD3.** This parameter produces dump output at 8 lines per inch.
- **CHARS=DUMP.** This parameter produces 204-character print lines.

You can code one or both of these parameters. You can place both on the same statement or one on each statement.

## Example of the CHARS Parameter

```
//OUTDS2 OUTPUT CHARS=(GT12,GB12,GI12)
```

In this example, the output from the 3800 will be printed in three upper and lower case fonts: GT12, Gothic 12-pitch; GB12, Gothic Bold 12-pitch; and GI12, Gothic Italic 12-pitch.

# OUTPUT JCL: CKPTLINE

## CKPTLINE Parameter

*Parameter Type:* Keyword, optional

*Purpose:* Use the CKPTLINE parameter to specify the maximum number of lines in a logical page. JES uses this value, with the CKPTPAGE parameter, to determine when to take checkpoints while printing the output data set or transmitting the systems network architecture (SNA) data set.

*Note:* JES3 support for this parameter is limited to the 3800 Printing Subsystem Models 3 and 8.

*Syntax:*

```
CKPTLINE=nnnnn
```

## Subparameter Definition

**nnnnn**

Specifies the maximum number of lines in a logical page. nnnnn is a number from 0 through 32767.

## Defaults

If you do not code the CKPTLINE parameter, JES2 uses an installation default specified at initialization. JES3 provides no installation default.

## Example of the CKPTLINE Parameter

```
//OUTDS3 OUTPUT CKPTLINE=4000,CKPTPAGE=5
```

In this example, the output data set will be checkpointed after every 5 logical pages. Each logical page contains 4000 lines.



## CKPTPAGE Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the CKPTPAGE parameter to specify the number of logical pages:

- To be printed before JES takes a checkpoint.
- To be transmitted as a single systems network architecture (SNA) chain to an SNA work station before JES takes a checkpoint.

The number of lines in these logical pages is specified in the CKPTLINE parameter.

**Note:** JES3 support for this parameter is limited to the 3800 Printing Subsystem Models 3 and 8.

**Syntax:**

```
CKPTPAGE=nnnnn
```

### Subparameter Definition

**nnnnn**

Specifies the number of logical pages to be printed or transmitted before the next output data set checkpoint is taken. nnnnn is a number from 1 through 32767.

OUTPUT

### Defaults

If you do not code the CKPTPAGE parameter, JES2 uses an installation default specified at initialization; the default may also indicate whether checkpoints are to be based on page count or time. JES3 provides no installation default.

### Relationship to Other Parameters

If you code both the CKPTPAGE and CKPTSEC parameters:

- JES2 uses the value on the CKPTSEC parameter, provided the installation did not specify at initialization that checkpoints are to be based only on page count or time.
- JES3 uses the value on the CKPTPAGE parameter.

### Example of the CKPTPAGE Parameter

```
//OUTDS4 OUTPUT CKPTPAGE=128,CKPTLINE=58
```

In this example, the output data set will be checkpointed after every 128 logical pages. Each logical page contains 58 lines.

# OUTPUT JCL: CKPTSEC

## CKPTSEC Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the CKPTSEC parameter to specify how many seconds are to elapse between each checkpoint of an output data set that JES is printing.

**Note:** JES3 support for this parameter is limited to the 3800 Printing Subsystem Models 3 and 8.

**Syntax:**

```
CKPTSEC=nnnnn
```

### Subparameter Definition

**nnnnn**

Specifies the number of seconds that is to elapse between checkpoints. nnnnn is a number from 1 through 32767.

### Defaults

If you do not code the CKPTSEC parameter, JES2 uses an installation default specified at initialization; the default may also indicate whether checkpoints are to be based on page count or time. JES3 provides no installation default.

### Relationship to Other Parameters

If you code both the CKPTPAGE and CKPTSEC parameters:

- JES2 uses the value on the CKPTSEC parameter, provided the installation did not specify at initialization that checkpoints are to be based only on page count or time.
- JES3 uses the value on the CKPTPAGE parameter.

### Example of the CKPTSEC Parameter

```
//OUTDS5 OUTPUT CKPTSEC=120
```

In this example, the output data set will be checkpointed after every 120 seconds, or 2 minutes.

## CLASS Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the CLASS parameter to assign this output data set to an output class.

**Note:** If a system output data set has the same class as the JOB statement MSGCLASS parameter, the job log appears on the same output listing as this output data set.

**References:** For more information on output classes, see “Assigning System Output Data Sets to Output Classes” on page 7-51.

**Syntax:**

```
CLASS={class-name}
      {*           }
```

### Subparameter Definition

**class-name**

Identifies the output class for the data set. The class-name is one character: A through Z or 0 through 9. The attributes of each output class are defined during JES initialization; specify the class with the desired attributes.

\*

Requests the output class in the MSGCLASS parameter on the JOB statement.



### Overrides

The class-name subparameter of the DD statement SYSOUT parameter overrides the OUTPUT JCL CLASS parameter. On the DD statement, you must code a null class-name in order to use the OUTPUT JCL CLASS parameter; for example:

```
//OUTDS DD SYSOUT=( , )
```

### Held Classes

If CLASS specifies a class-name that is defined to JES2 as a held class, the JOB MSGCLASS parameter should specify (1) the same class-name as the CLASS parameter or (2) another class that is also defined to JES2 as a held class. For information on the JOB MSGCLASS parameter, see “MSGCLASS Parameter” on page 10-14.

# OUTPUT JCL: CLASS

## Significance of Output Classes

To print this output data set and the messages from your job on the same output listing, code one of the following:

- The same output class in the DD SYSOUT parameter as in the JOB MSGCLASS parameter.
- DD SYSOUT=\* to default to the JOB MSGCLASS output class.
- DD SYSOUT=(,) to default to one of the following:
  1. The CLASS parameter in an explicitly or implicitly referenced OUTPUT JCL statement. In this case, the OUTPUT JCL CLASS parameter should specify the same output class as the JOB MSGCLASS parameter.
  2. The JOB MSGCLASS output class, if no OUTPUT JCL statement is referenced or if the referenced OUTPUT JCL statement contains CLASS=\*

## Examples of the CLASS Parameter

```
//OUTDS6 OUTPUT CLASS=D
//OUT1 DD SYSOUT=(,),OUTPUT=*.OUTDS6
```

In this example, JES processes the sysout data set defined in DD statement OUT1 in output class D.

```
//PRINTALL JOB ACCT123,MAEBIRD,MSGCLASS=H
//STEP1 EXEC PGM=PRINTER
//OUTDS7 OUTPUT CLASS=*
//OUTPTR DD SYSOUT=(,),OUTPUT=*.OUTDS7
```

In this example, JES processes the sysout data set defined in DD statement OUTPTR in output class H, as specified in the JOB statement MSGCLASS parameter. The same result could be obtained by the following:

```
//PRINTALL JOB ACCT123,MAEBIRD,MSGCLASS=H
//STEP1 EXEC PGM=PRINTER
//OUTPTR DD SYSOUT=H
```

## COMPACT Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the COMPACT parameter to specify a compaction table for JES to use when sending this system output data set, which is a systems network architecture (SNA) data set, to a SNA remote terminal.

**Syntax:**

```
COMPACT=compaction-table-name
```

### Subparameter Definition

**compaction-table-name**

Specifies a compaction table by a symbolic name. The name is 1 to 8 alphanumeric characters. The symbolic name must be defined by the installation during JES initialization.

### Defaults

If you do not code the COMPACT parameter, compaction is suppressed for the data set.

OUTPUT

### Overrides

This parameter overrides any compaction table value defined at the SNA remote terminal.

### Example of the COMPACT Parameter

```
//OUTDS8 OUTPUT DEST=N555R222,COMPACT=TBL77
```

In this example, the data set will be sent to remote terminal 222 at node 555; JES will use compaction table TBL77.

# OUTPUT JCL: CONTROL

## CONTROL Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the CONTROL parameter to specify either that each logical record starts with a carriage control character or that the output is to be printed with single, double, or triple spacing.

**Syntax:**

```
CONTROL={PROGRAM}  
CONTROL={SINGLE }  
CONTROL={DOUBLE }  
CONTROL={TRIPLE }
```

### Subparameter Definition

#### PROGRAM

Indicates that each logical record in the data set begins with a carriage control character. The carriage control characters are given in *Data Management Services*.

#### SINGLE

Indicates forced single spacing.

#### DOUBLE

Indicates forced double spacing.

#### TRIPLE

Indicates forced triple spacing.

### Defaults

If you do not code the CONTROL parameter, JES3 uses an installation default specified at initialization.

In a JES2 system, an installation default can be provided for each local device by an operator command.

### Example of the CONTROL Parameter

```
//OUTDS9 OUTPUT CONTROL=PROGRAM
```

In this example, the output is printed using the first character of each logical record for carriage control.

## COPIES Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the COPIES parameter to specify how many copies of the system output data set are to be printed. The printed output is in page sequence for each copy.

For printing on a 3800 Printing Subsystem, this parameter can instead specify how many copies of each page are to be printed before the next page is printed.

**References:** For more information on the COPIES parameter, see “Requesting Multiple Copies of an Output Data Set Using JES2” and “Requesting Multiple Copies of an Output Data Set Using JES3” on page 7-58.

**Syntax:**

<pre>COPIES={nnn         { (, (group-value [, group-value] ... ) ) }</pre>
<ul style="list-style-type: none"> <li>● You can omit the parentheses if you code only COPIES = nnn.</li> </ul> <p>The following are <b>not</b> valid:</p> <ul style="list-style-type: none"> <li>– A null group-value, for example, COPIES = (5,(,)) or COPIES = (5,)</li> <li>– A zero group-value, for example, COPIES = (5,(1,0,4))</li> <li>– A null within a list of group-values, for example, COPIES = (5,(1,,4))</li> </ul>

OUTPUT

### Subparameter Definition

**nnn**

Specifies how many copies of the data set are to be printed; each copy will be in page sequence order. nnn is a number from 1 through 255 in a JES2 system and from 1 through 254 in a JES3 system.

For a data set printed on a 3800, JES ignores nnn if any group-values are specified.

**group-value**

Specifies how many copies of each page are to be printed before the next page is printed. Each group-value is a number from 1 through 255 in a JES2 system and from 1 through 254 in a JES3 system. You can code a maximum of eight group-values. Their sum must not exceed 255 or 254. The total copies of each page equals the sum of the group-values.

*Note:*

- This subparameter is valid only for 3800 output.
- For 3800 output, this subparameter overrides an nnn subparameter, if coded.

# OUTPUT JCL: COPIES

## Defaults

If you do not code a COPIES parameter on any of the following, code it incorrectly, or code COPIES=0, the system uses a default of 1, which is the default for the DD COPIES parameter.

- DD statement
- OUTPUT JCL statement
- For JES2, the /\*OUTPUT statement
- For JES3, the SYSOUT initialization statement

## Overrides

A COPIES parameter on the sysout DD statement overrides the OUTPUT JCL COPIES parameter.

If the OUTPUT JCL statement contains a FORMDEF parameter, which specifies a SYS1.IMAGELIB member, the COPYGROUP parameter on a FORMDEF statement in that member overrides any group-value subparameters on the OUTPUT JCL COPIES parameter or the sysout DD COPIES parameter. For more information, see "FORMDEF Parameter" on page 14-31.

## Relationship to Other Parameters

If the OUTPUT JCL or the sysout DD statement contains a FLASH parameter, JES prints with the forms overlay the number of copies specified in one of the following:

- COPIES=nnn, if the FLASH count is larger than nnn. For example, if COPIES=10 and FLASH=(,12) JES prints 10 copies, all with the forms overlay.
- The sum of the group-values specified in the COPIES parameter, if the FLASH count is larger than the sum. For example, if COPIES=(,2,3,4) and FLASH=(,12) JES prints nine copies in groups, all with the forms overlay.
- The count subparameter in the FLASH parameter, if the FLASH count is smaller than nnn or the sum from the COPIES parameter. For example, if COPIES=10 and FLASH=(,7) JES prints seven copies with the forms overlay and three copies without.

## Relationship to Other Control Statements

For JES2, if you request copies of the entire job on the JES2 /\*JOBPARM COPIES parameter and also copies of the data set on the DD COPIES or OUTPUT JCL COPIES parameter, JES2 prints the number of copies equal to the **product** of the two requests.

## Examples of the COPIES Parameter

```
//RPTDS OUTPUT COPIES=4,FORMS=WKREPORT
```

This example asks JES to print four copies of the weekly report on forms named WKREPORT.



## OUTPUT JCL: COPIES

```
//EXPLD OUTPUT COPIES=(,(3)),FORMS=ACCT
```

This example asks JES to print the first page three times, then the second page three times, the third page three times, etc., on forms named ACCT.

---

```
//QUEST OUTPUT COPIES=(,(8,25,18,80)),FORMS=ANS
```

This example asks JES to print each page eight times before printing the next page, then 25 times before the next, then 18 times before the next, and finally 80 times before the next. The forms are named ANS.

---

```
//EXMP OUTPUT COPIES=(5,(3,2))
```

This example asks JES to do one of the following:

- If the data set is printed on other than a 3800, to print five copies.
- If it is printed on a 3800, to print each page three times before printing the next page and then to print each page twice before printing the next page.

OUTPUT

# OUTPUT JCL: DEFAULT

## DEFAULT Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the DEFAULT parameter to specify that this OUTPUT JCL statement can or cannot be implicitly referenced by a sysout DD statement. An OUTPUT JCL statement that contains a DEFAULT= YES parameter is called a default OUTPUT JCL statement.

**References:** For more information on the DEFAULT parameter, see "Processing System Output Data Sets Using the OUTPUT JCL Statement" on page 7-44.

**Syntax:**

```
DEFAULT={ [ YES | Y ] }  
        { [ NO | N ] }
```

### Subparameter Definition

#### YES

Indicates that this OUTPUT JCL statement can be implicitly referenced by sysout DD statements. This subparameter can also be coded as Y.

#### NO

Indicates that this OUTPUT JCL statement cannot be implicitly referenced by sysout DD statements. This subparameter can also be coded as N.

### Defaults

If you do not code DEFAULT= YES, the default is NO. In order to take effect, an OUTPUT JCL statement without DEFAULT= YES must be explicitly referenced in an OUTPUT parameter on a sysout DD statement.

### Location in the JCL

- A step-level OUTPUT JCL statement appears within a step, that is, anywhere after the first EXEC statement in a job.
- A job-level OUTPUT JCL statement appears after the JOB statement and before the first EXEC statement.
- You can place more than one job- or step-level default OUTPUT JCL statement in a job or step.
- You must place an OUTPUT JCL statement in the input stream **before** any sysout DD statement that explicitly or implicitly refers to it.

# OUTPUT JCL: DEFAULT

## References to Default OUTPUT JCL Statements

- A sysout DD statement makes an explicit reference in an OUTPUT parameter that specifies the name of an OUTPUT JCL statement.
- A sysout DD statement makes an implicit reference when it does not contain an OUTPUT parameter, and the job or step contains one or more default OUTPUT JCL statements.
- A sysout DD statement implicitly references all step-level default OUTPUT JCL statements in the same step.
- A sysout DD statement implicitly references all job-level default OUTPUT JCL statements when the step containing the DD statement does not contain any step-level default OUTPUT JCL statements.
- A sysout DD statement can explicitly reference a default OUTPUT JCL statement.

## Example of the DEFAULT Parameter

```
//EXMP2 JOB ACCT555,MAEBIRD,MSGCLASS=B
//OUTDAL OUTPUT DEFAULT=YES,DEST=DALLAS
//OUTPOK OUTPUT DEST=POK
//STEP1 EXEC PGM=REPORT
//OUTHERE OUTPUT CLASS=D
//SYSIN DD *
.
.
/*
//WKRPT DD UNIT=VIO,DISP=(,PASS)
//RPT1 DD SYSOUT=(,),OUTPUT=* .OUTHERE
//RPT2 DD SYSOUT=A
//STEP2 EXEC PGM=SUMMARY
//OUTHQ OUTPUT DEFAULT=YES,DEST=HQ
//WKDATA DD UNIT=VIO,DISP=(OLD,DELETE),DSNAME=* .STEP1.WKRPT
//MONTH DD SYSOUT=(,),OUTPUT=* .STEP1.OUTHERE
//SUM DD SYSOUT=A
//FULRPT DD SYSOUT=A,OUTPUT=( *.OUTDAL, *.OUTPOK)
```

OUTPUT

In this example, the JOB named EXMP2 contains two job-level OUTPUT JCL statements: OUTDAL and OUTPOK. OUTDAL is a default OUTPUT JCL statement because it contains DEFAULT=YES; OUTDAL can be implicitly referenced by a sysout DD statement. OUTPOK must be explicitly referenced in a sysout DD OUTPUT parameter for its processing options to be used. The purpose of both of these OUTPUT JCL statements is to specify a destination for an output data set.

STEP1 contains a step-level OUTPUT JCL statement: OUTHERE. The purpose of this statement is to specify that JES process the data set locally in output class D. OUTHERE can be used only if it is explicitly referenced.

STEP2 contains a step-level default OUTPUT JCL statement: OUTHQ. The purpose of this statement is to specify a destination for an output data set. OUTHQ can be implicitly referenced.

## OUTPUT JCL: DEFAULT

The references in this job are as follows:

- In STEP1 and STEP2, sysout DD statements RPT1 and MONTH explicitly reference OUTPUT JCL statement OUTHQ. These two output data sets are printed locally in the same output class.

*Note:* You can explicitly reference an OUTPUT JCL statement in a preceding job step.

- In STEP1, DD statement RPT2 implicitly references OUTPUT JCL statement OUTDAL. This implicit reference occurs because all of the following are true:
  1. DD statement RPT2 contains a SYSOUT parameter but does not contain an OUTPUT parameter. Thus, this DD statement is making an implicit reference.
  2. STEP1 does not contain a default OUTPUT JCL statement, so the implicit reference must be to job-level default OUTPUT JCL statements.
  3. OUTDAL is the only job-level default OUTPUT JCL statement.
- In STEP2, DD statement SUM implicitly references OUTPUT JCL OUTHQ because all of the following are true:
  1. DD statement SUM contains a SYSOUT parameter but does not contain an OUTPUT parameter. Thus, this DD statement is making an implicit reference.
  2. STEP2 contains a default OUTPUT JCL statement: OUTHQ. Therefore, the implicit reference is to OUTHQ and cannot be to any job-level default OUTPUT JCL statements.
- In STEP2, DD statement FULRPT explicitly references OUTPUT JCL statements OUTDAL and OUTPOK.

## DEST Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the DEST parameter to specify a destination for the system output data set. The DEST parameter can send a system data set to a remote or local terminal, a node, a node and remote work station, a local device or group of devices, or a node and userid.

**References:** For more information on the DEST parameter, see “Controlling Output Destination in a JES2 Network” on page 3-7 and “Controlling Output Destination Using JES3” on page 3-12.

**Syntax:**

```
DEST=destination
```

The destination subparameter for JES2 is one of the following:

```
LOCAL
name
Nnnnn
NnnRmmmm
NnnnRmmm
NnnnnRmm
nodename.userid
Rnnnn
RMnnnn
RTnnnn
Unnn
```

The destination subparameter for JES3 is one of the following:

```
device-name
group-name
LOCAL
nodename
nodename.remote
(type)
```

OUTPUT

### Subparameter Definition for JES2 Systems

**LOCAL**

Indicates any local device.

**name**

Specifies a local or remote device by a symbolic name defined by the installation during JES2 initialization. The name is 1 through 8 alphanumeric or national characters.

**Nnnnn**

Specifies a node. nnnn is 1 through 4 decimal numbers from 1 through 1000.

## OUTPUT JCL: DEST

### **NnnRmmmm**

### **NnnnRmmm**

### **NnnnnRmm**

Specifies a node and a remote work station connected to the node. The node number, indicated in the format by n, is 1 through 4 decimal numbers from 1 through 1000. The remote work station number, indicated in the format by m, is 1 through 4 decimal numbers from 1 through 4000. Do not code leading zeros in n or m. The maximum number of digits for n and m combined cannot exceed six.

*Note:* R0 is equivalent to LOCAL specified at node Nn.

### **nodename.userid**

Identifies the nodename and userid of the destination node. Use this parameter to route information between JES2 nodes and non-JES2 nodes. The nodename is 1 through 8 alphanumeric characters. The userid is 1 through 8 alphanumeric characters, except R, RM, RMT, comma, right parenthesis, or blank. Enclose the userid in apostrophes when it contains special characters or begins with a number.

*Note:* If a data set is queued for transmission and an operator changes its destination, the userid portion of the original routing is lost.

### **Rnnnn**

### **RMnnnn**

### **RMTnnnn**

Specifies a remote terminal. nnnn is 1 through 4 decimal numbers from 1 through 4000.

*Note:* R0 is equivalent to LOCAL.

### **Unnn**

Specifies a local terminal with special routing. nnn is 1 through 3 decimal numbers from 1 through 255.

## Subparameter Definition for JES3 Systems

### **device-name**

Specifies a local device by a symbolic name defined by the installation during JES3 initialization. device-name is 1 through 8 alphanumeric or national characters.

### **group-name**

Specifies a group of local devices, an individual remote station, or a group of remote stations by a symbolic name defined by the installation during JES3 initialization. group-name is 1 through 8 alphanumeric or national characters.

### **LOCAL**

Indicates any local device that is attached to the global processor and that does not belong to a group.

*Note:* When you code DEST=LOCAL, your installation must have at least one local device that is not assigned to a device group.

## **nodename**

Specifies a node by a symbolic name defined by the installation during JES3 initialization. **nodename** is 1 through 8 alphanumeric or national characters. If the **nodename** you specify is the same as the node you are working on, JES3 treats the output as though you specified **LOCAL**.

## **nodename.remote**

Specifies a node and either a remote work station or VM userid at that node, as follows:

### **nodename**

A symbolic name defined by the installation during JES3 initialization. The **nodename** is 1 to 8 alphanumeric or national characters.

### **remote**

A name for a remote work station. The name is 1 through 8 alphanumeric or national characters and must be defined at the node. Enclose it in apostrophes when it contains special characters or begins with a number.

## **(type)**

Specifies a device classification. **type** is in the form (**gggssss**), where **ggg** is the general device classification and **ssss** is the specific device classification; **type** must be enclosed in parentheses. The **type** must be defined by the installation during JES3 initialization. For example, **type** for a 3800 is (PRT3800).

## Defaults

If you do not code a **DEST** parameter, JES directs the sysout data set to the default destination for the input device from which the job was submitted.

If a specified destination is invalid, the job fails.

## Overrides

A **DEST** parameter on the sysout DD statement overrides the **OUTPUT JCL DEST** parameter.

## Examples of the DEST Parameter

```
//REMOT1 OUTPUT DEST=R444
```

In this example, JES2 sends the output data set to remote terminal 444.

---

```
//REMOT2 OUTPUT DEST=STAT444
```

In this example, JES sends the output data set to an individual remote station named by the installation **STAT444**.

---

```
//REMOT3 OUTPUT DEST=KOKVMBB8.DP58HHHD
```

In this example, JES sends the output data set to VM userid **DP58HHHD** at node **KOKVMBB8**.

## FCB Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the FCB parameter to specify:

- The forms control buffer (FCB) image JES is to use to guide printing of the system output data set by a 3211 Printer, 3203 Printer Model 5, 3800 Printing Subsystem, or 4248 Printer, or by a printer supported by systems network architecture (SNA) remote job entry (RJE).
- The carriage control tape JES is to use to control printing of the output data set by a 1403 Printer or by a printer supported by SNA RJE.
- The data-protection image JES is to use to control output by a 3525 Card Punch.

The FCB image specifies how many lines are to be printed per inch and the length of the form. JES loads the image into the printer's forms control buffer. The FCB image is stored in SYS1.IMAGELIB. IBM provides three standard FCB images:

- STD1, which specifies 6 lines per inch on an 8.5-inch-long form.
- STD2, which specifies 6 lines per inch on an 11-inch-long form.
- STD3, which in a JES3 system specifies 8 lines per inch for a dump.

**References:** For more information on the FCB parameter, see "Requesting Forms Control" on page 7-60. For more information on the forms control buffer, see *SPL: Data Management, Programming Support for the IBM 3505 Card Reader and IBM 3525 Card Punch*, or *IBM 3800 Printing Subsystem Programmer's Guide*.

### Syntax:

```
FCB={ fcb-name }  
    { STD      }
```

- Code the fcb-name as STD1 or STD2 only to request the IBM-supplied images.
- Code the fcb-name as STD3 only for a high-density dump in a JES3 system.

## Subparameter Definition

### fcb-name

Identifies the FCB image. The name is 1 to 4 alphanumeric or national characters and is the last characters of a SYS1.IMAGELIB member name:

- FCB2xxxx member, for a 3211, a 3203 model 5, or a printer supported by SNA.
- FCB3xxxx member, for a 3800.
- FCB4xxxx member, for a 4248.



## STD

Indicates the standard FCB. JES3 uses the standard FCB specified at JES3 initialization.

*Note:* STD is supported only on JES3 systems.

## Defaults

If you do not code the FCB parameter, the system checks the FCB image in the printer's forms control buffer; if it is a default image, as indicated by its first byte, JES uses it. If it is not a default image, JES loads the FCB image that is the installation default specified at JES initialization.

## Overrides

An FCB parameter on the sysout DD statement overrides the OUTPUT JCL FCB parameter.

## Relationship to Other Parameters

The FCB parameter is mutually exclusive with the FRID subparameter of the DD statement DCB parameter.

## Requesting a High-Density Dump in a JES3 System

You can request a high-density dump on the 3800 in a JES3 system through two parameters on the DD statement for the dump data set or on an OUTPUT JCL statement referenced by the dump DD statement:

- FCB=STD3. This parameter produces dump output at 8 lines per inch.
- CHARS=DUMP. This parameter produces 204-character print lines.

You can code one or both of these parameters. You can place both on the same statement or one on each statement.

## Example of the FCB Parameter

```
//OUTDS1 OUTPUT FCB=AA33
```

In this example, JES will print the output data set using the FCB image named AA33.

# OUTPUT JCL: FLASH

## FLASH Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the FLASH parameter to identify the forms overlay to be used in printing the output data set on a 3800 Printing Subsystem and, optionally, to specify the number of copies on which the forms overlay is to be printed.

**Note:** FLASH is valid only for a data set printed on a 3800.

**References:** For more information on the FLASH parameter, see "Requesting Forms Overlay" on page 7-62. For information on forms overlays, see the *Forms Design Reference Guide for the IBM 3800 Printing Subsystem*.

**Syntax:**

```
FLASH={ (overlay-name [ , count ] ) }
      { ( , count ) }
      { NONE }
      { STD }
```

The count subparameter is optional. If you omit it, you can omit the parentheses. However, if you omit it, you must not code it as a null; for example, FLASH=(ABCD,) is invalid.

## Subparameter Definition

**overlay-name**

Identifies the forms overlay frame that the operator is to insert into the printer before printing begins. The name is 1 to 4 alphanumeric or national characters.

**count**

Specifies the number, 0 to 255, of copies that JES is to flash with the overlay, beginning with the first copy printed. Code a count of 0 to flash **no** copies.

**NONE**

Suppresses flashing for this data set.

**STD**

Indicates the standard forms flash overlay. JES3 uses the standard forms overlay specified at JES3 initialization.

**Note:** STD is supported only on JES3 systems.

## Defaults

If you do not code a **FLASH** parameter or specify an installation default at JES2 or JES3 initialization, forms are not flashed.

If you specify an overlay-name without specifying a count, all copies are flashed. That is, the default for count is 255.

## Overrides

A **FLASH** parameter on the sysout DD statement overrides the **OUTPUT JCL FLASH** parameter.

## Relationship to Other Parameters

If the **OUTPUT JCL** or the sysout DD statement also contains a **COPIES** parameter, JES prints with the forms overlay the number of copies specified in one of the following:

- **COPIES**=nnn, if the **FLASH** count is larger than nnn. For example, if **COPIES**=10 and **FLASH**=(LTHD,12) JES prints 10 copies, all with the forms overlay.
- The sum of the group-values specified in the **COPIES** parameter, if the **FLASH** count is larger than the sum. For example, if **COPIES**=(,2,3,4) and **FLASH**=(LTHD,12) JES prints nine copies in groups, all with the forms overlay.
- The count subparameter in the **FLASH** parameter, if the **FLASH** count is smaller than nnn or the sum from the **COPIES** parameter. For example, if **COPIES**=10 and **FLASH**=(LTHD,7) JES prints seven copies with the forms overlay and three copies without.

OUTPUT

## Verification of Forms Overlay Frame

Before printing starts, JES does not verify that the operator inserted the correct forms overlay frame for flashing.

## Printing without Flashing

To print without flashing, specify one of the following:

- **FLASH**=NONE on the DD or **OUTPUT JCL** statement.
- Omit the **FLASH** parameter on all of the statements for the data set and on all JES initialization statements.
- **FLASH**=(,0) on the **OUTPUT JCL** statement.

## OUTPUT JCL: FLASH

### Example of the FLASH Parameter

```
//OUTDS1 OUTPUT COPIES=16,FLASH=(LTHD,7)
```

In this example, JES issues a message to the operator requesting that the forms overlay frame named LTHD be inserted in the printer. Then JES prints the first seven copies of the output data set with the forms overlay and the last nine without.

## FORMDEF Parameter

*Parameter Type:* Keyword, optional

*Purpose:* Use the FORMDEF parameter to identify a library member that contains statements to tell the Print Services Facility (PSF) how to print the system output data set on a 3800 Printing Subsystem Model 3. The statements can specify the following:

- Overlay forms to be used during printing.
- Location on the page where overlays are to be placed.
- Suppressions that can be activated for specified page formats.

The data set must be in a member of the library named in the cataloged procedure that was used to initialize the PSF; this library is SYS1.IMAGELIB.

*Note:* FORMDEF can be specified only for data sets printed on a 3800 model 3.

*References:* For information on the FORMDEF statement in the SYS1.IMAGELIB member, see *Print Management Facility User's Guide and Reference*, and on SYS1.IMAGELIB, see *IBM 3800 Printing Subsystem Models 3 and 8 Programmer's Guide*.

*Syntax:*

```
FORMDEF=membername
```



### Subparameter Definition

**membername**

Specifies the name of a SYS1.IMAGELIB member. The name is 1 to 6 alphanumeric or national characters. The first two characters of the membername are pre-defined by the system.

### Overrides

The SYS1.IMAGELIB member specified by the OUTPUT JCL FORMDEF parameter can contain:

- Statements that override the installation's FORMDEF defaults in the PSF cataloged procedure.
- A FORMDEF statement with a COPYGROUP parameter. The COPYGROUP parameter overrides any group-value subparameters on the OUTPUT JCL COPIES parameter or the sysout DD COPIES parameter.

*Note:* The FORMDEF statement in SYS1.IMAGELIB does not override a sysout DD or OUTPUT JCL COPIES=nnn parameter.

# OUTPUT JCL: FORMDEF

## Example of the FORMDEF Parameter

```
//PRINT3 OUTPUT FORMDEF=JJPRT
```

In this example, PSF is to print the output data set on a 3800 model 3 according to the parameters in the SYS1.IMAGELIB member JJPRT.

## FORMS Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the FORMS parameter to identify the forms on which the output data set is to be printed or punched.

**Syntax:**

```
FORMS={form-name}
      {STD      }
```

### Subparameter Definition

#### form-name

Identifies the print or punch forms. form-name is 1 to 8 alphanumeric or national characters.

#### STD

Indicates the standard form. JES3 uses the standard form specified at JES3 initialization.

*Note:* STD is supported only on JES3 systems.

### Defaults

If you do not code a form-name subparameter, JES uses an installation default specified at initialization.

### Overrides

The form-name subparameter of the SYSOUT parameter on the sysout DD statement overrides the OUTPUT JCL FORMS parameter. Note that the SYSOUT form-name subparameter can be only four characters maximum while both the OUTPUT JCL FORMS form-name and the JES initialization default form names can be eight characters maximum.

### Example of the FORMS Parameter

```
//OUTDS1 OUTPUT FORMS=ACCT4010
```

In this example, the output data set will be printed on forms named ACCT4010.

# OUTPUT JCL: GROUPID

## GROUPID Parameter

**Parameter Type:** Keyword, optional, JES2 only

**Purpose:** Use the GROUPID parameter to specify that this output data set belongs to an output group. The data sets in an output group are processed together in the same location and time. Data sets to be grouped should have similar characteristics: the same output class, destination, process mode, and external writer name.

**Note:** GROUPID is supported only on JES2 systems.

**References:** For more information on grouping data sets, see "Grouping Data Sets Using the OUTPUT JCL Statement" on page 7-49.

**Syntax:**

```
GROUPID=output-group
```

## Subparameter Definition

### output-group

Specifies the name of an output group. The output-group is 1 to 8 alphanumeric characters and is selected by the programmer to define an output group for this job. The name is not installation-defined.

## Examples of the GROUPID Parameter

```
//EXMP5 JOB ACCT1984,MAEBIRD,MSGCLASS=A
//OUTRPT OUTPUT GROUPID=RPTGP,DEFAULT=YES,DEST=TDC
//STEP1 EXEC PGM=RPTWRIT
//SYSIN DD *
.
.
/*
//RPTDLY DD SYSOUT=C
//RPTWK DD SYSOUT=C
```

In this example, the DD statements RPTDLY and RPTWK implicitly reference the default OUTPUT JCL statement OUTRPT. JES2 creates two output groups:

1. Group RPTGP is created because of the GROUPID parameter in the OUTPUT JCL statement. It contains the two reports from the sysout DD statements RPTDLY and RPTWK and is printed at the destination TDC. The programmer named this group RPTGP.
2. The other group is named by JES2. It contains the system-managed data set for the job's messages.



## OUTPUT JCL: GROUPID

```
//EXAMP   JOB      MSGCLASS=A
//JOBOUT  OUTPUT   GROUPID=SUMM,DEST=HQS,CHARS=GT10
//STEP1   EXEC     PGM=RWRITE
//STEP1OUT OUTPUT  FORMS=STD,CHARS=GS10,DEST=LOCAL
//RPTDD1  DD       SYSOUT=A,OUTPUT=(*.STEP1OUT,*.JOBOUT)
//STEP2   EXEC     PGM=SWRITE
//STEP2OUT OUTPUT  FORMS=111,CHARS=GB10,DEST=LOCAL
//RPTDD2  DD       SYSOUT=B,OUTPUT=(*.STEP2OUT,*.JOBOUT)
```

This job causes JES2 to produce five sets of output:

1.1.1, containing the system-managed data sets. This set is specified through the JOB statement MSGCLASS parameter.

SUMM.1.1, containing a copy of the data set defined by DD statement RPTDD1. This set is specified through the second OUTPUT subparameter: \*.JOBOUT. It is for output class A.

SUMM.2.1, containing a copy of the data set defined by DD statement RPTDD2. This set is specified through the second OUTPUT subparameter: \*.JOBOUT. Because it is for output class B, it is in a separate subgroup from the SUMM.1.1 subgroup.

4.1.1, containing a copy of the data set defined by DD statement RPTDD1. This set is specified through the first OUTPUT subparameter: \*.STEP1OUT.

5.1.1, containing a copy of the data set defined by DD statement RPTDD2. This set is specified through the first OUTPUT subparameter: \*.STEP2OUT.

OUTPUT

# OUTPUT JCL: INDEX

## INDEX Parameter

**Parameter Type:** Keyword, optional, JES2 only

**Purpose:** Use the INDEX parameter to set the left margin for output on a 3211 Printer with the indexing feature. The width of the print line is reduced by the INDEX parameter value.

**Note:** INDEX is supported only on JES2 systems and only for output printed on a 3211 with the indexing feature. JES2 ignores the INDEX parameter if the printer is not a 3211 with the indexing feature.

**Syntax:**

```
INDEX=nn
```

## Subparameter Definition

**nn**

Specifies how many print positions the left margin on 3211 output is to be indented. nn is a decimal number from 1 through 31. n=1 indicates flush-left; n=2 through n=31 indent the print line by n-1 positions.

## Defaults

The default is 1, which indicates flush left. Thus, if you do not code an INDEX or LINDEX parameter, JES2 prints full-width lines.

## Relationship to Other Parameters

If you code both INDEX and LINDEX, JES2 ignores the first parameter and processes the last parameter it encounters. Note that you cannot index both the left and right margins.

## Example of the INDEX Parameter

```
//OUT17 OUTPUT INDEX=6
```

In this example, because the printed report is to be stapled, extra space is needed on the left. Assuming the data set is printed on a 3211 with the indexing feature, all lines are indented 5 print positions from the left page margin.

## JESDS Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the JESDS parameter to process the job's system data sets according to the parameters on this OUTPUT JCL statement. The system data sets consist of:

- The job log, which is a record of job-related information for the programmer. Printing of the job log is controlled by two JOB statement parameters: the MSGLEVEL parameter controls what is printed and the MSGCLASS parameter controls the system output class.
- The job's hard-copy log, which is a record of all message traffic for the job to and from the operator console.
- System messages for the job.

The class for the system data sets is the class that applies to the OUTPUT JCL statement. See "CLASS Parameter" on page 14-13.

**References:** For more information on the job log, see "Job Log" on page 3-14 and *System Commands*.

**Syntax:**

```
JESDS={ALL}
        {JCL}
        {LOG}
        {MSG}
```

OUTPUT

### Subparameter Definition

**ALL**

Indicates all of the job's system-managed data sets.

**JCL**

Indicates all JCL statements, cataloged or in-stream procedure statements, and JCL messages.

**LOG**

Indicates the job's hard-copy log.

**MSG**

Indicates any system messages for this job.

# OUTPUT JCL: JESDS

## Overrides

The NOLOG parameter on a JES2 /\*JOBPARM statement overrides the OUTPUT JCL JESDS=ALL parameter.

## Location in the JCL

Place an OUTPUT JCL statement containing JESDS before the first EXEC statement of the job. An OUTPUT JCL statement containing JESDS placed after an EXEC statement is a JCL error.

You can place more than one OUTPUT JCL statement containing JESDS before the first EXEC statement. JES creates a copy of the job's system data sets for each.

## Destination for the System Data Sets

If you want the job's system data sets processed at a particular destination, code a DEST parameter on the OUTPUT JCL statement containing JESDS. Otherwise, JES routes the system data sets to a local device.

## Example of the JESDS Parameter

```
//EXMP   JOB      MSGCLASS=A
//OUT1   OUTPUT   JESDS=ALL
//OUT2   OUTPUT   JESDS=ALL,DEST=AUSTIN
      .
      .
      .
```

In this example, JES produces two copies of the system data sets: one copy for OUTPUT JCL statement OUT1 and one copy for OUTPUT JCL statement OUT2. The copy for statement OUT2 is sent to AUSTIN.

## INDEX Parameter

**Parameter Type:** Keyword, optional, JES2 only

**Purpose:** Use the LINDEX parameter to set the right margin for output on a 3211 Printer with the indexing feature. The width of the print line is reduced by the LINDEX parameter value.

**Note:** LINDEX is supported only on JES2 systems and only for output printed on a 3211 with the indexing feature. JES2 ignores the LINDEX parameter if the printer is not a 3211 with the indexing feature.

**Syntax:**

```
LINDEX=nn
```

### Parameter Definition

**nn**

Specifies how many print positions the right margin on 3211 output is to be moved in from the full page width. nn is a decimal number from 1 through 31. n=1 indicates flush-right; n=2 through n=31 move the right margin over by n-1 positions.

### Defaults

The default is 1, which indicates flush right. Thus, if you do not code an INDEX or LINDEX parameter, JES2 prints full-width lines.

### Relationship to Other Parameters

If you code both INDEX and LINDEX, JES2 ignores the first parameter and processes the last parameter it encounters. Note that you cannot index both the left and right margins.

### Example of the LINDEX Parameter

```
//OUT18 OUTPUT LINDEX=21
```

In this example, the author of the report wanted extra space on the right side of the paper for notes. Assuming the data set is printed on a 3211 with the indexing feature, all lines are ended 20 print positions from the right page margin.

OUTPUT

# OUTPUT JCL: LINECT

## LINECT Parameter

**Parameter Type:** Keyword, optional, JES2 only

**Purpose:** Use the LINECT parameter to specify the maximum number of lines JES2 is to print on each output page.

**Note:** LINECT is supported only on JES2 systems.

**Syntax:**

```
LINECT=nnn
```

## Subparameter Definition

**nnn**

Specifies the maximum number of lines JES2 is to print on each page. nnn is a number from 0 through 255.

Specify LINECT=0 to keep JES2 from starting a new page when the number of lines exceeds the JES2 initialization parameter.

## Defaults

If you do not code the LINECT parameter, JES2 uses an installation default specified at initialization.

## Example of the LINECT Parameter

```
//PRNTDS OUTPUT LINECT=45
```

In this example, JES2 will start a new page after every 45 lines.

## MODIFY Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the MODIFY parameter to specify a copy-modification module that tells JES how to print the system output data set on a 3800 Printing Subsystem. The module can specify the following:

- Legends.
- Column headings.
- Where and on which copies the data is to be printed.

The module is defined and stored in SYS1.IMAGELIB using the IEBIMAGE utility program.

**Note:** MODIFY is supported only for the 3800 Printing Subsystem Model 1 and 2 and the 3800 Printing Subsystem Model 3 in compatibility mode. For the 3800 model 3, use the FORMDEF and PAGEDEF parameters to obtain the same functions.

**References:** For more information on the MODIFY parameter, see “Requesting Copy Modification” on page 7-58, and on the copy modification module and the IEBIMAGE utility program, see the *IBM 3800 Printing Subsystem Programmer's Guide*.

### Syntax:

```
MODIFY={module-name      }
      {([module-name][,trc])}
```

- You can omit the module-name, thereby obtaining the initialization default. For example, MODIFY=(,2).
- The trc subparameter is optional. If you omit it, you can omit the parentheses. However, if you omit it, you must not code it as a null; for example, MODIFY=(TAB1,) is invalid.

OUTPUT

### Subparameter Definition

#### module-name

Identifies a copy-modification module in SYS1.IMAGELIB. The module-name is 1 through 4 alphanumeric or national characters.

## OUTPUT JCL: MODIFY

### **trc**

Identifies which table-name in the CHARS parameter is to be used. This **table reference character** is 0 for the first table-name specified, 1 for the second, 2 for the third, or 3 for the fourth. The CHARS parameter used is on the following, in override order:

1. The DD statement.
2. This OUTPUT JCL statement.
3. A statement in the SYS1.IMAGELIB member specified on the OUTPUT JCL PAGEDEF parameter.
4. A statement in the SYS1.IMAGELIB member obtained by default.
5. A JES3 initialization statement.

If the **trc** value is greater than the number of table-names in the CHARS parameter, JES2 uses the first table named in the CHARS parameter and JES3 uses the default character arrangement table.

### **Defaults**

If you do not code module-name in the MODIFY parameter, JES3 uses an installation default specified at initialization. JES2 provides no installation default at initialization.

If you do not specify **trc**, the default is 0.

### **Overrides**

A MODIFY parameter on the sysout DD statement overrides the OUTPUT JCL MODIFY parameter.

### **Relationship to Other Parameters**

The second character of each logical record can be a TRC code, so that each record can be printed in a different font. This way of specifying fonts is indicated by the OUTPUT JCL TRC parameter.

### **Example of the MODIFY Parameter**

```
//OUTDS1 OUTPUT CHARS=(GT12,GB12,GI12),MODIFY=(MODA,2)
```

In this example, JES loads the MODA module in SYS1.IMAGELIB into the 3800 and uses the GI12, Gothic Italic 12-pitch font, which is the third one specified in the CHARS parameter.



## PAGEDEF Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the PAGEDEF parameter to identify a library member that contains statements to tell the Print Services Facility (PSF) how to print the output data set on a 3800 Printing Subsystem Model 3. The statements can specify the following:

- Logical page length and width.
- Fonts.
- Page segments.
- Multiple page types or formats.
- Lines within a page; for example:
  - Line origin.
  - Carriage controls.
  - Spacing.
- Multiple logical pages on a physical page.

The member must be in the library named in the cataloged procedure that was used to initialize the PSF; this library is SYS1.IMAGELIB.

**Note:** PAGEDEF can be specified only for data sets printed on a 3800 model 3.

**References:** For information on the PAGEDEF statement in the SYS1.IMAGELIB member, see *Print Management Facility User's Guide and Reference*, and on SYS1.IMAGELIB, see *IBM 3800 Printing Subsystem Models 3 and 8 Programmer's Guide*.

**Syntax:**

```
PAGEDEF=membername
```

### Subparameter Definition

**membername**

Specifies the name of the SYS1.IMAGELIB member. membername is 1 to 6 alphanumeric or national characters; the first two characters are pre-defined by the system.

# OUTPUT JCL: PAGEDEF

## Overrides

The statements in the SYS1.IMAGELIB member specified by the OUTPUT JCL PAGEDEF parameter override the installation's PAGEDEF defaults in the PSF cataloged procedure.

The PSF uses the following parameters, in override order, to select the font list:

1. Font list in the SYS1.IMAGELIB member specified by an OUTPUT JCL PAGEDEF parameter.
2. DD CHARS parameter.
3. OUTPUT JCL CHARS parameter.
4. DD UCS parameter.
5. OUTPUT JCL UCS parameter.
6. JES installation default for the device.
7. Font list on the PAGEDEF parameter in the PSF cataloged procedure.

## Example of the PAGEDEF Parameter

```
//OUTDS1 OUTPUT PRMODE=PAGE,PAGEDEF=SSPGE
```

In this example, PSF is to print the output data set on a 3800 model 3 operating in page mode. The printing is to be done according to the parameters in the SYS1.IMAGELIB member SSPGE.

## PIMSG Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the PIMSG parameter to indicate that messages from a functional subsystem should or should not be printed in the output listing after the output data set.

The functional subsystem, Print Services Facility (PSF), accumulates messages in the PSF address space for errors that occur while processing line-mode data when PAGEDDEF is specified and while processing page-mode data. The system prints these messages at the end of the output data set, unless you code PIMSG=NO.

**Note:** PIMSG can be specified only for data sets printed on a 3800 Printing Subsystem Models 3 and 8.

**Syntax:**

```
PIMSG={ [YES|Y] }
        { [NO|N] }
```

### Subparameter Definition

#### YES

Requests that the system print the messages generated by a functional subsystem. This subparameter can also be coded as Y.

#### NO

Requests that the system not print the messages generated by a functional subsystem. This subparameter can also be coded as N.

### Defaults

If you do not code the PIMSG parameter and the output data set is printed by the PSF on a 3800 model 3, the default is YES.

### Example of the PIMSG Parameter

```
//DS17 OUTPUT PRMODE=PAGE,PAGEDDEF=H7MEM2,PIMSG=NO
```

In this example, PSF prints the output data set on a 3800 model 3 operating in page mode. PSF processes the data set according to the statements in SYS1.IMAGELIB member H7MEM2. The output does not contain PSF messages.

# OUTPUT JCL: PRMODE

## PRMODE Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the PRMODE parameter to identify the process mode required to print this system output data set. JES schedules the data set to a printer that can operate in the specified mode.

In a JES2 system, you can specify any of the modes that the installation defined during JES2 initialization. A JES2 installation can specify up to four modes for each device.

In a JES3 system, you can specify only LINE or PAGE.

**Syntax:**

```
PRMODE={LINE      }  
        {PAGE      }  
        {process-mode}
```

## Subparameter Definition

### LINE

Indicates that the data set is to be scheduled to a line-mode printer.

### PAGE

Indicates that the data set is to be scheduled to a page-mode printer.

### process-mode

Specifies the required process mode. The process-mode is 1 to 8 alphanumeric characters and must have been specified during JES2 initialization.

For an NJE-transmitted data set, use PRMODE to request specific processing without having to obtain output classes for the node that processes the data set.

*Note:* The process-mode subparameter is supported only on JES2 systems.

## Defaults

If you do not code the PRMODE parameter, JES schedules output processing as follows:

- If the system output data set does not contain page-mode formatting controls, to a printer with a process mode of line.
- If the system output data set contains page-mode formatting controls,
  - In a JES3 system, to a 3800 Printing Subsystem Model 3 running in page mode.
  - In a JES2 system, to any printer with a process mode of page.

## Printing a Line-Mode Data Set Using PSF

To print a line-mode data set using the Print Services Facility (PSF) and the enhanced capabilities of the 3800 model 3, code `PRMODE=PAGE`. The PSF formats this line-mode data set using the installation's default values for `PAGEDEF` and `FORMDEF` defined in the PSF cataloged procedure; if these defaults are unsatisfactory, code the `PAGEDEF` and `FORMDEF` parameters on the `OUTPUT JCL` statement.

## Example of the PRMODE Parameter

```
//DS18  OUTPUT  PRMODE=LINE
```

In this example, JES schedules the output data set to a printer with a process mode of line.

## PRTY Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the PRTY parameter to specify the priority at which the output data set enters the output queue. A data set with a higher priority is printed sooner.

**Syntax:**

```
PRTY=nnn
```

### Subparameter Definition

**nnn**

Specifies the initial priority. nnn is a decimal number from 0 through 255; 0 is the lowest priority while 255 is the highest.

### Defaults

If you do not code the PRTY parameter, JES3 uses an installation default specified at initialization. JES2 uses a priority that is calculated for all output.

### Overrides

In JES2 systems, the installation can specify at JES2 initialization that JES2 is to ignore the OUTPUT JCL PRTY parameter.

### Example of the PRTY Parameter

```
//PRESRPT OUTPUT PRTY=200,FORMS=TOPSEC
```

In this example, JES prints one copy of the president's report, PRESRPT, on forms named TOPSEC. Because a priority of 200 is specified, the report is probably printed immediately after entering the output queue.

## THRESHLD Parameter

**Parameter Type:** Keyword, optional, JES3 only

**Purpose:** Use the THRESHLD parameter to specify the maximum size for a sysout data set. JES3 calculates the sysout data set size as the number of records multiplied by the number of copies requested. When this size exceeds the THRESHLD value, JES3 creates a new unit of work, on a data set boundary, and queues it for printing. Consequently, copies of the sysout data set may be printed simultaneously by different printers.

Use the THRESHLD parameter for jobs that generate many large data sets or many copies of one large data set.

**Note:** THRESHLD is supported only on JES3 systems.

**Syntax:**

```
THRESHLD=limit
```

### Subparameter Definition

**limit**

Specifies the maximum number of records for a single sysout data set. limit is a decimal number from 1 through 99999999.

OUTPUT

### Defaults

If you do not code the THRESHLD parameter, JES3 uses an installation default specified at initialization.

# OUTPUT JCL: THRESHLD

## Example of the THRESHLD Parameter

```
//STEP4 EXEC PGM=RPTWRT
//SYSDS3 OUTPUT DEFAULT=YES,THRESHLD=10000
//RPT1 DD SYSOUT=A,COPIES=10
//RPT2 DD SYSOUT=A,COPIES=2
//RPT3 DD SYSOUT=A,COPIES=5
```

In this example, the report data sets, RPT1, RPT2, and RPT3, are processed in sysout class A. All three DD statements implicitly reference the step-level default OUTPUT JCL statement SYSDS3; therefore, the THRESHLD value specified in the OUTPUT JCL statement applies to the three reports combined. JES3 is to print the following:

COPIES	DATA SET	RECORDS IN DATA SET	TOTAL RECORDS
10	RPT1	1000	10000
2	RPT2	2000	4000
5	RPT3	500	2500
Total			16500

Because the total exceeds the THRESHLD limit, JES3 divides the sysout data sets into two units of work. RPT1 is printed as one unit, and the other two data sets are printed together as another unit. If the THRESHLD limit had been 20000, all three data sets would have been printed as one unit of work.



## TRC Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the TRC parameter to specify whether the output data set contains table reference character (TRC) codes or not. If so, a TRC code is the second character in each logical record; it immediately follows a carriage control character. The TRC code identifies which table-name in the CHARS parameter is to be used to print the record.

**Note:** TRC is supported only on JES3 systems. However, in a JES2 system, TRC can be specified for a data set processed in a functional subsystem address space (FSA); JES2 passes the TRC value to the FSA. Thus, TRC can be specified for a data set printed on a 3800 Printing Subsystem Model 3 by the Print Services Facility (PSF).

**Syntax:**

```
TRC={ [YES|Y] }
     { [NO|N ] }
```

### Subparameter Definition

#### YES

Indicates that the data set contains TRC codes. This subparameter can also be coded as Y.

#### NO

Indicates that the data set does not contain TRC codes. This subparameter can also be coded as N.

### Defaults

If you do not code the TRC parameter, the default is NO.

### Relationship to Other Parameters

A table reference character for the entire data set can be specified in the OUTPUT JCL MODIFY parameter.

# OUTPUT JCL: TRC

## Example of the TRC Parameter

```
//WRTR JOB ACNO77,MAEBIRD,MSGCLASS=B
//DS23 OUTPUT DEFAULT=YES,FORMS=STD,CONTROL=PROGRAM,TRC=YES
//STEP1 EXEC PGM=DLYRPT
//DAILY DD SYSOUT=A,CHARS=(GT12,GB12,GI12)
```

In this example, sysout DD statement **DAILY** implicitly references the default job-level **OUTPUT JCL** statement **DS23**. This **OUTPUT JCL** statement directs JES3 to print the daily report on standard forms. The sysout data set defined by DD statement **DAILY** contains carriage control characters in the first character of each logical record and a TRC code in the second character. The TRC characters in the records are 0 to use the font GT12; 1 to use GB12; and 2 to use GI12.

## JCS Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the UCS parameter to identify:

- The universal character set (UCS) image JES is to use in printing the system output data set.
- A print train (print chain or print band) JES is to use in printing the system output data set on an impact printer.
- A character-arrangement table for a data set printed on a 3800 Printing Subsystem in a JES2 system. In this use, the UCS parameter acts like a CHARS parameter.

The UCS image specifies the special character set to be used. JES loads the image into the printer's buffer. The UCS image is stored in SYS1.IMAGELIB. IBM provides the special character set codes in Figure 14-2.

**References:** For more information on the UCS parameter, see "Requesting a Special Character Set Using the UCS Feature" on page 7-59, and *SPL: Data Management*.

**Syntax:**

```
UCS=character-set-code
```

OUTPUT

## Subparameter Definition

### character-set-code

Identifies a universal character set. The character-set-code is 1 to 4 alphanumeric characters. See Figure 14-2 for IBM standard special character set codes.

## OUTPUT JCL: UCS

1403	3203 Model 5	3211	Characteristics
AN	AN	A11	Arrangement A, standard EBCDIC character set, 48 characters
HN	HN	H11	Arrangement H, EBCDIC character set for FORTRAN and COBOL, 48 characters
		G11	ASCII character set
PCAN	PCAN		Preferred alphanumeric character set, arrangement A
PCHN	PCHN		Preferred alphanumeric character set, arrangement H
PN	PN	P11	PL/I alphanumeric character set
QN	QN		PL/I preferred alphanumeric character set for scientific applications
QNC	QNC		PL/I preferred alphanumeric character set for commercial applications
RN	RN		Preferred character set for commercial applications of FORTRAN and COBOL
SN	SN		Preferred character set for text printing
TN	TN	T11	Character set for text printing, 120 characters
XN			High-speed alphanumeric character set for 1403, Model 2
YN			High-speed preferred alphanumeric character set for 1403, Model N1

*Note:* Where three values exist (for the 1403, 3211, and 3203 Model 5 printers), code any one of them. JES selects the set corresponding to the device on which the data set is printed.

Not all of these character sets may be available at your installation. Also, an installation can design character sets to meet special needs and assign a unique code to them. Follow installation procedures for using character sets.

Figure 14-2. Special Character Sets for the 1403, 3203 Model 5, and 3211 Printers

### Defaults

If you do not code the UCS parameter, the system checks the UCS image in the printer's buffer; if it is a default image, as indicated by its first byte, JES uses it. If it is not a default image, JES loads the UCS image that is the installation default specified at JES initialization.

On an impact printer, if the chain or train does not contain a valid character set, JES asks the operator to specify a character set and to mount the corresponding chain or train.

### Overrides

For printing on a printer with the UCS feature, a UCS parameter on the sysout DD statement overrides the OUTPUT JCL UCS parameter. For printing on a 3800, a CHARS parameter on the sysout DD statement or the OUTPUT JCL statement overrides all UCS parameters.

For a data set scheduled to the Print Services Facility (PSF), the PSF uses the following parameters, in override order, to select the font list:

1. Font list in the SYS1.IMAGELIB member specified by an OUTPUT JCL PAGEDEF parameter.
2. DD CHARS parameter.
3. OUTPUT JCL CHARS parameter.
4. DD UCS parameter.
5. OUTPUT JCL UCS parameter.
6. JES installation default for the device.

7. Font list on the PAGEDEF parameter in the PSF cataloged procedure.

See “PAGEDEF Parameter” on page 14-43 for more information.

## Using Special Characters Sets

To use a special character set, SYS1.IMAGELIB must contain an image of the character set, and the chain or train for the character set must be available. IBM provides standard special character sets, and the installation may provide user-designed special character sets.

## Example of the UCS Parameter

```
//PRTDS9 OUTPUT UCS=A11
```

In this example, JES uses standard EBCDIC character set arrangement A, with 48 characters, to print the output data set on a 3211 printer.

# OUTPUT JCL: WRITER

## WRITER Parameter

**Parameter Type:** Keyword, optional

**Purpose:** Use the WRITER parameter to name an external writer to process the output data set rather than JES. An external writer is an IBM- or installation-written program.

**References:** For information about external writers, see *SPL: Job Management*.

**Syntax:**

```
WRITER=name
```

## Subparameter Definition

**name**

Identifies an external writer. name is 1 to 8 alphanumeric characters.

Two names are reserved for JES: INTRDR for JES2 and STDWTR for JES3. Code INTRDR to specify that JES2 is to treat this data set as an input job stream. For more information, see "Specifying the Internal Reader" on page 7-52.

## Defaults

If you do not code the WRITER parameter, the installation's job entry subsystem processes the output data set.

## Overrides

The writer-name subparameter of the SYSOUT parameter on the sysout DD statement overrides the OUTPUT JCL statement WRITER parameter.

## Starting an External Writer

A system command, from the operator or in the input stream, must start an external writer before the system processes the data set.

## Example of the WRITER Parameter

```
//MYOUT JOB ACCT928,MAEBIRD,MSGCLASS=B
// START XWTR
//MYDS OUTPUT WRITER=XWTR
//STEP1 EXEC PGM=REPORT
//RPT1 DD SYSOUT=A,OUTPUT=MYDS
```

In a JES2 system, the second statement is a JCL command statement to start the IBM-supplied external writer. This writer is invoked by the XWTR cataloged procedure in SYS1.PROCLIB. The sysout DD statement RPT1 explicitly references OUTPUT JCL statement MYDS, which specifies that the external writer, XWTR, is to process the sysout data set.

---

```
/**START XWTR
//MYOUT JOB ACCT928,MAEBIRD,MSGCLASS=B
//MYDS OUTPUT WRITER=XWTR
//STEP1 EXEC PGM=REPORT
//RPT1 DD SYSOUT=A,OUTPUT=MYDS
```

This example is for a JES3 system.

... ..

... ..

... ..

... ..



## Chapter 15. Coding Special JCL Statements

This chapter details the coding of the special JCL statements. They are:

- Command statement
- Comment statement
- CNTL statement
- Delimiter statement
- ENDCNTL statement
- Null statement
- PEND statement
- PROC statement

These statements are arranged alphabetically in the following pages.

## Special JCL: JCL Command Statement

### JCL Command Statement

**Purpose:** Use the JCL command statement to enter an operator command through the input stream.

**Note:** The JCL command statement is supported only on JES2 systems. JES3 ignores all JCL command statements; use JES3 command statements.

The system usually executes an in-stream command as soon as it is read. Therefore, the command will **not** be synchronized with the execution of the job step to which it pertains. To synchronize a command with the job processing, tell the operator the commands you want and when they should be issued, and let the operator enter them from the console.

JES2 processes each command according to installation options for the input device from which the job was read.

**References:** For more information on commands and for descriptions of their parameters, see *System Commands*.

**Syntax:**

```
// command [parameters] [comments]
```

The command statement consists of the characters // in columns 1 and 2 and three fields: operation (command), parameter, and comments.

Do not continue a command statement.

### Command

Code the command as follows:

- Precede and follow the command with one or more blanks.
- Code the command or a valid abbreviation for the command. The following operator commands can be entered through the input stream.

CANCEL	MONITOR	SEND	STOP
CHNGDUMP	MOUNT	SET	STOPMN
DISPLAY	PAGEADD	SETDMN	UNLOAD
HOLD	RELEASE	SLIP	VARY
LOG	REPLY	START	WRITELOG
MODIFY	RESET		

## Special JCL: JCL Command Statement

### Parameter Field

Code any required parameters. When more than one parameter is coded, separate them by commas.

### Comments Field

The comments field follows the parameter field after at least one intervening blank.

### Location in the JCL

A command statement can appear immediately before a JOB statement, an EXEC statement, a null statement, or another command statement. However, a command statement must not be placed before the first JOB statement in an input stream.

If a command statement contains errors, it is not executed. If the erroneous statement is between two jobs in the input stream, the system does not issue a message to indicate that the command is not executed.

### Example of the Command Statement

```
// DISPLAY TS,LIST
```

In response to this command statement, the system displays the number and userid of all active time-sharing users of the system.

## Special JCL: Comment Statement

### Comment Statement

**Purpose:** Use the comment statement to enter a comment on the output listing. The comment statement is used primarily to document a program and its resource requirements.

**Syntax:**

```
/**comments
```

The comment statement consists of the characters `/**` in columns 1, 2, and 3 and one field: comments.

Code the comments in columns 4 through 80. The comments field does not need to be preceded or followed by blanks.

Do not continue a comment statement using continuation conventions. Instead, code additional comment statements.

### Location in the JCL

Place a comment statement anywhere after the `JOB` statement. You can place a comment statement between continuations of `JCL` statements.

### Listing of Comments Statements

Use the `MSGLEVEL` parameter on the `JOB` statement to request an output listing of all the `JCL` statements for your job. On this listing, comment statements have `***` in columns 1, 2, and 3.

### Example of the Comment Statement

```
/**THE COMMENT STATEMENT CANNOT BE CONTINUED,  
/**BUT IF YOU HAVE A LOT TO SAY, YOU CAN FOLLOW A  
/**COMMENT STATEMENT WITH MORE COMMENT  
/**STATEMENTS.
```

## NTL Statement

**Purpose:** Use the CNTL statement to mark the beginning of program control statements in the input stream. Program control statements specify control information for a subsystem.

The program control statements are ended by an ENDCNTL statement and are called a CNTL/ENDCNTL group. A DD statement CNTL parameter refers to an earlier CNTL statement. This reference to a CNTL statement lets the subsystem use the program control statements in the referenced CNTL/ENDCNTL group in processing the data set defined on the referencing DD statement.

**References:** For more information on the CNTL DD parameter, see “CNTL Parameter” on page 12-23.

The program control statements are documented in the publications for the subsystems. For example, see *IBM 3800 Printing Subsystem Models 3 and 8 Programmer's Guide* for information on program control statements for the Print Services Facility (PSF).

### Syntax:

```
//label CNTL * [comments]
```

The CNTL statement consists of the characters // in columns 1 and 2 and four fields: label, operation (CNTL), parameter (\*), and comments.

### Label Field

Code a label on every CNTL statement, as follows:

- Each label must be unique within the job.
- The label must begin in column 3.
- The label is 1 through 8 alphanumeric or national characters.
- The first character must be alphabetic or national.
- The label must be followed by at least one blank.

### Parameter Field

The parameter field must contain only an asterisk. The asterisk must be preceded and followed by at least one blank.

### Comments Field

The comments field follows the asterisk after at least one intervening blank.

## Special JCL: CNTL

### Location in the JCL

A CNTL statement can appear in a job step or a cataloged or in-stream procedure.

### Program Control Statements

Program control statements supply control information for a subsystem. A subsystem can require one or more program control statements. The one or more statements must be immediately preceded by a CNTL statement and immediately followed by an ENDCNTL statement.

Program control statements have the same format as JCL statements and follow all JCL coding rules.

Do not code a JCL statement within a program control group.

### Program Control Statements in Procedures

You can code symbolic parameters on program control statements in a cataloged or in-stream procedure.

You can override parameters on program control statements in a procedure. Follow the rules used for overriding DD statement parameters in a procedure. For more information, see "Modifying Parameters on a DD Statement" on page 9-6.

### Example of the CNTL Statement

```
//STEP1      EXEC  PGM=PRINT
//ALPHA      CNTL  *
//PRGCNTL    PRINTDEV BUFNO=20 ,PIMSG=YES ,DATAACK=BLOCK
//OMEGA      ENDCNTL
//AGAR       DD    UNIT=3800-3 ,CNTL=* .ALPHA
```

The PSF subsystem uses the BUFNO, PIMSG, and DATAACK options of the PRINTDEV control statement to print the data set for DD statement AGAR on a 3800 Model 3.

## Delimiter Statement

**Purpose:** Use the delimiter statement to indicate the end of data placed in the input stream.

**Syntax:**

```
/* [comments]
xx [comments]
```

A delimiter statement consists of the characters /\* or the two characters specified in a DD statement DLM parameter in columns 1 and 2 and one field: comments.

Code the comments in columns 4 through 80. The comments are preceded by at least one blank.

Do not continue a delimiter statement.

### Relationship to the DD Statement DLM Parameter

The system recognizes a delimiter other than /\* if a DLM parameter is coded on the DD \* or DD DATA statement that defines the in-stream data set.

If the data is preceded by a DD \* statement that does not contain a DLM parameter, a delimiter statement is optional. If the data is preceded by a DD DATA statement, a delimiter statement is required.

### Example of the Delimiter Statement

```
//JOB54 JOB , 'C BROWN', MSGLEVEL=(2,0)
//STEP1 EXEC PGM=SERS
//DD1 DD *
      .
      .
      data
      .
/* END OF DATA FOR DATA SET DD1
```

## Special JCL: ENDCNTL

### ENDCNTL Statement

**Purpose:** Use the ENDCNTL statement to mark the end of the program control statements following a CNTL statement.

**References:** For more information on program control statements, see "CNTL Statement" on page 15-5.

**Syntax:**

```
//[label] ENDCNTL [comments]
```

The ENDCNTL statement consists of the characters // in columns 1 and 2 and three fields: label, operation (ENDCNTL), and comments.

#### Label Field

Code a label on the ENDCNTL statement, as follows:

- Each label must be unique within the job.
- The label must begin in column 3.
- The label is 1 through 8 alphanumeric or national characters.
- The first character must be alphabetic or national.
- The label must be followed by at least one blank.

#### Comments Field

The comments field follows the ENDCNTL after at least one intervening blank.

#### Location in the JCL

The ENDCNTL statement immediately follows the one or more program control statements following a CNTL statement. Thus, the ENDCNTL statement can appear in a job step or in a cataloged or in-stream procedure.

#### Example of the ENDCNTL Statement

```
//STEP1      EXEC  PGM=PRINT  
//ABLE      CNTL  *  
//STATE1    PRINTDEV BUFNO=20,PIMSG=YES,DATAACK=BLOCK  
//BAKER     ENDCNTL  
//CALLER    DD    UNIT=3800-3,CNTL=*.ABLE
```



## Null Statement

Use the null statement to mark the end of a job.

*Note:* The null statement is supported only on JES3 systems.

### Syntax:

```
//
```

- The null statement consists of the characters // in columns 1 and 2.
- The rest of the statement **must** be blank.

### Location in the JCL

Place a null statement (1) at the end of a job's control statements and data and (2) at the end of an input stream.

The system can also recognize the end of a job when it reads the next JOB statement or when the input stream contains no more records.

A null statement that does not end an input stream should be immediately followed by a JOB statement. The system ignores statements between a null statement and the next valid JOB statement.

If a null statement follows a control statement that is being continued, the system treats the null statement as a blank comment field and assumes that the control statement contains no other parameters.

Spec JCL

### Example of the Null Statement

```
//MYJOB JOB      , 'C BROWN'
//STEP1 EXEC    PROC=FIELD
//STEP2 EXEC    PGM=XTRA
//DD1  DD       UNIT=3400-5
//DD2  DD       *
      .
      .
      data
      .
/*
//
```

The NULL statement indicates the end of job MYJOB.

## Special JCL: PEND

### PEND Statement

**Purpose:** Use the PEND statement to mark the end of an in-stream procedure.

**Syntax:**

```
//[name] PEND [comments]
```

The PEND statement consists of the characters // in columns 1 and 2 and three fields: name, operation (PEND), and comments.

Do not continue a PEND statement.

#### Name Field

A name is optional on the PEND statement. If used, code it as follows:

- Each name must be unique within the job.
- The name must begin in column 3.
- The name is 1 through 8 alphanumeric or national characters.
- The first character must be alphabetic or national.
- The name must be followed by at least one blank.

If a name is not coded, column 3 must be blank.

#### Comments Field

The comments field follows PEND after at least one intervening blank.

#### Location in the JCL

A PEND statement follows the statements of an in-stream procedure. Never place a PEND statement in a cataloged procedure.

#### Examples of the PEND Statement

```
//PROCEND1 PEND THIS STATEMENT IS REQUIRED FOR IN-STREAM PROCEDURES
```

This PEND statement contains a comment.

---

```
// PEND
```

This PEND statement contains only // and the operation field with the necessary blanks.

## ROC Statement

**Purpose:** The PROC statement **must** mark the beginning of an in-stream procedure and, optionally, may mark the beginning of a cataloged procedure. For either procedure, the PROC statement can assign default values to symbolic parameters, if coded, in the procedure.

### Syntax:

```
//[name] PROC [parameters] [comments]
```

A PROC statement consists of the characters // in columns 1 and 2 and four fields: name, operation (PROC), parameter, and comments.

**Multiple Parameters:** When more than one parameter is coded, separate parameters by commas. For example, //P1 PROC PARM1=OLD,PARM2=222001.

**Special Characters:** When a parameter value contains special characters, enclose the value in apostrophes. The enclosing apostrophes are not considered part of the value. For example, //P2 PROC PARM3='3400-6'.

Code each apostrophe that is part of a value as two consecutive apostrophes. For example, //P3 PROC PARM4='O'DAY'.

**Continuation onto Another Statement:** End each statement with a comma after a complete parameter. For example:

```
//P4 PROC PARM5=OLD,PARM6='SYS1.LINKLIB(P40)',
//      PARM7=SYSDA,PARM8='(CYL,(10,1))'
```

Spec JCL

### Name Field

A name is required on an in-stream PROC statement; it is optional on a PROC statement that begins a cataloged procedure. Code it as follows:

- Each name must be unique within the job.
- The name must begin in column 3.
- The name is 1 through 8 alphanumeric or national characters.
- The first character must be alphabetic or national.
- The name must be followed by at least one blank.

If a name is not coded, column 3 must be blank.

### Parameter Field

The parameters on a PROC statement assign default values to symbolic parameters on procedure statements. An in-stream PROC statement requires parameters only if the procedure contains symbolic parameters.

If coded, the parameter field must be preceded and followed by at least one blank.

# Special JCL: PROC

## Comments Field

The comments field follows the parameter field, if coded, or PROC, if not, after at least one intervening blank.

## Overrides

To override a default parameter value on a PROC statement, code the same parameter on the EXEC statement that calls the procedure.

## Using Symbolic Parameters

To assign a value to a symbolic parameter, code:

- The symbolic parameter, an equals sign, and the value.
- Omit the ampersand that precedes the symbolic parameter in the procedure.

For example, if a symbolic parameter on a DD statement is DSNNAME=&N, code on the PROC statement:

```
//P5 PROC N=MYDATA
```

*Note:* All symbolic parameters in a procedure must be assigned values when the procedure is to be executed. Otherwise, the system terminates the step that calls the procedure. To be safe, assign default values for all symbolic parameters on the PROC statement, even though you intend the calling EXEC statement to supply values for the symbolic parameters.

If a symbolic parameter is concatenated with other information, for example, &JOBNO.321, the value assigned to the symbolic parameter cannot exceed a combined total of 120 characters.

## Examples of the PROC Statement

```
//DEF      PROC      STATUS=OLD,LIBRARY=SYSLIB,NUMBER=777777
//NOTIFY   EXEC      PGM=ACCUM
//DD1     DD        DSNNAME=MGMT,DISP=( &STATUS,KEEP ),UNIT=3400-6,
//        VOLUME=SER=888888
//DD2     DD        DSNNAME=&LIBRARY,DISP=(OLD,KEEP),UNIT=3350,
//        VOLUME=SER=&NUMBER
```

Three symbolic parameters are defined in this cataloged procedure: &STATUS, &LIBRARY, and &NUMBER. Values are assigned to the symbolic parameters on the PROC statement. These values are used when the procedure is called and values are not assigned to the symbolic parameters on the calling EXEC statement.

---

```
//CARDS   PROC
```

This PROC statement can be used to mark the beginning of an in-stream procedure named CARDS.

## Chapter 16. Coding JES2 Control Statements

Code JES2 control statements with JCL statements to control the input and output processing of jobs. The rules for coding in Chapter 2, “Coding Conventions for JCL, JES2, and JES3 Statements” apply to the JES2 control statements.

### Location in the JCL

Place JES2 control statements, except the command and */\*PRIORITY* statements, after the JOB statement and its continuations. JES2 ignores JES2 control statements, except the command and */\*PRIORITY* statements, that appear before the JOB statement or between continued JOB statements.

Do not include JES2 control statements in a cataloged or in-stream procedure. JES2 ignores JES2 control statements in a procedure.

### Internal Reader

Use the following control statements when submitting jobs to the internal reader. The use of the internal reader is described in “Specifying the Internal Reader” on page 7-52, and in *SPL: JES2 Initialization and Tuning* and *SPL: Job Management*.

```
/*DEL  
/*EOF  
/*PURGE  
/*SCAN
```

JES2

## JES2: Command

### Command Statement

**Purpose:** Use the command statement to enter a JES2 operator command through the input stream, the internal reader, or the system console.

JES2 usually executes an in-stream command as soon as it is read. Therefore, the command will **not** be synchronized with the execution of the job step to which it pertains. To synchronize a command with the job processing, tell the operator the commands you want and when they should be issued, and let the operator enter them from the console.

Examples in this book illustrate the format for commands entered through the input stream. Commands entered through an operator console should not have /\* in columns 1 and 2.

**References:** For more information on the command statement and the JES2 verbs and operands, see *JES2 Commands*.

**Syntax:**

```
/*$command-verb,operand[,operand]... [N]
```

The JES2 command statement consists of:

- The characters /\* in columns 1 and 2.
- \$ or a character chosen by the installation in column 3.
- The command verb beginning in column 4.
- A comma.
- Operands up through column 71.
- N in column 72 if JES2 is **not** to write the command on the operator console.
- Blanks in columns 73 through 80. JES2 ignores these columns.

Do not continue command statements from one statement to the next, instead code as many command statements as you need.

### Parameter Definition

**command-verb**

Specifies the operator command that JES2 is to perform. You can enter the following JES2 commands in the input stream.

```
$A $E $I $O $T  
$B $F $L $P $TRACE  
$C $G $M $R $V  
$D $H $N $S $Z
```

**operand**

Specifies options for the command.

**N in column 72**

Indicates that JES2 is not to repeat the command on the operator console.

## Location in the JCL

Place command statements before jobs being entered through the input stream. JES2 ignores any command statements within a job.

If a job contains a JES2 /\*XMIT statement, place the command statement:

- Before the /\*XMIT statement if you want JES2 to process and display the command at the input node only.
- After the /\*XMIT statement if you want JES2 to process and display the command at the execution node only.

## Examples of the Command Statement

```
/*$SI3-5
```

This command statement starts initiators three through five. The command is \$\$ and the operand is I3-5. JES2 executes the command immediately and repeats the command on the operator console.

---

```
/*$TRDR1,H=Y
```

In response to this command, JES2 places all jobs being read by reader 1 in a hold status. If a job contains a JES2 /\*ROUTE XEQ or /\*XEQ control statement that specifies an execution node different from the input node, JES2 holds the job at the execution node, not the input node.

## JES2: /\*JOBPARM

### /\*JOBPARM Statement

**Purpose:** Use the /\*JOBPARM statement to specify job-related parameters for JES2.

**References:** For more information on job related processing options, see "Specifying Data Set Processing Options" on page 7-43.

**Syntax:**

```
/*JOBPARM parameter[,parameter]...
```

The parameters are:

```
{BURST|B}={Y|N}
{BYTES|M}=nnnnn
{CARDS|C}=nnnnnnn
{COPIES|N}=nnn
{FORMS|F}={xxxxxxxx|STD}
{LINECT|K}=nnn
{LINES|L}=nnnn
{NOLOG|J}
{PAGES|G}=nnnnn
{PROCLIB|P}=ddname
{RESTART|E}={Y|N}
{ROOM|R}=xxxx
{SYSAFF|S}={*|ANY|cccc}[,IND]
{TIME|T}=nnnn
```

The /\*JOBPARM statement consists of the characters /\* in columns 1 and 2, JOBPARM in columns 3 through 9, a blank in column 10, and parameters in columns 11 through 71. JES2 ignores columns 72 through 80.

You cannot continue a /\*JOBPARM statement, but you can code as many /\*JOBPARM statements as necessary in a given input stream.

You can code any number of the above parameters on a single /\*JOBPARM statement.

### Parameter Definition

**BURST={Y|N}**

Specifies the default burst characteristic of all sysout data sets that JES2 produces for this job. BURST is valid only when the data set is directed to a 3800 printer with the burst option installed.

**Y**

Burst the output listing for this data set.

**N**

Continuously fold the output listing for this data set.



**BYTES = nnnnn**

Specifies the maximum bytes of output the system is to produce for this job. The nnnnn is a decimal number, in thousands of bytes, from 0 through 99999.

When the job's output exceeds the value you specify in the BYTES parameter, JES2 sends a message to the operator. The job may be terminated, depending on the installation options specified at initialization.

**CARDS = nnnnnnn**

Specifies the estimated number of output cards for this job. The nnnnnnn is a value from 0 to 9999999.

**COPIES = nnn**

Specifies the number of times the spool lines or bytes are to be printed or punched. The nnn is a decimal number from 1 through 255. An installation can reduce the upper limit of this value during JES2 initialization.

The COPIES parameter is ignored and only one copy is produced if any of the following is true:

- The FREE parameter is coded on the output DD statement.
- HOLD = YES is coded on any sysout DD statement in the job.
- The class to which the data set is assigned is a held output class, and the message class is also a held class. The message class is specified in the JOB statement MSGCLASS parameter.

**FORMS = {xxxxxxxx|STD}**

Specifies the print and/or punch forms JES2 is to use for output data sets for which FORMS is not specified on the DD statement or on a JES2 /\*OUTPUT statement.

**xxxxxxxx**

Identifies the print or punch forms. The xxxxxxxx is 1 through 8 alphanumeric or national characters.

**STD**

Indicates that JES2 is to use the default specified at JES2 initialization.

**LINECT = nnn**

Specifies the maximum number of lines that JES2 is to print on each output page for job output. The nnn is a value from 0 through 255.

If you code LINECT = 0, JES2 does not eject to a new page when the number of output lines exceeds the page limit that the installation specified during JES2 initialization.

If you code LINECT on the /\*OUTPUT statement, it overrides the LINECT value on the /\*JOBPARM statement and the linect value in the accounting information parameter of the JOB statement.

## JES2: /\*JOBPARM

### **LINES = nnnn**

Specifies the maximum lines of output the system is to produce from this job. The nnnn is a decimal number, in thousands of lines, from 0 through 9999.

### **NOLOG**

Indicates that you do not want the job's JES2 hard-copy log as output. The job's hard-copy log lists the job-related console messages and operator replies produced during processing of the job.

### **PAGES = nnnnn**

Specifies the maximum number of pages of output the system is to produce for this job. The nnnnn is a value from 0 through 99999.

### **PROCLIB = ddname**

Requests a JES2 procedure library by its ddname in the system procedure library, SYS1.PROCLIB. These JES2 procedure library ddnames are in the format PROCnn, where nn is 1 or 2 decimal numbers from 1 through 99. The requested JES2 procedure library is used to convert the JCL for this job. If this parameter is omitted or appears but the library cannot be found, the library can be specified by class-related initialization parameters; otherwise, the default, PROC00, is used.

### **RESTART = {Y|N}**

Requests one of the following, if this job is executing before a re-IPL and JES2 warm start, and the job cannot restart from a step or checkpoint.

**Y**

Requests that JES2 queue the job for re-execution from the beginning of the job.

**N**

Requests that JES2 take no special action.

*Note:* If you do not specify RESTART, JES2 assumes N. However, the installation may override this default in JES2 initialization parameters.

### **ROOM = xxxx**

Indicates the programmer's room number. The xxxx is an alphanumeric value from 1 through 4 characters. JES2 places the room number on the job's separators for routing system output data sets back to the programmer.

### **SYSAFF = {\*|ANY|cccc}[,IND]**

Indicates the systems that are eligible to process the job. The parameter indicates from 1 through 7 system affinities.

**\***

Indicates the system that read the job.

**ANY**

Indicates any system in the JES2 multi-access spool configuration.

**cccc**

Indicates a specific system. cccc must be the 4-alphanumeric-character system id of one of the systems in the JES2 multi-access spool configuration. To specify more than one system, code: SYSAFF=(cccc,cccc,...).

## ,IND

After any of the other SYSAFF specifications, indicates that JES2 is to use system scheduling in independent mode.

## TIME = nnnn

Estimates the job execution time in minutes of real time. The nnnn is a value from 0 through 9999 minutes.

## Overrides

- The /\*JOBPARM statement parameters override the installation defaults specified at JES2 initialization.
- An OUTPUT JCL statement can override parameters on a /\*JOBPARM statement.
- A JES2 /\*OUTPUT statement can override parameters on a /\*JOBPARM statement.
- Any /\*JOBPARM statement parameter value overrides the equivalent parameter value from the accounting field in JES2 format of the JOB statement or from any preceding /\*JOBPARM statement in this job's JCL.

## Location in the JCL

Place the /\*JOBPARM statement after the JOB statement.

## Execution Node

JES2 normally processes /\*JOBPARM statements at the node of execution.

When you place a /\*JOBPARM statement **in front of** a /\*ROUTE XEQ or /\*XEQ statement, JES2 at the input node checks the /\*JOBPARM statement for syntax and parameter validity. After processing the /\*ROUTE XEQ or /\*XEQ statement, JES2 then passes the /\*JOBPARM statement to the execution node, where syntax and parameter validity are again checked.

When you place a /\*JOBPARM statement **after** a /\*ROUTE XEQ or /\*XEQ statement, JES2 passes the /\*JOBPARM to the execution node and performs all syntax and parameter validity processing at the execution node only.

**COPIES Parameter in Remote Processing:** In remote processing, the COPIES parameter on the /\*JOBPARM statement determines the number of output copies only when the execution node is a JES2 node. The /\*JOBPARM COPIES parameter is not supported by RSCS, DOS/VSE POWER, or JES3.

## JES2: /\*JOBPARM

### Example of the /\*JOBPARM Statement

```
/*JOBPARM L=60,R=4222,T=50,P=PROC03,N=5
```

The parameter specifications mean the following:

**L=60**

The job's estimated spool output will be 60,000 lines.

**R=4222**

The programmer's room is 4222. JES2 places this information in the separators for both printed and punched data sets.

**T=50**

The job's estimated execution time is 50 minutes.

**P=PROC03**

The procedure library that JES2 is to use to convert the JCL for this job is PROC03.

**N=5**

The estimated 60,000 lines of output will be printed five times.

## /\*MESSAGE Statement

**Purpose:** Use the /\*MESSAGE statement to send messages to the operator console when the job is read in by JES2.

**Syntax:**

```
/*MESSAGE message
```

The /\*MESSAGE statement consists of the characters /\* in columns 1 and 2, MESSAGE in columns 3 through 9, a blank in column 10, and the message starting in any column from 11 through 71. JES2 ignores columns 72 through 80.

### Relationship to the /\*ROUTE XEQ Statement

If the /\*MESSAGE statement is in a job that also contains a JES2 /\*ROUTE XEQ statement:

- Placing the /\*MESSAGE statement before the /\*ROUTE XEQ statement directs JES2 to send the message to the operators at the input node and the execution node.
- Placing the /\*MESSAGE statement after the /\*ROUTE XEQ statement directs JES2 to send the message only to the operator at the execution node.

### Location in the JCL

If the /\*MESSAGE statement is after the JOB statement, JES2 appends the job number to the beginning of the message.

If the /\*MESSAGE statement is not within a job, JES2 appends the input device name to the beginning of the message.

### Example of the /\*MESSAGE Statement

```
/*MESSAGE CALL DEPT 58 WHEN PAYROLL JOB IS FINISHED--EX.1946
```

This statement requests that the operator call department 58 at extension 1946 when the payroll job is complete.

## JES2: /\*NETACCT

### /\*NETACCT Statement

**Purpose:** Use the /\*NETACCT statement to specify an account number that is available to all the nodes in a network. JES2 uses the account number as is or translates it to local account numbers.

**Syntax:**

```
/*NETACCT network-account-number
```

The /\*NETACCT statement consists of the characters /\* in columns 1 and 2, NETACCT in columns 3 through 9, a blank in column 10, and the network account number starting in any column from 11 through 71. JES2 ignores columns 72 through 80.

### Parameter Definition

**network-account-number**

Specifies the job's accounting number. The network-account-number is 1 through 8 alphanumeric characters.

### Defaults

If no /\*NETACCT statement is specified, JES2 attempts to find a network account number using the local account number in a table lookup.

### Overrides

If you supply both a /\*NETACCT and a local account number, JES2 uses the local account number on the input node.

### Location in the JCL

Place the /\*NETACCT statement after the JOB statement.

If a job contains more than one /\*NETACCT statement, JES2 uses the network account number from the last statement.

JES2 ignores the /\*NETACCT statement on any node other than the input node.

### Example of the /\*NETACCT Statement

```
/*NETACCT NETNUM10
```

JES2 transmits the network account number, NETNUM10, with the job to the destination node.

## /\*NOTIFY Statement

**Purpose:** Use the /NOTIFY statement to direct a job's notification messages to a user.

**Note:** The /\*NOTIFY statement does not affect where the job is executed or where output is printed or punched.

**Syntax:**

```

                {          {userid }}
                {nodename{userid }}
/*NOTIFY      {          {/userid }}
                {(userid)}}
                {          }
                {userid   }

```

The /\*NOTIFY statement consists of the characters /\* in columns 1 and 2, NOTIFY in columns 3 through 8, a blank in column 10, and a parameter starting in any column from 11 through 71. JES2 ignores columns 72 through 80.

Do not code a comma, a right parenthesis, or a blank character in the nodename or userid.

### Parameter Definition

**nodename.userid**

**nodename:userid**

**nodename/userid**

**nodename(userid)**

Names the node and the user to be notified. The nodename and the userid are each 1 through 8 alphanumeric characters. They must be separated by a delimiter.

When you specify a nodename and userid, JES2 sets the origin node field in the job's network job header to the specified node, even though the job origin node may be different.

**userid**

Identifies a user. The userid is 1 through 8 alphanumeric characters.

When you specify only a userid, JES2 sends notification messages to that userid at the origin node.

### Overrides

The JES2 /\*NOTIFY statement overrides the NOTIFY parameter on the JOB statement.

## JES2: /\*NOTIFY

### Relationship to Other Control Statements

If you submit a job with a JOB statement NOTIFY parameter or the job includes a JES2 /\*NOTIFY statement, then the mode of the job (independent or not) must match that of the system at which the job is submitted. That is, for TSO-submitted jobs, you cannot change the system affinity using the JES2 /\*JOBPARM SYSAFF parameter.

### Examples of the NOTIFY Statement

```
/*NOTIFY VMNODE.VMUSER
```

JES2 sends notification messages to user VMUSER on node VMNODE.

---

```
/*NOTIFY TSOUSER
```

JES2 sends notification messages to user TSOUSER on the job's origin node.



## /\*OUTPUT Statement

**Purpose:** Use the /\*OUTPUT statement to specify characteristics and options for a system output data set or a group of system output data sets. This statement supplies processing options in addition to and in place of the options specified on the DD statement that defines the system output data set.

**References:** For more information on the /\*OUTPUT statement, see “Specifying Data Set Processing Options” on page 7-43.

**Note:** You should use the OUTPUT JCL statement in place of the /\*OUTPUT control statement because of the OUTPUT JCL statement’s enhanced output processing capabilities.

### Syntax:

```
/*OUTPUT code parameter[,parameter]...
```

The parameters are:

```
{BURST|B}={Y|N}
{CHARS|X}={xxxx|(xxxx[,xxxx]...)}
{CKPTLNS|E}=nnnnn
{CKPTPGS|P}=nnnnn
{COMPACT|Z}=nn
{COPIES|N}=(nnn[, (group-value[,group-value]...)] )
{COPYG|G}={nnn|(nnn[,nnn]...)}

        {LOCAL                }
        {name                  }
        {Nnnnn                 }
        {NnnRmmmm              }
        {NnnnRmmmm             }
        {NnnnnRmm              }
        {                        }
        {      {userid}        }
{DEST|D}={nodename{:userid }}
        {      {/userid}      }
        {      {(userid)}     }
        {Rnnnn                 }
        {RMnnnn                 }
        {RMTnnnn                }
        {Unnn                   }

{FCB|C}=xxxx
{FLASH|O}={NONE|(overlay-name[,count])}
{FLASHC|Q}=(overlay-name[,count])
{FORMS|F}={xxxx|STD}

{{INDEX|I}=nn }
{{LINDEX|L}=nn}

{LINECT|K}=nnn
{MODIFY|Y}=(module-name[,trc])
{MODTRC|M}=trc
{UCS|T}=xxxx
```

## JES2: /\*OUTPUT

The /\*OUTPUT statement consists of the characters /\* in columns 1 and 2, OUTPUT in columns 3 through 8, a blank in column 9, a code beginning in column 10, followed by a blank and the keyword parameters. JES2 ignores columns 72 through 80.

Code \* in column 10 to indicate that this /\*OUTPUT statement is a continuation of the previous /\*OUTPUT statement. An \* in column 10 causes JES2 to treat the /\*OUTPUT statement as a continuation, even through the previous /\*OUTPUT statement does not immediately precede the continuation.

Do not specify \* in column 10 on the first /\*OUTPUT statement in a job.

### Parameter Definition

#### code

Identifies the /\*OUTPUT statement. The code is 1 through 4 alphanumeric characters. A DD statement refers to a /\*OUTPUT statement by specifying this code in its code-name subparameter on the SYSOUT parameter. The referenced /\*OUTPUT statement specifies processing options for the system output data set defined in the referencing DD statement.

A code of \* indicates that this /\*OUTPUT statement is a continuation of the previous /\*OUTPUT statement.

*Note:* If you specify the code-name subparameter on a DD statement SYSOUT parameter in a job or job step that contains an OUTPUT JCL statement, JES2 uses the code-name as the name of an output form instead of as a reference to a /\*OUTPUT statement.

If more than one /\*OUTPUT statement has the same code starting in column 10, JES2 uses the parameters from the first /\*OUTPUT statement.

#### BURST = {Y|N}

Indicates the default burst characteristic of all output data sets that JES2 prints on a 3800 Printing Subsystem equipped with a burster-timmer-stacker in this job.

#### Y

Requests that the printed output is to be burst into separate sheets.

#### N

Requests that the printed output is to be in a continuous fanfold.

#### CHARS = xxxx

#### CHARS = (xxxx[,xxxx]...)

Specifies the name of a character-arrangement table for all output that JES2 prints on a 3800 Printing Subsystem in this job. The xxxx is 1 through 4 alphanumeric or national characters. Up to four names can be coded.

#### CKPTLNS = nnnnn

Specifies the maximum number of lines or cards contained in a logical page. The nnnnn is a decimal number from 0 through 32767 for printers and 1 through 32767 for punches. The default is specified in the JES2 initialization parameter for the device.

**CKTPGS = nnnnn**

Specifies the number of logical pages to be printed before the next checkpoint is taken. The nnnnn is a decimal number from 1 through 32767. The default is specified in the JES2 initialization parameter for the device.

**COMPACT = nn**

Specifies a compaction table for JES2 to use when sending this system output data set, which is a systems network architecture (SNA) data set, to a SNA remote terminal.

*Note:* The COMPACT parameter has no effect on compaction for NJE sessions; it applies only to SNA RJE sessions.

**COPIES = nnn****COPIES = (nnn[,group-value[,group-value]...])**

Specifies how many copies of the system output data set to be printed. The printed output is in page sequence for each copy.

For printing on a 3800 Printing Subsystem, this parameter can instead specify how many copies of each page are to be printed before the next page is printed.

If you route a job that has a COPIES parameter, the parameter will be used only if the receiving node is a JES2 node.

**nnn**

Specifies how many copies of the data set are to be printed; each copy will be in page sequence order. The nnn is a decimal number from 1 through 255, subject to an installation-specified limit. nnn is ignored for the 3800 if group values are specified.

If you omit or incorrectly code the nnn parameter of COPIES, it defaults to 1 and a warning message is issued.

**group-value**

Specifies how many copies of each page are to be printed before the next page is printed. Each group-value is a decimal number from 1 through 255. You can code a maximum of eight group-values. Their sum must not exceed 255 or the installation-specified limit. The total copies of each page equals the sum of the group-values.

*Note:*

- This subparameter is valid only for 3800 output.
- For 3800 output, this subparameter overrides the nnn subparameter.

The following are not valid:

- A null group-value, for example, COPIES = (5,(,)) or COPIES = (5,)
- A zero group-value, for example, COPIES = (5,(1,0,4))
- A null within a list of group-values, for example, COPIES = (5,(1,,4))

## JES2: /\*OUTPUT

**COPYG = nnn**

**COPYG = (nnn[,nnn]...)**

Specifies how many copies of each page are to be printed before the next page is printed. Each nnn is a decimal number from 1 to 255. You can code a maximum of eight group values. Their sum must not exceed 255. The total copies of each page equals the sum of the group values.

*Note:* This parameter is valid only for 3800 output. If you code COPYG and JES2 prints the data set on an impact printer, JES2 ignores COPYG.

**DEST = destination**

**DEST = (destination[,destination]...)**

Specifies one to four different destinations for the system output data set. The destination subparameters follow:

**LOCAL**

Indicates any local device.

**name**

Specifies a local or remote device by a symbolic name defined by the installation during JES2 initialization. The name is 1 through 8 alphanumeric or national characters.

**Nnnnn**

Specifies a node. nnnn is 1 through 4 decimal numbers from 1 through 1000. For example, N0103.

**NnnRmmmm**

**NnnnRmmm**

**NnnnnRmm**

Specifies a node and a remote work station connected to the node. The node number, indicated in the format by n, is 1 through 4 decimal numbers from 1 through 1000. The remote work station number, indicated in the format by m, is 1 through 4 decimal numbers from 1 through 4000. Do not code leading zeros in n or m. The maximum number of digits for n and m combined cannot exceed six.

*Note:* R0 is equivalent to LOCAL specified at node Nn.

**nodename.userid**

**nodename:userid**

**nodename/userid**

**nodename(userid)**

Identifies the nodename and userid of the destination node. Use this parameter to route a sysout data set between JES2 nodes and non-JES2 nodes. The nodename is 1 through 8 alphanumeric characters. The userid is 1 through 8 alphanumeric characters. Enclose userid in apostrophes when it contains special characters or begins with a number. Do not code a comma, right parenthesis, blank, R, RM, or RMT in the userid.

When using the nodename.userid form of the DEST parameter, you may use continuation statements to specify up to 4 destinations. The continuation statement must contain the characters /\* in columns 1 and 2, OUTPUT in columns 3 through 8, a blank in column 9, an \* in or following column 10, followed by one or more blanks, and the characters DEST = with the specified destinations. For example:

```
/*OUTPUT ABCD DEST=(POK.USER27, NYC.USER31)  
/*OUTPUT *   DEST=(BOCA.USER58, STL.USER22)
```

The form nodename:userid is used by a VM user who submits a job to an MVS system and wants the output returned to the VM user's virtual reader. The form nodename.userid makes the output print on the local VM printer.

*Note:* If a data set is queued for transmission and an operator changes its destination, the userid portion of the original routing is lost.

## **Rnnnn**

## **RMnnnn**

## **RMTnnnn**

Specifies a remote terminal. nnnn is 1 through 4 decimal numbers from 1 through 4000.

*Note:* R0 is equivalent to LOCAL.

## **Unnn**

Specifies a local terminal with special routing. nnn is 1 through 3 decimal numbers from 1 through 255.

## **FCB = xxxx**

Specifies the forms control buffer (FCB) image JES2 is to use to guide printing of the system output data set. The xxxx is 1 through 4 alphanumeric or national characters and is the last characters of a SYS1.IMAGELIB member name:

- FCB2xxxx member, for a 3211 Printer, a 3203 Printer Model 5, or a printer supported by systems network architecture (SNA).
- FCB3xxxx member, for a 3800 Printing Subsystem.
- FCB4xxxx member, for a 4248 Printer.

IBM provides two standard FCB images. Code STD1 or STD2 only to request them.

- STD1, which specifies 6 lines per inch on an 8.5-inch-long form.
- STD2, which specifies 6 lines per inch on an 11-inch-long form.

If the printer on which JES2 is to print the data set does not have the forms control buffer feature, JES2 sends the operator a message to mount the proper carriage control tape.

## JES2: /\*OUTPUT

**FLASH = NONE**

**FLASH = overlay-name**

**FLASH = (overlay-name[,count])**

Identifies the forms overlay to be used in printing the system output data set on a 3800 Printing Subsystem and, optionally, specifies the number of copies on which the forms overlay is to be printed.

**NONE**

Suppresses flashing for this data set.

**overlay-name**

Identifies the forms overlay frame that the operator is to insert into the printer before printing begins. The name is 1 to 4 alphanumeric or national characters.

Do not omit the overlay-name. The count subparameter is optional. If you omit it, you can omit the parentheses. However, if you omit it, you must not code it as a null; for example, FLASH=(ABCD,) is invalid.

Before printing starts, JES2 does not verify that the operator inserted the correct forms overlay frame for flashing.

**count**

Specifies the number, 1 through 255, of copies that JES2 is to flash with the overlay, beginning with the first copy printed.

JES2 determines the maximum number of copies to flash with the forms overlay by the value of nnn or the group value total on the COPIES parameter. If the count value of the FLASH parameter is greater than the value of the COPIES parameter, JES2 ignores the difference and uses the lower value.

The count subparameter of the FLASH parameter overrides the count subparameter of the FLASHC parameter.

**Defaults:** If you omit this parameter and did not specify FLASH on the DD statement or FLASHC on the /\*OUTPUT statement, JES2 uses the default specified at JES2 initialization.

If you specify an overlay-name without specifying a count, JES2 flashes all copies. That is, the default for count is 255. If you specify 0 for count, JES2 also flashes all copies.

**FLASHC = count**

Specifies the number, 0 through 255, of copies that JES2 is to flash with the overlay, beginning with the first copy printed.

*Note:* For the 3800 printer, if you specify FLASH and omit FLASHC, JES2 flashes all copies.

The count subparameter of the FLASH parameter overrides the count subparameter of the FLASHC parameter.

**FORMS = {xxxx|STD}**

Identifies the forms on which JES2 is to print or punch the system output data set.

**xxxx**

Identifies the print or punch forms. form-name is 1 through 4 alphanumeric or national characters.

**STD**

Indicates that JES2 is to use the default specified at JES2 initialization.

**INDEX = nn**

Sets the left margin for output on a 3211 Printer with the indexing feature. The width of the print line is reduced by the INDEX parameter value. The nn specifies how many print positions the left margin on the 3211 output is to be indented. nn is a decimal number from 1 through 31. n=1 indicates flush-left; n=2 through n=31 indent the print line by n-1 positions.

JES2 ignores the INDEX parameter on all printers except the 3211 with the indexing feature.

INDEX and LINDEX are mutually exclusive; if both are coded, JES2 uses the last one encountered.

**LINDEX = nn**

Sets the right margin for output on a 3211 Printer with the indexing feature. The width of the print line is reduced by the LINDEX parameter value. The nn specifies how many print positions the right margin on 3211 output is to be moved in from the full page width. nn is a decimal number from 1 through 31. n=1 indicates flush-right; n=2 through n=31 move the right margin over by n-1 positions.

JES2 ignores the LINDEX parameter on all printers except the 3211 with the indexing feature.

INDEX and LINDEX are mutually exclusive; if both are coded, JES2 uses the last one encountered.

**LINECT = nnn**

Specifies the maximum number of lines JES2 is to print on each output page. The nnn is a number from 0 through 255.

Specify LINECT=0 to keep JES2 from starting a new page when the number of lines exceeds the JES2 initialization parameter.

If you code LINECT on the /\*OUTPUT statement, it overrides the LINECT value on the /\*JOBPARM statement and the linect value in the accounting information parameter of the JOB statement.

If the LINECT parameter is omitted from the /\*OUTPUT statement, JES2 obtains the value from one of the following sources, in order:

1. The LINECT parameter on the /\*JOBPARM statement.
2. The linect field of the accounting information parameter on the JOB statement.
3. The value specified at JES2 initialization.

## JES2: /\*OUTPUT

**MODIFY = module-name**

**MODIFY = (module-name[,trc])**

Specifies a copy-modification module that tells JES2 how to print the system output data set on a 3800 Printing Subsystem. The module can specify legends, column headings, blanks, and where and on which copies the data is to be printed. The module is defined and stored on SYS1.IMAGELIB using the IEBIMAGE utility program.

**module-name**

Identifies a copy-modification module in SYS1.IMAGELIB. The module-name is 1 through 4 alphanumeric or national characters.

Do not omit the module-name.

**trc**

Identifies which table-name in the CHARS parameter is to be used. This **table reference character** is 0 for the first table-name specified, 1 for the second, 2 for the third, or 3 for the fourth.

If the trc value is greater than the number of table-names in the CHARS parameter, JES2 uses the first table named in the CHARS parameter.

The trc subparameter is optional. If you omit it, you can omit the parentheses. However, if you omit it, you must not code it as a null; for example, MODIFY=(TAB1,) is invalid. If you omit the trc subparameter, JES2 uses the first table-name.

The trc subparameter of the MODIFY parameter overrides the trc subparameter of the MODTRC parameter.

**MODTRC = trc**

Identifies which table-name in the CHARS parameter is to be used. This **table reference character** is 0 for the first table-name specified, 1 for the second, 2 for the third, or 3 for the fourth.

If the trc value is greater than the number of table-names in the CHARS parameter, JES2 uses the first table named in the CHARS parameter.

The trc subparameter of the MODIFY parameter overrides the trc subparameter of the MODTRC parameter.

**UCS = xxxx**

Identifies the universal character set (UCS) image JES2 is to use in printing the system output data set. The xxxx is 1 through 4 alphanumeric characters. See Figure 12-1 on page 12-128 for IBM standard special character set codes.

### Overrides

- /\*OUTPUT statement parameters override all equivalent DD statement parameters.
- If a /\*OUTPUT statement contains duplicate parameters, the last parameter overrides all preceding duplicates, except for the DEST parameter.



- Any parameter coded on subsequent /\*OUTPUT statements overrides the same parameter on previous /\*OUTPUT statements.
- JES2 adds any parameter you code on subsequent /\*OUTPUT statements that you did not code on previous /\*OUTPUT statements to the previous /\*OUTPUT statement.
- If you code LINECT on the /\*OUTPUT statement, it overrides the LINECT value on the /\*JOBPARM statement and the linect value in the accounting information parameter of the JOB statement.

### Relationship to Other Control Statements

- JES2 processes /\*OUTPUT statements placed after a /\*ROUTE XEQ statement at the execution node only.
- JES2 processes /\*OUTPUT statements placed before a /\*ROUTE XEQ statement at both the input node and the execution node.

### Location in the JCL

Place the /\*OUTPUT statement after the JOB statement.

### Example of the /\*OUTPUT Statement

```
/*OUTPUT ABCD COPIES=6,COPYG=(1,2,3),DEST=RMT23
```

This statement refers to all system output data sets defined by a DD statement that specifies SYSOUT=(c,,ABCD). Six copies of each page of output are printed. If the printer is a 3800, first one copy of each page is printed, then two copies of each page, and finally, three copies of each page. If the printer is not a 3800, COPYG is ignored and six copies of the entire data set are printed. The output is sent to remote terminal 23.

## JES2: /\*PRIORITY

### /\*PRIORITY Statement

**Purpose:** Use the /\*PRIORITY statement to assign the queue selection priority for your job and all of its output, except the JES2 hard-copy log.

A job with a higher priority is selected for execution sooner.

**Note:** Depending on the JES2 initialization options in use at your installation, JES2 may ignore the /\*PRIORITY statement.

**References:** For more information on the /\*PRIORITY statement, see "Assigning a Priority to a Job for JES2" on page 5-20.

#### Syntax:

```
/*PRIORITY p
```

The /\*PRIORITY statement consists of the characters /\* in columns 1 and 2, PRIORITY in columns 3 through 10, a blank in column 10, and the priority starting in any column from 11 through 71. JES2 ignores columns 72 through 80.

### Parameter Definition

**P**

Requests a priority. The p is a number from 0 through 15. The highest priority is 15.

Follow your installation's rules in coding a priority.

### Overrides

A priority specified on a /\*PRIORITY statement overrides a priority specified in the PRTY parameter on a JOB statement.

### Relationship to Other Control Statements

If a /\*PRIORITY statement is not present or if JES2 ignores the /\*PRIORITY statement, the system derives the priority from the following, in override order:

1. The PRTY parameter on the JOB statement.
2. The accounting information on a /\*JOBPARM statement.
3. The accounting information on the JOB statement.
4. An installation default specified at JES2 initialization.

**Location in the JCL**

The /\*PRIORITY statement must immediately precede the JOB statement. If not, or if **p** is not a number from 0 through 15, JES2 ignores the /\*PRIORITY statement and flushes the input stream until the next JOB statement or another /\*PRIORITY statement.

In a JES2 network, the /\*PRIORITY statement must immediately follow the /\*XMIT statement and precede a JOB statement. If the /\*PRIORITY statement does not immediately follow an /\*XMIT statement, JES2 ignores the /\*PRIORITY statement at any node except the input node.

**Example of the PRIORITY Statement**

```
/*PRIORITY 7
```

This statement assigns a job queue selection priority of 7. This value has meaning only in relation to other jobs in the system.

## JES2: /\*ROUTE

### /\*ROUTE Statement

**Purpose:** Use the /\*ROUTE statement to specify the destination of output that is not routed by a DEST parameter or to identify the network node where the job is to execute.

**References:** For more information, see "Routing a Job in a Network (JES2)" on page 3-7.

**Syntax:**

		{LOCAL	}
		{name	}
		{Nnnnn	}
		{NnnRmmmm	}
		{NnnnRmmm	}
		{NnnnnRmm	}
		{	[.userid ]}
	{PRINT}	{	[:userid ]}
/*ROUTE	{PUNCH}	{nodename	[/userid ]}
		{	[(userid)]}
		{Rnnnn	}
		{RMnnnn	}
		{RMTnnnn	}
		{Unnn	}
		{Nnnnn	}
		{	[.vmquestid ]}
		{	[:vmquestid ]}
/*ROUTE	XEQ	{nodename	[/vmquestid ]}
		{	[(vmquestid)]}

The /\*ROUTE statement consists of the characters /\* in columns 1 and 2; ROUTE in columns 3 through 7; at least one blank followed by PRINT, PUNCH, or XEQ; at least one blank followed by one of the destinations or nodes; and at least one blank before column 72. JES2 ignores columns 72 through 80.

Code only one destination or node on each /\*ROUTE statement.

### Parameter Definition

**PRINT**

Requests that JES2 route the job's printed output.

**PUNCH**

Requests that JES2 route the job's punched output.

**XEQ**

Requests that JES2 route the job to a network node for execution.

**LOCAL**

Indicates any local device at the node that submitted the job.

**name**

Specifies a local or remote device. The name is a symbolic name defined by the installation during JES2 initialization and is 1 through 8 alphanumeric or national characters.

**Nnnnn**

Specifies a node where the job is to be executed or the output printed or punched. nnnn is 1 through 4 decimal numbers from 1 through 1000.

**NnnRmmmm****NnnnRmmm****NnnnnRmm**

Specifies a node and a remote work station connected to the node. The node number, indicated in the format by n, is 1 through 4 decimal numbers from 1 through 1000. The remote work station number, indicated in the format by m, is 1 through 4 decimal numbers from 1 through 4000. Do not code leading zeros in n or m. The maximum number of digits for n and m combined cannot exceed six.

*Note:* R0 is equivalent to specifying LOCAL at node Nn.

**nodename.userid****nodename:userid****nodename/userid****nodename(userid)**

Specifies a node and a TSO or VM userid at that node. The node is a symbolic name defined by the installation during system initialization; nodename is 1 to 8 alphanumeric or national characters. The userid must be defined at the node; userid for TSO is 1 to 7 alphanumeric or national characters and for VM is 1 to 8 alphanumeric or national characters. Enclose userid in apostrophes when it contains special characters or begins with a number. For example, STL,'VM/370' and POK,'921PPC'. Do not code a comma, right parenthesis, blank, R, RM, or RMT in the userid.

A userid requires a node; therefore, code nodename.userid. You **cannot** code a userid without a nodename.

*Note:* If a data set is queued for transmission and an operator changes its destination, the userid portion of the routing is lost.

**Rnnnn****RMnnnn****RMTnnnn**

Specifies a remote terminal for the output. nnnn is 1 through 4 decimal numbers from 1 through 4000 or the maximum number of work stations assigned to JES2.

*Note:* R0 is equivalent to LOCAL.

When coding the long form, REMOTEnnn, on the JES2 /\*SIGNON statement, JES2 restricts RMT to 1 through 3 decimal numbers from 1 through 999.

## JES2: /\*ROUTE

### Unnn

Specifies a local terminal with special routing. nnn is 1 through 3 decimal numbers from 1 through 255.

### nodename.vmguestid

### nodename:vmguestid

### nodename/vmguestid

### nodename(vmguestid)

Identifies the network node where the job is to execute. The nodename identifies an MVS JES2 system, an MVS JES3 system, a VSE POWER node, or a VM system. Nodename should not specify the local node; if it does, the job executes locally. The nodename is 1 through 8 alphanumeric, national, or special characters specified during JES2 initialization.

The vmguestid identifies a guest system running in a virtual machine (VM), for example, an MVS system running under VM. Do not specify a work station or terminal in this parameter. The vmguestid is 1 through 8 alphanumeric, national, or special characters, except a comma, a right parenthesis, a blank, R, RM, or RMT.

## Location in the JCL

Place the /\*ROUTE statement after the JOB statement.

Place any /\*ROUTE XEQ statements before any DD \* or DD DATA statement.

## Processing of /\*ROUTE Statements

- JES2 processes /\*ROUTE XEQ statements on the input node only.
- When /\*ROUTE PRINT or /\*ROUTE PUNCH statement follows a /\*ROUTE XEQ statement, JES2 processes the /\*ROUTE PRINT or /\*ROUTE PUNCH statement on the execution node only.
- When a /\*ROUTE PRINT or /\*ROUTE PUNCH statement precedes a /\*ROUTE XEQ statement, JES2 processes the /\*ROUTE PRINT or /\*ROUTE PUNCH statement on both the input and execution nodes. However, printing or punching occurs at the node specified on the /\*ROUTE PRINT or /\*ROUTE PUNCH statement.

## Multiple /\*ROUTE Statements

JES2 uses the last /\*ROUTE statement of each category if more than one /\*ROUTE PRINT or PUNCH or XEQ statement appears in a job.

## Examples of the ROUTE Statement

```
/*ROUTE PRINT RMT6
```

This statement sends the printed output to remote terminal 6.

---

```
/*ROUTE PUNCH PUNCH2
```

This statement sends the punched output to device PUNCH2, which was identified to the system during initialization.

---

```
/*ROUTE XEQ DENVER
```

This statement sends the job to the node named DENVER for execution.

## JES2: /\*SETUP

### /\*SETUP Statement

**Purpose:** Use the /\*SETUP statement to indicate volumes needed for executing a part of the job.

**References:** For more information on the /\*SETUP statement, see "The JES2 SETUP Statement" on page 3-26.

**Syntax:**

```
/*SETUP volume-serial-number[,volume-serial-number]...
```

The /\*SETUP statement consists of the characters /\* in columns 1 and 2, SETUP in columns 3 through 7, a blank in column 10, and the volume serial number(s) starting in any column from 11 through 71. JES2 ignores columns 72 through 80.

Do not continue the /\*SETUP statement; code as many /\*SETUP statements as necessary.

### Parameter Definition

**volume-serial-number**

Identifies a volume required for execution of the job.

### Location in the JCL

Place all /\*SETUP statements after the JOB statement.

To prevent JES2 from requesting the setup on a node other than the node of execution, the /\*SETUP statement should follow any /\*ROUTE XEQ or /\*XEQ statement. If JES2 processes the /\*SETUP statement before processing the /\*ROUTE XEQ or /\*XEQ statement, JES2 requests the setup on both the input and execution nodes.

### Example of the /\*SETUP Statement

```
/*SETUP 666321,149658
```

When the job enters the system, JES2 issues a message to the operator console, asking the operator to mount the requested volumes. JES2 then places the job in hold status until the operator mounts the volumes and releases the job.



## /\*SIGNOFF Statement

**Purpose:** Use the /\*SIGNOFF statement to tell JES2 to terminate a remote job stream processing session. At the completion of the current print and/or punch streams, JES2 disconnects the station from the system. If JES2 is reading jobs from a remote station when the output completes, JES2 disconnects the remote station when the input is completed.

**Note:** The remote terminal access processor processes the /\*SIGNOFF statement if it is in a job stream.

Both systems network architecture (SNA) and binary synchronous communication (BSC) remote work stations can use the /\*SIGNOFF statement. SNA remote stations can also use the LOGOFF command to end a session with JES2. The LOGOFF command has some options that the /\*SIGNOFF statement does not provide.

**References:** For information on the LOGOFF command, see *SPL: JES2 Initialization and Tuning* and *SPL: VTAM*.

### Syntax:

```
/*SIGNOFF
```

The /\*SIGNOFF statement consists of the characters /\* in columns 1 and 2 and /\*SIGNOFF in columns 3 through 9.

### Example of the /\*SIGNOFF Statement

```
/*SIGNOFF
```

This statement requests that JES2 terminate a remote job stream processing session.

## JES2: /\*SIGNON

### /\*SIGNON Statement

**Purpose:** Use the /\*SIGNON statement to tell JES2 to begin a remote job stream processing session. The /\*SIGNON statement can override the remote identification number normally assigned to the remote station. This statement is optional for all work stations except non-multi-leaving remote stations on a switched line. For non-multi-leaving remote stations, JES2 transmits the /\*SIGNON statement alone as part of the initial connection process.

**Note:** The remote terminal access processor processes the /\*SIGNON statement if it is in a job stream. When the terminal access processor processes the /\*SIGNON statement, the line being processed is restarted.

Systems network architecture (SNA) remote work stations must use the LOGON command instead of the /\*SIGNON statement to notify JES2 of a connection request.

**References:** For information on the LOGON command, see *SPL: JES2 Initialization and Tuning* and *SPL: VTAM*.

#### Syntax:

/*SIGNON	{REMOTEnnn}	{RMTnnnn }	[password1]	[password2]
		{RMnnnn }		

The /\*SIGNON statement consists of the characters /\* in columns 1 and 2, SIGNON in columns 3 through 8, blanks in columns 9 through 15, and REMOTEnnn, or RMTnnnn, or RMnnnn starting in column 16. The /\*SIGNON statement can optionally contain two passwords: one beginning in column 25 and the other in column 73.

### Location in the JCL

Place the /\*SIGNON statement at the end of the JES2/RTP input stream for multi-leaving remote stations.

### Parameter Definition

#### REMOTEnnn

Specifies the identification number assigned to the remote station asking to sign on. The nnn is 1 through 3 decimal numbers from 1 through 999.

**Note:** Do **not** code any leading zeroes in nnn.

Code REMOTEnnn with the same characters as RMTnnn on the /\*ROUTE statement. If you code REMOTEnnn on the /\*SIGNON statement, you are restricted to coding RMTnnn with only three numbers on the /\*ROUTE statement.

**RMnnnn****RMTnnnn**

Specifies a remote station. nnnn is 1 through 4 decimal numbers from 1 through 4000.

**password1**

Specifies the password assigned to a switched connection that allows the remote station access to JES2 for remote job stream processing. The installation assigns this password during JES2 initialization. The operator can change or delete this password with the \$T command.

**password2**

Specifies the password for the remote station that is signing on; this password identifies the remote station as a valid remote job entry (RJE) station. The installation assigns this password during JES2 initialization.

**Examples of the /\*SIGNON Statement**

```
/*SIGNON      REMOTE123PSWD
```

This statement requests that remote station 123 begin a remote job stream processing session. PSWD, beginning in column 25, is the password assigned to the switched connection.

---

```
/*SIGNON      RMT1000  PSWD
```

This statement requests that remote station 1000 begin a remote job stream processing session. PSWD, beginning in column 25, is the password assigned to the switched connection.

## JES2: /\*XEQ

### /\*XEQ Statement

**Purpose:** Use the /\*XEQ statement to identify the network node where the job is to execute. It performs the same function as the /\*ROUTE XEQ statement.

**Syntax:**

```
/*XEQ {Nnnnn }  
      {nodename[.vmguestid]}
```

The /\*XEQ statement consists of the characters /\* in columns 1 and 2, XEQ in columns 3 through 5, a blank in column 6, and a node starting in any column starting with 7.

### Parameter Definition

**Nnnnn**

Specifies a node where the job is to be executed. nnnn is 1 through 4 decimal numbers from 1 through 1000.

**nodename**

Identifies the network node where the job is to execute. The nodename identifies an MVS JES2 system, an MVS JES3 system, a VSE POWER node, or a VM system. Nodename should not specify the local node; if it does, the job executes locally. The nodename is 1 through 8 alphanumeric, national, or special characters specified during JES2 initialization.

**vmguestid**

Identifies a guest system running in a virtual machine (VM), for example, an MVS system running under VM. Do not specify a work station or terminal in this parameter. The vmguestid is 1 through 8 alphanumeric, national, or special characters, except a comma, a right parenthesis, a blank, R, RM, or RMT.

### Location in the JCL

Place the /\*XEQ statement after the JOB statement. JES2 ignores the /\*XEQ statement for any node except the input node.

### Multiple /\*XEQ Statements

JES2 uses the node named on the last /\*XEQ statement if a job contains more than one /\*XEQ statement.

### Example of the XEQ Statement

```
/*XEQ ATLANTA
```

JES2 routes and executes this job on the node defined as ATLANTA.

## /\*XMIT Statement

**Purpose:** Use the /\*XMIT statement to transmit records from a JES2 node to either another JES2 node or an eligible non-JES2 node, for example, a VM or JES3 node. JES2 does not process or check the records for JES2 validity. JES2 builds header and trailer records from information on the JOB statement immediately preceding the /\*XMIT statement. Then JES2 transmits all the records following the /\*XMIT statement.

The records may consist of a job input stream or an in-stream DD \* or DD DATA data set. If the records are a job input stream and the destination node can process JCL, the transmitted input stream is executed.

The records end when JES2 reads in the input stream a delimiter:

/\*  
The two-character delimiter specified by a DLM parameter on this /\*XMIT statement

The records can also end when the input stream runs out of card images or, if the records are being read from an internal reader, the internal reader is closed.

### Syntax:

```

      {name                }
      {Nnnnn               }
/*XMIT {nodename[.userid ] } [,DLM=xx]
      {      [:userid ]   }
      {      [/userid ]   }
      {      [(userid)]   }
      {nodename[.vmguestid]}
      {      [:vmguestid]}
      {      [/vmguestid]}
      {      [(vmguestid)]}

```

The /\*XMIT statement consists of the characters /\* in columns 1 and 2, XMIT in columns 3 through 6, a blank in column 7, a nodename or node-number starting in any column starting with 8, and optionally followed, with no intervening blank, by a comma and the delimiter parameter.

Do not continue an /\*XMIT statement.

## Parameter Definition

### name

Specifies the destination node. The name is a symbolic name defined by the installation during JES2 initialization and is 1 through 8 alphanumeric or national characters.

### Nnnnn

Specifies the destination node. nnnn is 1 through 4 decimal numbers from 1 through 1000.

## JES2: /\*XMIT

### **nodename**

Identifies the destination network node. The nodename identifies an MVS JES2 system, an MVS JES3 system, a VSE POWER node, or a VM system. The nodename is 1 through 8 alphanumeric, national, or special characters specified during JES2 initialization.

### **userid**

Identifies the destination terminal or work station. The userid must be defined at the node. The userid for TSO is 1 to 7 alphanumeric or national characters and for VM is 1 to 8 alphanumeric or national characters. Enclose userid in apostrophes when it contains special characters or begins with a number. For example, STL,'VM/370' and POK,'921PPC'. Do not code a comma, right parenthesis, blank, R, RM, or RMT in the userid.

If the data set is queued for transmission and an operator changes its destination node, the userid portion of the destination is lost.

### **vmguestid**

Identifies the destination guest system running in a virtual machine (VM), for example, an MVS system running under VM. The vmguestid is 1 through 8 alphanumeric, national, or special characters, except a comma, a right parenthesis, a blank, R, RM, or RMT.

If the data set is queued for transmission and an operator changes its destination node, the vmguestid portion of the destination is lost.

### **DLM = xx**

Specifies two characters that indicate the end of the data being transmitted.

Code any two characters for the delimiter. If the specified delimiter contains any special characters, enclose it in apostrophes. In this case, a special character is any character that is neither alphanumeric nor national.

Failing to code enclosing apostrophes produces unpredictable results.

If the delimiter contains an ampersand or an apostrophe, code each ampersand or apostrophe as two consecutive ampersands or apostrophes. Each pair of consecutive ampersands or apostrophes counts as one character.

If you specify a DLM parameter, you must terminate the transmitted records with the characters in the DLM parameter. The characters you assign as delimiters override any delimiter implied by the defaults.

The characters // are not valid delimiters unless specifically indicated by DLM=//.

## **Defaults**

/\*

If you specify for DLM only one character or more than two characters, JES2 uses /\*.

**Location in the JCL**

Place the /\*XMIT statement immediately after a JOB statement.

Code only one /\*XMIT statement in a job.

**Example of the XMIT Statement**

```
/*XMIT ATLANTA,DLM=AA
```

JES2 transmits to the node ATLANTA all records following the /\*XMIT statement up to the specified delimiter, AA.





## Chapter 17. Coding JES3 Control Statements

Code JES3 control statements with JCL statements to control the input and output processing of jobs. The rules for coding in Chapter 2, “Coding Conventions for JCL, JES2, and JES3 Statements” apply to the JES3 control statements.

### Location in the JCL

Place JES3 control statements, except the command and `//*PAUSE` statements, after the `JOB` statement and its continuations. JES3 ignores JES3 statements, except the command and `//*PAUSE` statements, that appear before the `JOB` statement or between continued `JOB` statements.

Do not include JES3 control statements in a cataloged procedure. JES3 ignores JES3 control statements in a cataloged procedure.

### Internal Reader

Use the following control statements when submitting jobs to the internal reader. Use of the internal reader is described in “Specifying the Internal Reader” on page 7-52 and in *SPL: JES3 Initialization and Tuning*, and *SPL: Job Management*.

```
/*DEL  
/*EOF
```

# JES3

## Examples of JES3 Control Statements

The following shows JES3 control statements in relation to each other and to JCL statements for a job entered from a remote work station. No actual job should require all of these statements.

```
/**MESSAGE,CN1,ENTER A START COMMAND FOR THIS JOB
/**PAUSE
//TEST1 JOB ,,MSGCLASS=A
/**NETACCT PNAME=MAEBIRD,ACCT=2K14920
/**NET NETID=N1,NHOLD=0
/**PROCESS CI
/**PROCESS MAIN
/**PROCESS OUTSERV
/**DATASET DDNAME=STEP1.DD1
.
.
data
.
/**ENDDATASET
/**ENDPROCESS
/**OPERATOR THIS IS TEST JOB TEST1.
/**MAIN CLASS=C
/**FORMAT PR,DDNAME=STEP1.DD2,DEST=ANYLOCAL,COPIES=2
/**ROUTE XEQ NODE1
//FARJOB1 JOB ,,MSGCLASS=A
//STEP1 EXEC PGM=CHECKER
//DD1 DD DSNAME=INPUT
//DD2 DD SYSOUT=A
/*
```

---

The following is an ordinary job entered through the local input stream.

```
//RUN2 JOB ,,MSGCLASS=A
/**MAIN CLASS=B
/**FORMAT PR,DDNAME=STEPS.DD2,DEST=LOCAL,COPIES=5
//STEPS EXEC PGM=WRITER
//DD1 DD DSNAME=IN1,DISP=OLD,UNIT=3350,VOLUME=SER=MH2244
//DD2 DD SYSOUT=A
/*
```

## Command Statement

**Purpose:** Use the command statement to enter a JES3 operator command through the input stream or the operator console.

JES3 usually executes an in-stream command as soon as it is read. Therefore, the command will **not** be synchronized with the execution of the job step to which it pertains. To synchronize a command with the job processing, tell the operator the commands you want and when they should be issued, and let the operator enter them from the console.

**References:** For more information on the command statement and the JES3 verbs and operands, see *JES3 Operator's Library*.

### Syntax:

Entered through input stream:

```
/**command-verb[,operand],...
```

Entered through operator console:

```
*command-verb,[operand]...
```

- JES3 ignores columns 73 through 80.
- Do not continue a command statement on another statement.

## Parameter Definition

### \*command-verb

Indicates one of these JES3 commands:

Command	Short Form
CALL	X
CANCEL	C
DELAY	D
DISABLE	H
ENABLE	N
ERASE	E
FAIL	
FREE	
INQUIRY	I
MESSAGE	Z
MODIFY	F
RESTART	R
SEND	T
START	S
SWITCH	
VARY	V

You **cannot** specify a \*DUMP or \*RETURN command on a JES3 command statement.

## JES3: Command

### operand

Specifies an operand that pertains to the command-verb.

### Location in the JCL

- Place all command statements before the first JOB statement in the input stream, if jobs are also being submitted. JES3 treats any command statements that follow the JOB statement as comment statements.
- You can enter several command statements at one time.
- You can place command statements at the beginning of the cards in an active card reader that is being restarted.
- Command statements can be entered through card, tape, or disk readers.
- Command statements **cannot** be entered through an internal reader.

### Examples of the Command Statement

```
/**VARY,280,OFFLINE  
/**V,281,OFFLINE  
/**VARY,282,OFF  
  
/**V,280-282,OFF
```

In this example, the first three statements each vary one device offline. Alternatively, the fourth statement varies all three devices offline. If you place these cards in card reader 01C, for example, and it is currently not in use, the operator would then enter:

```
*X CR,IN=01C
```

---

```
/**MESSAGE,CN1,OUTPUT FROM JOB X REQUIRES SPECIAL CONTROLS
```

This statement instructs the operator from a remote location. Place this statement before the first job in the input stream.

**//\*DATASET Statement**

**Purpose:** Use the **//\*DATASET** statement to identify the beginning of an in-stream data set, which can contain JCL and/or data. The data set can be used as input to a dynamic support program (DSP), such as OUTSERV.

**Note:** Make sure the operator includes a **C** operand on the **\*CALL** command for the reader that reads a job containing this statement if it contains a **MODE=C** parameter.

**Syntax:**

```

//*DATASET DDNAME=ddname [ ,MODE={E|C} ] [ ,J={YES|NO} ] [ ,CLASS={MSGCLASS} ]
                               [ {C} ] [ {NO} ] [ {class} ]

```

**Parameter Definition****DDNAME = ddname**

Specifies the ddname of the DD statement that defines the spooled data set. The ddname can refer to only one DD statement in the job. You must qualify this ddname to identify the job step and, if appropriate, the procedure step, that contains the DD statement. Qualification is in the form **stepname.ddname** or **stepname.procstepname.ddname**. The ddname must match exactly the ddname on the DD statement.

**MODE = {E|C}**

Defines the card-reading mode.

**E**

Indicates that JES3 is to read the cards as EBCDIC with validity checking.

**C**

Indicates that JES3 is to read the cards in card image form, that is, in column binary or data mode 2.

**MODE=C** is not valid for jobs read from disk or tape, or for jobs submitted from remote work stations.

**J = {YES|NO}**

Indicates how JES3 is to recognize the end of the data set.

JES3 ignores the J parameter when you specify **MODE=C**, so you must terminate the data set with a **//\*ENDDATASET** statement.

**NO**

Indicates that a **JOB** statement terminates the data set.

**YES**

Indicates that a **//\*ENDDATASET** statement terminates the data set, because **JOB** statements may appear in the data set.

## JES3: **//\*DATASET**

**CLASS = {NO|MSGCLASS|class}**

Defines the output class that JES3 is to use for processing the job. If you omit the CLASS parameter, the default is NO.

**NO**

Indicates that the system is to assign a default output class.

**MSGCLASS**

Indicates that the data set has the same output class as the MSGCLASS specified on the JOB statement.

**class**

Specifies the output class to be used.

### Examples of the **//\*DATASET** Statement

```
//*PROCESS OUTSERV
//*DATASET DDNAME=MYPRINT, J=YES
.
.
data
.
.
//*ENDDATASET
//*FORMAT PR, DDNAME=MYPRINT, COPIES=5
```

In this example, the **//\*DATASET** statement marks the beginning of the in-stream data set MYPRINT. The **//\*FORMAT PR** statement requests five copies of it. A **//\*ENDDATASET** statement marks the end of the data set.

---

```
//JOB1 JOB
//MYPRC PROC
//PROC1 EXEC PGM=FETCH
//DD6 DD DSNAME=TYPE2
.
.
// PEND
//STEP1 EXEC PROC=MYPRC
.
.
//*DATASET DDNAME=STEP1.PROC1.DD6, J=YES
.
.
data
.
.
//*ENDDATASET
//STEP2 EXEC PGM=A
```

In this example, the **//\*DATASET** statement marks the beginning of the in-stream data set defined in DD statement DD6. Note the qualification in the ddname: STEP1 identifies the EXEC statement that calls the in-stream procedure, PROC1 is the procedure step that contains the DD statement, and DD6 is the ddname of the DD statement. The end of the data set is indicated by a **//\*ENDDATASET** statement.

## **/\*ENDDATASET Statement**

**Purpose:** Use the **/\*ENDDATASET** statement to indicate the end of an in-stream data set that was begun with a **/\*DATASET** statement.

**Syntax:**

```
/*ENDDATASET
```

### **Location in the JCL**

The **/\*ENDDATASET** statement must appear immediately after the last record for the data set.

### **Example of the /\*ENDDATASET Statement**

```
/*DATASET DDNAME=INFO,J=YES  
.  
data  
.  
/*ENDDATASET
```

In this example, the **/\*ENDDATASET** statement marks the end of the in-stream data set INFO.

## JES3: **//\*ENDPROCESS**

### **//\*ENDPROCESS Statement**

**Purpose:** Use the **//\*ENDPROCESS** statement to indicate the end of a series of **//\*PROCESS** statements. The **//\*ENDPROCESS** statement is optional when a JCL statement follows the last **//\*PROCESS** statement in the job, but is required when no JCL statements follow the last **//\*PROCESS** statement.

**Syntax:**

```
//*ENDPROCESS comments
```

### **Location in the JCL**

Do not place any **//\*PROCESS** statements after the **//\*ENDPROCESS** statement.

### **Example of the **//\*ENDPROCESS** Statement**

```
//*ENDPROCESS    END OF PROCESS STATEMENTS
```



**//\*FORMAT PR Statement**

**Purpose:** Use the **//\*FORMAT PR** statement to specify to JES3 processing instructions for print data sets. These instructions permit special processing of output data sets, such as:

- Multiple destinations.
- Multiple copies of output with different attributes.
- Forced single or double space control.
- Printer overflow checking.

You can code several **//\*FORMAT PR** statements for a data set to specify special requirements for different copies of the data set. You can also code a **//\*FORMAT PU** statement for the same data set, thereby both printing and punching it.

Output classes are established at JES3 initialization. These classes group output data sets with similar characteristics for job output. You should determine if you should use one of these classes or if you should code a **//\*FORMAT PR** statement.

**Note:** The **//\*FORMAT PR** statement applies only to output data sets printed by JES3. The statement is ignored for data sets sent to a TSO userid or processed by an external writer.

**References:** For more information, see “Specifying Data Set Processing Options” on page 7-43.

**Syntax:**

```
//*FORMAT PR,DDNAME={,|ddname|SYMSG|JESJCL|JESMSG}[ ,parameter]...
```

The parameters are:

```
[CARRIAGE={6|carriage-tape-name}]
[FCB={6|image-name} ]
CHARS={STANDARD|(table-name[,table-name]...)}
CHNSIZE={DS|(nnn[,mmm])}
COMPACT=compaction-table-name
CONTROL={PROGRAM|SINGLE|DOUBLE|TRIPLE}
COPIES={{nnn|1}|group-value[,group-value]...}
      {ANYLOCAL }
      {device-name }
      {device-address }
      {group-name }
DEST={nodename[.remote] }
      {LOCAL }
      {( [,device-name ])}
      {(type[,device-address])}
      {( [,group-name ])}
      {( [,LOCAL ])}
EXTWTR=name
FLASH={STANDARD|(overlay-name[,count])}
FORMS={STANDARD|form-name}
MODIFY=(module-name[,trc])
OVFL={ON|OFF}
PRTY=nnn
STACKER={STANDARD|S|C}
THRESHLD=limit
TRAIN={STANDARD|train-name}
```

# JES3: **//\*FORMAT PR**

## Parameter Definition

### **PR**

Indicates that this statement is associated with a print data set.

**DDNAME =** {,|**ddname**|**SYSMSG**|**JESJCL**|**JESMSG**}

Indicates a null DDNAME parameter. This null makes the parameters on this **//\*FORMAT PR** statement the defaults for the job. These parameters then apply to all of the job's print data sets, except those covered by a **//\*FORMAT PR** statement with **DDNAME = ddname**.

**Effect of DDNAME =, Parameter:** A **//\*FORMAT PR** statement that contains **DDNAME =,** is called **nonspecific**; a statement that contains **DDNAME = ddname** is called **specific**.

When you code a nonspecific **//\*FORMAT PR** statement, the parameters you specify become the defaults for the job. These parameters apply to all of the job's print data sets unless overridden by a specific **//\*FORMAT PR** statement.

Parameters coded on a nonspecific **//\*FORMAT PR** statement are overridden by parameters coded on sysout DD statements or by parameters in the JES3 SYSOUT initialization statement.

### **ddname**

Specifies the ddname of the DD statement that defines the data set you want to print. The ddname can refer to only one DD statement in the job. You must qualify this ddname to identify the job step and, if appropriate, the procedure step, that contains the DD statement. Qualification is in the form **stepname.ddname** or **stepname.procstepname.ddname**. The ddname must match exactly the ddname on the DD statement. (See the example for the **//\*DATASET** statement.)

### **SYSMSG**

Requests printing of system messages.

### **JESJCL**

Requests printing of JCL statements and messages.

### **JESMSG**

Requests printing of JES3 and operator messages about the job.

**CARRIAGE =** {**6**|**carriage-tape-name**}

Specifies the carriage tape for the 3211, 3203 Model 5, or 1403 Printer for printing this output class.

### **6**

Indicates the installation standard carriage tape.

### **carriage-tape-name**

Identifies the name of the carriage tape. The name is 1 through 8 characters.

For the 3211 and 3203-5, SYS1.IMAGELIB must contain a module for each carriage tape name.

*Note:* You cannot code both the CARRIAGE and FCB parameters on the same **//\*FORMAT PR** statement.

**CHARS = {STANDARD|(table-name[,table-name]...)}**

Specifies the name of one or more character-arrangement tables for printing the data set on a 3800 Printing Subsystem.

You can omit the parentheses if you code only one table-name.

## **STANDARD**

Indicates the standard character-arrangement table, which was specified at JES3 initialization.

## **table-name**

Specifies the name of a character-arrangement table. Each table-name is 1 to 4 alphanumeric or national characters. When coding more than one table-name, parentheses are required around the list and null positions are invalid in the list.

**CHNSIZE = {DS|(nnn[,mmm])}**

Specifies the number of logical records to be transmitted to a work station as a systems network architecture (SNA) chain and indicates whether normal output checkpoints are to be taken for this data set.

*Note:* This parameter is valid only when transmitting to a SNA work station.

Be careful in selecting subparameters, because each affects performance differently. Sending the data set as a SNA chain provides the best performance, but can cause duplicate data to be written to the output device if operator intervention is required. The remote operator can eliminate duplicate data by issuing commands to reposition and restart the output writers.

When an end-of-chain indicator is sent in the data set, JES3 takes an output checkpoint. You can provide additional checkpoints for critical data by sending an end-of-chain indicator. For example, when printing bank checks, you can have an output checkpoint taken for each check by specifying each check as a SNA chain.

## **DS**

Indicates that the data set is to be sent as a single SNA chain and that JES3 is to take normal output checkpoints. DS is the default.

## **nnn**

Specifies the SNA chain size in pages. nnn is a decimal number from 1 through 255. The size of a page is determined by:

- The value of mmm.
- The carriage control characters in the data that skip to channel 1.

## **mmm**

Specifies the number of logical records in a page, when the data contains no carriage control characters. mmm is a decimal number from 1 through 255.

## JES3: **//\*FORMAT PR**

### **COMPACT = compaction-table-name**

Specifies the compaction table for JES3 to use when sending a systems network architecture (SNA) data set to a SNA remote terminal. The compaction-table-name is a symbolic name defined by the installation during JES3 initialization. The name is 1 to 8 alphanumeric characters.

JES3 performs compaction using an installation-defined default table and writes a message to the console when:

- You do not specify a compaction table.
- You specify an invalid compaction table.
- You specify a compaction table that JES3 cannot find.

If the installation has not defined a default table, JES3 sends the data without compacting it.

If the remote printer does not support compaction, JES3 ignores the COMPACT parameter and sends the data without compacting it.

### **CONTROL = {PROGRAM|SINGLE|DOUBLE|TRIPLE}**

Indicates either that each logical record starts with a carriage control character or that the output is to be printed with single, double, or triple spacing.

#### **PROGRAM**

Indicates that each logical record in the data set begins with a carriage control character. The carriage control characters can be in either the extended USASCII code or the actual channel command code. The carriage control characters are given in *Data Management Services*.

#### **SINGLE**

Requests forced single spacing.

#### **DOUBLE**

Requests forced double spacing.

#### **TRIPLE**

Requests forced triple spacing.

### **COPIES = {nnn|1|group-value[,group-value]...}**

Indicates how many copies of the data set are to be printed. If a COPIES parameter is not specified, the default is 1.

#### **nnn**

Specifies how many copies of the data set are to be printed; each copy will be in page sequence order. nnn is a number from 0 through 255. If you code COPIES=0, JES3 does not print this data set.

You can omit the parentheses if you code only nnn.

JES3 ignores nnn if any group-values are specified.

**group-value**

Specifies how many copies of each page are to be printed before the next page is printed. Each group-value is a number from 1 through 255. You can code a maximum of eight group-values. Their sum must not exceed 255. The total copies of each page equals the sum of the group-values.

This subparameter is valid only for 3800 output. It overrides an nnn subparameter.

**DEST = destination**

Routes the output to a printer. This parameter overrides the **//\*MAIN** statement **ORG** parameter.

If you omit **DEST**, JES3 assigns the first available printer that is in the origin group and that fulfills all processing requirements. The origin group is the group of printers defined for the local or remote submitting location. If the job originated at a remote job processing (RJP) terminal, JES3 returns the output to the originating terminal group.

**ANYLOCAL**

Indicates any local device (a printer being used for the output class specified on the **DD** statement) attached to the global processor.

**device-name**

Specifies a local device by a symbolic name defined by the installation during JES3 initialization. device-name is 1 to 8 alphanumeric or national characters.

**device-address**

Specifies the 3-character physical device address.

**group-name**

Specifies a group of local devices, an individual remote station, or a group of remote stations by a symbolic name defined by the installation during JES3 initialization. group-name is 1 to 8 alphanumeric or national characters.

**nodename**

Specifies a node by a symbolic name defined by the installation during JES3 initialization. nodename is 1 to 8 alphanumeric or national characters.

**remote**

Specifies a remote work station or VM userid to which the receiving node directs output. remote is 1 to 8 characters.

**LOCAL**

Indicates any local device that is attached to the global processor and that does not belong to a group.

*Note:* When you code **DEST = LOCAL**, your installation must have at least one local device that is not assigned to a group.

**(type)**

Specifies a device classification. type is in the form (**gggssss**) where **ggg** is the general device classification and **ssss** is the specific device classification. The type must be enclosed in parentheses. The type must be defined by the installation during JES3 initialization. For example, type for a 3800 is (PRT3800).

## JES3: **//\*FORMAT PR**

### **EXTWTR = name**

Identifies the external writer that is to process the data set at the destination node. name is 1 to 8 alphanumeric characters and must identify a module defined to the remote JES3 node that is to execute the job.

### **FCB = {6image-name}**

Specifies the forms control buffer (FCB) image JES3 is to use to guide printing of the output data set by a 3211 Printer, 3203 Printer Model 5, or 3800 Printing Subsystem or by a printer supported by systems network architecture (SNA) remote job processing (RJP).

If the data set is to be produced on some other device, JES3 ignores the FCB parameter.

**6**

Indicates the standard FCB. JES3 uses the standard FCB specified at JES3 initialization.

### **image-name**

Specifies the name of the FCB image. The name is 1 to 4 alphanumeric or national characters and is the last characters of a SYS1.IMAGELIB member:

- The FCB2xxxx member for a 3211, 3203 model 5, or printer supported by SNA.
- FCB3xxxx member for a 3800.

*Note:* You cannot code both the CARRIAGE and FCB parameters on the same **//\*FORMAT PR** statement.

### **FLASH = {STANDARD|(overlay-name[,count])}**

Identifies the forms overlay to be used in printing the output data set on a 3800 Printing Subsystem and, optionally, to specify the number of copies on which the forms overlay is to be printed.

You can omit the parentheses if you code only one overlay-name.

If you omit the count subparameter or specify a count of 0, JES3 flashes all copies with the specified overlay.

### **STANDARD**

Indicates the standard forms flash overlay. JES3 uses the standard forms overlay specified at JES3 initialization.

### **overlay-name**

Identifies the forms overlay frame that the operator is to insert into the printer before printing begins. The name is 1 to 4 alphanumeric or national characters.

### **count**

Specifies the number of copies, 0 to 255, of copies that JES3 is to flash with the overlay, beginning with the first copy printed. If you specify a count of 0, JES3 flashes all copies.

*Note:* See the *Forms Design Reference Guide for the IBM 3800 Printing Subsystem* for information on designing and making forms overlays.

**FORMS = {STANDARD|form-name}**

Specifies the forms on which the output data set is to be printed.

**STANDARD**

Indicates the standard form. JES3 uses the standard form specified at JES3 initialization.

**form-name**

Specifies the print forms. form-name is 1 to 8 alphanumeric characters.

**MODIFY = (module-name[,trc])**

Specifies a copy modification module that tells JES3 how to print the output data set on a 3800 Printing Subsystem. The module can specify how to replace blanks or data in the data set. You can omit the parentheses if you code only a module-name.

The module is defined and stored in SYS1.IMAGELIB using the IEBIMAGE utility program. See the *IBM 3800 Printing Subsystem Programmer's Guide* for more information.

If you omit the trc subparameter, JES3 prints the data set with the first character-arrangement table coded in the CHARS parameter.

**module-name**

Identifies a copy modification module in SYS1.IMAGELIB. module-name is 1 to 4 alphanumeric or national characters.

**trc**

Identifies which table-name in the CHARS parameter is to be used. This table reference character is 0 for the first table-name specified, 1 for the second, 2 for the third, or 3 for the fourth.

**OVFL = {ON|OFF}**

Specifies whether or not the printer program should test for forms overflow.

Because the overflow test is a responsibility of the terminal package for the remote RJP terminal, JES3 ignores OVFL for remote job processing. For additional information on the use of OVFL, see "Requesting Printer Form and Character Control" on page 7-58.

**ON**

Indicates that the printer program should eject whenever the end-of-forms indicator (channel 12) is sensed.

**OFF**

Indicates that forms overflow control is not to be used.

**PRTY = nnn**

Specifies the priority at which the data set enters the output queue. nnn is a decimal number from 0 through 255; 0 is the lowest priority while 255 is the highest.

## JES3: **//\*FORMAT PR**

### **STACKER = {STANDARD|S|C}**

Specifies the stacker for the 3800 Printing Subsystem output.

#### **STANDARD**

Indicates the standard installation default. This default is specified at JES3 initialization.

#### **S**

Indicates the burster-trimmer-stacker, in which the output is burst into separate sheets.

#### **C**

Indicates the continuous forms stacker, in which the output is left in continuous fanfold.

### **THRESHLD = limit**

Specifies the maximum size for the output data set. JES3 calculates the output data set size as the number of records multiplied by the number of copies requested. When this size exceeds the THRESHLD value, JES3 creates a new unit of work, on a data set boundary, and queues it for printing. Consequently, copies of the output data set may be printed simultaneously by different printers.

Use the THRESHLD parameter for jobs that generate many large data sets. Grouping data sets as a single unit of work for an output service writer may decrease the time required for the output service writer to process the data sets.

The value specified in this parameter overrides the value specified during JES3 initialization.

#### **limit**

Specifies the maximum records for a single output data set. limit is a decimal number from 1 through 99999999. The default is 99999999.

### **TRAIN = {STANDARD|train-name}**

Specifies the printer train to be used in printing the output data set. See "Requesting a Special Character Set Using the UCS Feature" on page 7-59 for the IBM-supplied trains. Because these trains are not standard machine features, verify that the installation has the required printer train before specifying it.

Do not code the TRAIN parameter for output destined for a remote job processing (RJP) terminal.

#### **STANDARD**

Indicates the standard installation default. This default is specified at JES3 initialization.

#### **train-name**

Specifies an installation-supplied printer train. Check with your installation for the names of trains.



## **Relationship to Sysout DD and OUTPUT JCL Statements**

- JES3 ignores the processing options specified on a default **//\*FORMAT** statement when a sysout DD statement explicitly or implicitly references an **OUTPUT JCL** statement.
- JES3 ignores the processing options specified on a default **OUTPUT JCL** statement when a **//\*FORMAT** statement explicitly references a sysout DD statement.
- When a sysout DD statement explicitly references an **OUTPUT JCL** statement and a **//\*FORMAT** statement explicitly references the same DD statement, the processing options from both the **OUTPUT JCL** and **//\*FORMAT** statements apply. Two separate sets of output are created from the data set defined by the sysout DD statement; one according to the processing options on the **OUTPUT JCL** statement, and the other according to the processing options on the **//\*FORMAT** statement.

## **Relationship to **//\*PROCESS** Statement**

JES3 accumulates **//\*FORMAT PR** statements within a job and applies them to any JES3 **//\*PROCESS** statement that is normally affected by a **//\*FORMAT PR** statement.

## **Examples of the **//\*FORMAT PR** Statement**

```
//*FORMAT PR,DDNAME=REPORT,COPIES=2
```

This statement requests two copies of the data set defined by DD statement **REPORT**. Any printer with standard forms, train, and carriage tape can be used.

---

```
//*FORMAT PR,DDNAME=,DEST=ANYLOCAL
```

This statement specifies that all data sets without **//\*FORMAT PR** statements are to be printed on any local printer.

## JES3: **//\*FORMAT PU**

### **//\*FORMAT PU Statement**

**Purpose:** Use the **//\*FORMAT PU** statement to specify to JES3 processing instructions for punch data sets. These instructions permit special processing of output data sets, such as:

- Multiple destinations.
- Multiple copies of output with different attributes.

You can code several **//\*FORMAT PU** statements for a data set to specify special requirements for different copies of the data set. You can also code a **//\*FORMAT PR** statement for the same data set, thereby both printing and punching it.

Output classes are established at JES3 initialization. These classes group output data sets with similar characteristics for job output. You should determine if you should use one of these classes or if you should code a **//\*FORMAT PU** statement.

**Note:** The **//\*FORMAT PU** statement applies only to output data sets punched by JES3. The statement is ignored for data sets sent to a TSO userid or processed by an external writer.

**References:** For more information, see "Specifying Data Set Processing Options" on page 7-43.

#### **Syntax:**

```
//*FORMAT PU,DDNAME={,|ddname}[,parameter]...
```

The parameters are:

```
CHNSIZE={DS| (nnn[,mmm])}  
COMPACT=compaction-table-name  
COPIES={nnn|1}
```

```
        {ANYLOCAL           }  
        {device-name       }  
        {device-address    }  
        {group-name        }  
DEST={nodename[.remote]   }  
      {LOCAL                }  
      {( [ ,device-name   ] )}  
      {( (type ,device-address) )}  
      {( [ ,group-name    ] )}  
      {( [ ,LOCAL         ] )}
```

```
EXTWTR=name  
FORMS={STANDARD| form-name}  
INT={YES|NO}
```

**Parameter Definition****PU**

Indicates this statement is associated with a punch data set.

**DDNAME = {,|ddname}**

Indicates a null DDNAME parameter. This null makes the parameters on this **/\*FORMAT PU** statement the defaults for the job. These parameters then apply to all of the job's punch data sets, except those covered by a **/\*FORMAT PU** statement with **DDNAME = ddname**.

**Effect of DDNAME =, Parameter:** A **/\*FORMAT PU** statement that contains **DDNAME =,** is called **nonspecific**; a statement that contains **DDNAME = ddname** is called **specific**.

When you code a nonspecific **/\*FORMAT PU** statement, the parameters you specify become the defaults for the job. These parameters apply to all of the job's punch data sets unless overridden by a specific **/\*FORMAT PU** statement.

Parameters coded on a nonspecific **/\*FORMAT PU** statement are overridden by parameters coded on sysout DD statements or by parameters in the JES3 SYSOUT initialization statement.

**ddname**

Specifies the ddname of the DD statement that defines the data set you want to punch. The ddname can refer to only one DD statement in the job. You must qualify this ddname to identify the job step and, if appropriate, the procedure step, that contains the DD statement. Qualification is in the form **stepname.ddname** or **stepname.procname.ddname**. The ddname must match exactly the ddname on the DD statement. (See the example for the **/\*DATASET** statement.)

**CHNSIZE = {DS|(nnn|mmm)}**

Specifies the number of logical records to be transmitted to a work station as a systems network architecture (SNA) chain and indicates whether normal output checkpoints are to be taken for this data set.

*Note:* This parameter is valid only when transmitting to a SNA work station.

Be careful in selecting subparameters, because each affects performance differently. Sending the data set as a SNA chain provides the best performance, but can cause duplicate data to be written to the output device if an operator intervention is required. The remote operator can eliminate duplicate data by using existing commands to reposition and restart the output writers.

When an end-of-chain indicator is sent in the data set, JES3 takes an output checkpoint. You can provide additional checkpoints for critical data by sending an end-of-chain indicator. For example, when punching bank checks, you can have an output checkpoint taken for each check by specifying each check as a SNA chain.

## JES3: **//\*FORMAT PU**

### **DS**

Indicates that the data set is to be sent as a single SNA chain and that JES3 is to take normal output checkpoints. DS is the default.

### **nnn**

Specifies the SNA chain size in pages. nnn is a decimal number from 1 through 255. The size of a page is indicated by the value you assign to mmm.

### **mmm**

Specifies the number of logical records in a page. mmm is a decimal number from 1 through 255.

### **COMPACT = compaction-table-name**

Specifies the compaction table for JES3 to use when sending a systems network architecture (SNA) data set to a SNA remote terminal. The compaction-table-name is a symbolic name defined by the installation during JES3 initialization. The name is 1 to 8 alphanumeric characters.

JES3 performs compaction using an installation-defined default table and writes a message to the console when:

- You do not specify a compaction table.
- You specify an invalid compaction table.
- You specify a compaction table that JES3 cannot find.

If the installation has not defined a default table, JES3 sends the data without compacting it.

If the remote punch does not support compaction, JES3 ignores the COMPACT parameter and sends the data without compacting it.

### **COPIES = {nnn|1}**

Indicates how many copies of the data set are to be punched. nnn is a number from 0 through 255. If you code COPIES=0, JES3 does not punch this data set. If a COPIES parameter is not specified, the default is 1.

### **DEST = destination**

Routes the output to a punch. This parameter overrides the **//\*MAIN** statement **ORG** parameter.

If you omit **DEST**, JES3 assigns the first available punch that is in the origin group and that fulfills all processing requirements. The origin group is the group of punches defined for the local or remote submitting location. If the job originated at a remote job processing (RJP) terminal, JES3 returns the output to the originating terminal group.

### **ANYLOCAL**

Indicates any local device (a punch being used for the output class specified on the **DD** statement) attached to the global processor.

### **device-name**

Specifies a local device by a symbolic name defined by the installation during JES3 initialization. device-name is 1 to 8 alphanumeric or national characters.

**device-address**

Specifies the 3-character physical device address.

**group-name**

Specifies a group of local devices, an individual remote station, or a group of remote stations by a symbolic name defined by the installation during JES3 initialization. group-name is 1 to 8 alphanumeric or national characters.

**nodename**

Specifies a node by a symbolic name defined by the installation during JES3 initialization. nodename is 1 to 8 alphanumeric or national characters.

**remote**

Specifies a remote work station or VM userid to which the receiving node directs output. remote is 1 to 8 characters.

**LOCAL**

Specifies any local device that is attached to the global processor and that does not belong to a group.

*Note:* When you code `DEST=LOCAL`, your installation must have at least one local device that is not assigned to a group.

**(type)**

Specifies a device classification. type is in the form **(gggssss)** where **ggg** is the general device classification and **ssss** is the specific device classification. The type must be enclosed in parentheses. The type must be defined by the installation during JES3 initialization. For example, type for a 3525 is **(PUN3525)**.

**EXTWTR = name**

Identifies the external writer that is to process the data set at the destination node. name is 1 to 8 alphanumeric characters and must identify a module defined in the remote JES3 node that is to execute the job.

**FORMS = {STANDARD|form-name}**

Specifies the forms on which the output data set is to be punched.

**STANDARD**

Indicates the standard form. JES3 uses the standard form specified at JES3 initialization.

**form-name**

Specifies the punch forms. form-name is 1 to 8 alphanumeric characters.

**INT = {YES|NO}**

Specifies whether or not the output is to be interpreted. If an INT parameter is not coded, the default is NO.

**YES**

Requests that JES3 try to punch the output data set on a 3525 Card Punch (PUN3525I) with a Multiline Card Print feature.

*Note:* If the DEST parameter does not send output to a 3525I, JES3 ignores `INT=YES`, if specified.

## JES3: **//\*FORMAT PU**

**NO**

Requests that the cards not be interpreted.

### **Relationship to Sysout DD and OUTPUT JCL Statements**

- JES3 ignores the processing options specified on a default **//\*FORMAT** statement when a sysout DD statement explicitly or implicitly references an **OUTPUT JCL** statement.
- JES3 ignores the processing options specified on a default **OUTPUT JCL** statement when a **//\*FORMAT** statement explicitly references a sysout DD statement.
- When a sysout DD statement explicitly references an **OUTPUT JCL** statement and a **//\*FORMAT** statement explicitly references the same DD statement, the processing options from both the **OUTPUT JCL** and **//\*FORMAT** statements apply. Two separate sets of output are created from the data set defined by the sysout DD statement; one according to the processing options on the **OUTPUT JCL** statement, and the other according to the processing options on the **//\*FORMAT** statement.

### **Relationship to **//\*PROCESS** Statement**

JES3 accumulates **//\*FORMAT PU** statements within a job and applies them to any **JES3 **//\*PROCESS**** statement that is normally affected by a **//\*FORMAT PU** statement.

### **Example of the **//\*FORMAT PU** Statement**

```
//*FORMAT PU,DDNAME=PUNCHOUT,DEST=PU1,FORMS=RED-STRP
```

This statement requests that one copy of the data set defined by DD statement **PUNCHOUT** be punched on device **PU1**. Before processing, the operator is requested to insert **RED-STRP** cards into the punch.

## /\*MAIN Statement

**Purpose:** Use the /\*MAIN statement to define the processor requirements for the current job. Many of the parameters are used to override parameters on the JES3 STANDARDS initialization statement.

**Note:** If any parameter is misspelled or contains an invalid value, JES3 writes the following to the JESMSG data set: the /\*MAIN statement, the relative error position on the statement, and an error message. Then JES3 abnormally terminates the job.

### Syntax:

```

/*MAIN parameter[,parameter]...

The parameters are:

ACMAIN=processor-id
BYTES=( [nnnnnn] [[,WARNING|,W]]
        (          [[,CANCEL|,C] ]
        (          [[,DUMP|,D]   ] )
CARDS=( [nnn] [[,WARNING|,W]]
        (      [[,CANCEL|,C] ]
        (      [[,DUMP|,D]   ] )
CLASS=class-name
DEADLINE=(time,type[,date      ]
          (          [,rel,cycle])
EXPDTCHK={YES|NO}
FAILURE={RESTART|CANCEL|HOLD|PRINT}
FETCH={ALL|NONE|SETUP|[/](ddname[,ddname]...)}
HOLD={YES|NO}
IORATE={MED|HIGH|LOW}
JOURNAL={YES|NO}
LINES=( [nnn] [[,WARNING|,W]]
        (      [[,CANCEL|,C] ]
        (      [[,DUMP|,D]   ] )
LREGION=nnnnK
MSS={JOB|HWS}
ORG={group-name|nodename[.remote]}
PAGES=( [nnnnnnnn] [[,WARNING|,W]]
        (          [[,CANCEL|,C] ]
        (          [[,DUMP|,D]   ] )
PROC={ST|xx}
RINGCHK={YES|NO}
SETUP={JOB|HWS|THWS|[/](ddname[,ddname]...)}
SPART=partition-name
SYSTEM={ANY|JGLOBAL|JLOCAL|[/](main-name[,main-name]...)}
TRKGRPS=(primary-quantity,secondary-quantity)
TYPE={ANY|VS2}
UPDATE=(dsn[,dsn]...)
USER=userid

```

## Parameter Definition

### ACMAIN = processor-id

Identifies the job with the specified processor, even though the job was not submitted from or executed on that processor. This parameter has no effect if it specifies the processor that the job is running on. ACMAIN allows:

- Sysout data sets to be sent to the specified processor. See the USER parameter. When used for sysout data sets, the ACMAIN parameter applies to all sysout output for the job.
- A job termination message to be sent to the specified processor if the JOB statement contains a NOTIFY parameter.

### processor-id

Identifies a processor in the complex.

### BYTES = ([nnnnnn][[WARNING|W][[CANCEL|C][[DUMP|D]])

Specifies the maximum number of bytes of data to be spooled for this job, and the action to be taken if the maximum is exceeded.

If BYTES is not specified, the installation default for this job class applies.

### nnnnnn

Specifies the number of bytes in thousands. nnnnnn is 6 decimal numbers from 1 through 999999.

### WARNING|W

If the maximum is exceeded, requests that JES3 issue an operator warning message and continue processing.

Any messages about this parameter following the warning message will reflect the number specified on the STANDARD initialization statement or the system default, **not** the specified maximum.

### CANCEL|C

If the maximum is exceeded, requests that JES3 cancel the job.

### DUMP|D

If the maximum is exceeded, requests that JES3 cancel the job and ask for a storage dump.

### CARDS = ([nnn][[WARNING|W][[CANCEL|C][[DUMP|D]])

Specifies the maximum number of cards to be punched for this job, and the action to be taken if the maximum is exceeded.

If you specify CARDS=0 the zero applies only to the quantity of punched output; it does **not** cancel the action to be taken if the maximum is exceeded. If a record is then sent to a punch, JES3 will warn, cancel, or dump, depending on the second parameter.

*Note:* When punching dump output, JES3 ignores CARDS=0.

If CARDS is not specified, the installation default for this job class is used.



**nnn**

Specifies the number of cards in hundreds. nnn is 3 decimal numbers from 1 through 999.

**WARNING|W**

If the maximum is exceeded, requests that JES3 issue an operator warning message and continue processing.

Any subsequent messages about this parameter will reflect the number specified on the STANDARD initialization statement or the system default, **not** the maximum specified in the CARDS parameter.

**CANCEL|C**

If the maximum is exceeded, requests that JES3 cancel the job.

**DUMP|D**

If the maximum is exceeded, requests that JES3 cancel the job and ask for a storage dump.

**CLASS = class-name**

Specifies the job class for this job. class-name can be 1 to 8 characters.

This CLASS parameter overrides the JOB statement CLASS parameter; if you omit this CLASS parameter, JES3 uses the job class specified in the JOB statement CLASS parameter. If the class-name is a single-character, you may instead use the JOB statement CLASS parameter.

If neither CLASS nor LREGION is specified, JES3 determines the logical region size based on initialization parameters.

**DEADLINE = (time,type[,date],rel,cycle)**

Specifies when the job is required.

If you specify the current date but submit the job after the specified time, JES3 changes the priorities to make the job the same priority level it would have been had it been submitted before the deadline but not completed.

Potentially, deadline scheduling can interfere with dumping a portion of the job queue. For example, if JOB A is waiting to be scheduled, has a priority of 7, and, in one minute, is due to have its priority increased to 9, JOB A could be missed by dump job processing, if the dump job facility is dumping the entire job queue and currently dumping priority 8 jobs. The dump job facility processes the jobs with the highest priority first. If the dump job facility does not finish processing priority 8 jobs before JOB A becomes priority 9, JOB A will not be dumped.

Deadline scheduling information is not sent with a job when the job is transferred via NJE to another node; the destination node may use different deadline scheduling algorithms, if any.

**time**

Specifies the deadline time, expressed as one of the following:

**nM**

The job is to be scheduled within n minutes. n is 1 to 4 numbers from 0 through 1440.

**nH**

The job is to be scheduled within n hours. n is 1 or 2 numbers from 0 through 24.

**hhhh**

The job is to be scheduled by the time of day, hhhh, in 24-hour clock time (0800 is 8:00 a.m.). hhhh is 4 numbers from 0000 to 2400.

**type**

Identifies the type of deadline. The type is defined by the installation and is one character: A through Z or 0 through 9. If the type is not defined, JES3 abnormally terminates the job.

**date**

Specifies the date when the time parameter takes effect. date, in the format **mmddy**, is given as the month (01-12), day (01-31), and year (00-99).

If both date and rel,cycle are omitted, JES3 assumes (1) the current date, if the deadline time is later in the day, or (2) the next day's date, if the deadline time has already past today.

**rel**

Specifies on which day within the cycle the deadline falls. rel is 1 to 3 numbers from 1 through 366. The value of rel depends on the specified cycle, as follows:

- Values coded with WEEKLY default to 7 if greater than 7. Sunday is day 1; Saturday is day 7.
- Values coded with MONTHLY default to 31 if greater than 31. The values 29, 30, and 31 are treated as the last day of the month.
- Values coded with YEARLY default to 365, for all non-leap years, if greater than 365. Leap year defaults to 366.

**cycle**

Specifies periodic runs. cycle is coded as WEEKLY, MONTHLY, or YEARLY.

For example, DEADLINE=(1200,B,1,WEEKLY) indicates that the job reaches its deadline at 12 noon every Sunday.

**EXPDTCHK = {YES|NO}**

Specifies whether or not JES3 is to perform expiration date checking for scratch IBM standard label (SL) output tape volumes.

**YES**

Requests expiration date checking. Tape volumes premounted for SL scratch requests must have expired dates.

**NO**

Requests that expiration dates not be checked.

**FAILURE = {RESTART|CANCEL|HOLD|PRINT}**

Specifies the job recovery option to be used if the system fails.

**RESTART**

Requests that JES3 restart the job when the failing processor is restarted. Do not specify RESTART for jobs that use the DEQ at DEMOUNT facility for tape volumes.

**CANCEL**

Requests that JES3 cancel the job for printing.

**HOLD**

Requests that JES3 hold the job for restart.

**PRINT**

Requests that JES3 print the job and then hold the job for restart.

**FETCH = {ALL|NONE|SETUP|/|(ddname[,ddname]...)}**

Determines the fetch messages that will be issued to the operator for disk and tape volumes for this job.

If FETCH is not specified, the installation default for this job class applies.

**ALL**

Requests that JES3 issue fetch messages to the operator for all removable volumes specified in DD statements that request JES3-setup devices. This subparameter does not apply to permanently resident volumes.

**NONE**

Requests that JES3 not issue fetch messages.

**SETUP**

Requests that JES3 issue fetch messages to the operator for the volumes specified in all DD statements identified in the /\*MAIN SETUP parameter. If you code FETCH = SETUP without also coding the /\*MAIN SETUP parameter, JES3 will issue fetch message as though you had specified FETCH = ALL.

**ddname**

Requests that JES3 issue fetch messages for only the volumes specified in DD statement ddname.

If you code a list of ddnames and the list cannot be contained on a single statement, FETCH = must be repeated on the continuation statement.

## JES3: //\*MAIN

/

Coding /ddname requests that JES3 **not** issue fetch messages for any volumes specified in DD statement ddname.

**HOLD = {YES|NO}**

### YES

Indicates that the job is to enter the system in operator-hold status and be withheld from processing until the operator requests its release. However, if an error occurs during input service processing, the job is not held for operator intervention but is scheduled for converter-interpreter processing.

This parameter has the same function as TYPRUN = HOLD on the JOB statement.

### NO

Indicates that the job is to enter the system normally. Processing does not require operator intervention.

**IORATE = {MED|HIGH|LOW}**

Indicates the I/O-to-processor ratio for a job. Use this parameter to balance the mixture of jobs selected for execution on the processor.

If IORATE is not specified, the installation default for this job class applies.

**JOURNAL = {YES|NO}**

Indicates whether or not JES3 is to create a job journal for the job.

If JOURNAL is not specified, JES3 uses an installation default specified at initialization.

### YES

Indicates that the job is to have a job journal.

### NO

Indicates that the job is not to have a job journal.

**LINES = ((nnn)[,WARNING|,W][,CANCEL|,C][,DUMP|,D])**

Indicates the maximum number of lines of data to be printed for this job, and the action to be taken if the maximum is exceeded.

If you specify LINES = 0 the zero applies only to the number of lines; it does **not** cancel the action to be taken if the maximum is exceeded. If a record is sent to be printed, JES3 will warn, cancel, or dump, depending on the second parameter.

*Note:* JES3 ignores any line count specification when printing the output for a SYSABEND or SYSUDUMP sysout data set.

If LINES is not specified, the installation default for this job class applies.

### nnn

Specifies the number of lines, in thousands. nnn is 3 decimal numbers from 1 through 999.

**WARNING|W**

If the maximum is exceeded, requests that JES3 issue an operator warning and continue processing.

Any messages about this parameter following the warning message will reflect the number specified on the **STANDARD** initialization statement or the system default, **not** the maximum specified in the **LINES** parameter.

**CANCEL|C**

If the maximum is exceeded, requests that JES3 cancel the job.

**DUMP|D**

If the maximum is exceeded, requests that JES3 cancel the job and ask for a storage dump.

**LREGION = nnnnK**

Specifies the approximate size of the largest step's working set in real storage during execution. **LREGION** (logical region) is used by JES3 to improve scheduling on the processor.

If neither **CLASS** nor **LREGION** is coded, JES3 determines the logical region size based on initialization parameters.

Consult your system programmer when specifying an **LREGION** parameter. If the values selected for **LREGION** are too small, the job may take longer to run.

**MSS = {JOB|HWS}**

Requests mass storage system (**MSS**) support and indicates how the **MSS** devices should be allocated. This parameter overrides the installation default defined at JES3 initialization.

**JOB**

Specifies that each request for 3330V Disk Storage is to be assigned to a separate device.

**HWS**

Specifies that 3330V devices are to be reused in subsequent job steps in order to minimize the number of devices needed for the job.

*Note:* The **JOB** and **HWS** options are independent across device types. For example, **HWS** can be specified in the **SETUP** parameter for non-**MSS** devices while **JOB** is specified in the **MSS** parameter for **MSS** devices, or vice versa.

**ORG = {group-name|nodename|.remote}}**

Indicates that the job's output is to be directed to the specified origin group or networking node.

Normally, output from a job is directed to the group of devices or node from which it originated. This parameter is used to override the origin group name or nodename of the device actually used to enter the job into the JES3 system.

## JES3: /\*MAIN

### **group-name**

Specifies an origin group.

### **nodename**

Specifies a network node. nodename is 1 to 8 characters.

### **.remote**

Specifies a remote work station or VM userid. remote is 1 to 8 characters and must be separated from the nodename by a period.

**Overriding an ORG Parameter:** If you do not want a particular data set in the job to go to the destination on the ORG parameter, change its destination in one of the following ways:

- If the data set is not scheduled to a **held** class, override the destination on the ORG parameter by directly referencing the DD statement that defines the output data set in a /\*FORMAT statement. Specify the desired destination in the /\*FORMAT statement DEST parameter.
- If the data set is a sysout data set, including one scheduled to a **held** class, override the ORG parameter by explicitly or implicitly referencing an OUTPUT JCL statement on the sysout DD statement. Specify the desired destination in the OUTPUT JCL statement DEST parameter.

**PAGES = (nnnnnnnn|[[,WARNING|,W|[[,CANCEL|,C|[[,DUMP|,D]])**

Indicates the maximum number of pages to be printed for this job, and the action to be taken if the maximum is exceeded.

If PAGES is not specified, the installation default for this job class applies.

### **nnnnnnnn**

Specifies the number of pages. nnnnnnnn is 8 decimal numbers from 1 through 16777215.

### **WARNING|W**

If the maximum is exceeded, requests that JES3 issue an operator warning message and continue processing.

Any messages about this parameter following the warning message will reflect the number specified on the STANDARD initialization statement or the system default value, **not** the maximum specified in the PAGES parameter.

### **CANCEL|C**

If the maximum is exceeded, requests that JES3 cancel the job.

### **DUMP|D**

If the maximum is exceeded, requests that JES3 cancel the job and ask for a storage dump.

**PROC = {ST|xx}**

Names the procedure library that the system is to search for cataloged procedures specified in EXEC statements in the job. If a procedure cannot be found in the named library, JES3 abnormally terminates the job.

If this parameter is omitted, the default depends on the source of the job. If the job is submitted as a batch job, the default is ST. If the job is submitted from an internal reader, the default may be another procedure library, as specified by the installation.

**ST**

Indicates the standard default procedure library.

**xx**

Identifies the last 2 characters of the ddname of a procedure library. xx is defined by the installation. If this parameter is coded, only the specified library is searched; the standard procedure library is not searched.

**RINGCHK = {YES|NO}**

Indicates whether or not JES3 is to check the status of the tape reel ring for tape devices set up by JES3.

**YES**

Indicates that a validation check is to be made.

**NO**

Indicates that ring checking is to be by-passed. for this job.

**SETUP = {JOB|HWS|THWS|/|/(ddname[,ddname]...)}**

Modifies the standard setup algorithm used in assigning devices to a job before its execution.

If SETUP is omitted, the device requirements for mountable tape and disk volumes are based on an installation default defined at initialization.

**JOB**

Requests job setup, which is allocation of all JES3-managed devices required in the job before the job executes. JES3 mounts the initial volumes necessary to run all steps before the job executes. JOB overrides the SETUP parameter on the JES3 STANDARDS initialization statement.

**HWS**

Requests high watermark setup, which is allocation of the minimum number of devices required to run the job. The minimum number is equal to the greatest number of devices of each type needed for any one job step. High watermark setup does not cause premounting of all mountable volumes.

**THWS**

Requests high watermark setup for tapes but job setup for disks.

**DHWS**

Requests high watermark setup for disks but job setup for tapes.

**ddname**

Specifies explicit setup, which is allocation of the volumes needed for DD statement ddname before the job executes. JES3 premounts the indicated volumes. When requesting explicit setup, specify enough devices so that JES3 can allocate all the required devices at any one time. If too few devices are specified, JES3 cancels the job.

## JES3: /\*MAIN

You must qualify this ddname to identify the job step and, if appropriate, the procedure step, that contains the DD statement. Qualification is in the form **stepname.procstepname.ddname**. The ddname must match exactly the ddname on the DD statement. (See the example for the /\*DATASET statement.)

If you code a list of ddnames and the list cannot be contained on a single statement, SETUP = must be repeated on the continuation statement.

/  
Coding /ddname specifies that JES3 is **not** to set up explicitly any volumes specified in DD statement ddname.

### **SPART = partition-name**

Indicates the spool partition in which JES3 is to allocate spool space to this job.

#### **partition-name**

Specifies the name of the spool partition. **partition-name** is 1 to 8 characters and must match the partition name specified in the NAME parameter of a JES3 SPART initialization statement. If the name does not match, JES3 ignores the SPART parameter and uses the installation default.

The SPART parameter overrides all other spool partition definitions for this job, except the SPART parameter on the JES3 SYSOUT initialization statement for output data sets. If SPART is not specified, JES3 allocates spool data sets to a specific spool partition according to the following priorities, in override order:

1. The spool partition associated with the job's output class and defined by a JES3 SYSOUT initialization statement.
2. The spool partition associated with the job's class and defined by a JES3 CLASS initialization statement.
3. The spool partition associated with the executing processor and defined by a JES3 MAINPROC initialization statement.
4. The default spool partition.

### **SYSTEM = {ANY|JGLOBAL|JLOCAL|[/(main-name[,main-name]...)]}**

Indicates the processor that is to execute this job. If a specific processor is named, the processor name must also be specified on the CLASS initialization statement for the job class.

#### **ANY**

Indicates any global or local system that satisfies the job's requirements.

#### **JGLOBAL**

Indicates that the job is to run on the global processor only.

#### **JLOCAL**

Indicates that the job is to run on a local processor only.

#### **main-name**

Indicates that the job is to run on one of the specific processors named.



/  
Coding /main-name specifies that the job is not to run on the specific processor named.

If you omit a **SYSTEM** parameter, the job runs on the processor used for the job's class. Therefore, a **SYSTEM** parameter is usually unneeded. Check with your system programmer.

The control program specified in the **TYPE** parameter must be running on the processor specified in the **SYSTEM** parameter, or JES3 abnormally terminates the job.

If you are using different levels of MVS in your complex, code the **SYSTEM** parameter to request a system that can interpret and execute the level of JCL you are using. For information on how to do this, see *SPL: JES3 Initialization and Tuning*.

The **SYSTEM** parameter must be consistent with any JES3 **//\*PROCESS** statements.

**TRKGRPS = (primary-quantity,secondary-quantity)**

Specifies the number of track groups to be assigned to the job. A track group is a number of spool space allocation units. The size of the track group is defined in the **GRPSZ** parameter on the JES3 **BUFFER** or **SPART** initialization statement.

**primary-quantity**

Specifies the number of track groups to be initially allocated. This quantity is one decimal number from 1 to 9.

**secondary-quantity**

Specifies the number of track groups to be allocated when the currently allocated groups are filled and more space is needed. This quantity is one decimal number from 1 to 9.

The **//\*MAIN TRKGRPS** parameter overrides a **TRKGRPS** parameter on the **CLASS** or **MAINPROC** initialization statement. However, when a sysout **DD** statement specifies an output class, the **TRKGRPS** parameter for that output class overrides the **//\*MAIN TRKGRPS** parameter.

**TYPE = {ANY|VS2}**

Indicates the control program that is to execute this job. If you omit a **TYPE** parameter, the job runs under the control program used for the job's class.

**ANY**

Indicates that JES3 is to use any control program that satisfies the job's requirements. In present systems, JES3 schedules the job on MVS.

**VS2**

Indicates that JES3 is to schedule the job on MVS.

**UPDATE = (dsn[,dsn]...)**

Identifies the procedure library data set(s) this job is to update. This parameter causes all jobs using this data set and any concatenated data sets to be held until the update is complete.

## JES3: /\*MAIN

### **dsn**

Specifies the data set name.

### **USER = userid**

Identifies the job with the specified TSO user, even though the job was not submitted via TSO by that user. USER allows:

- Sysout data sets to be sent to a processor for use by the TSO user. See the ACMAIN parameter.
- The TSO user interacting with a global or local processor to issue the TSO OUTPUT command to access sysout data sets.
- The TSO user interacting with any processor to inquire about the status of the job or to cancel the job.
- The TSO user to control a job submitted via an input source other than the internal reader, provided the installation does not force job naming conventions.

### **userid**

Identifies a TSO userid. userid is 1 to 7 alphanumeric or national characters.

Enclose the userid in apostrophes when it contains special characters or begins with a number.

## Location in the JCL

When you specify **ORG** on a /\*MAIN statement, the /\*MAIN statement should precede all /\*FORMAT statements that do not contain a DEST parameter. If it does not, JES3 uses the default destination for the /\*FORMAT statements; their output is sent to the node where the job entered the system.

When specifying **ORG** on a /\*MAIN statement that is part of a remote job, place the /\*MAIN statement immediately after the second JOB statement.

## Example of the /\*MAIN Statement

```
/*MAIN SYSTEM=SY1,LINES=(5,C),SETUP=HWS,  
/*FAILURE=RESTART,DEADLINE=(0800,A,3,WEEKLY)
```

The job executes on processor SY1. It is estimated to produce not more than 5000 lines of printed output; if the output exceeds 5000 lines, JES3 is to cancel the job. HWS specifies high watermark setup, so JES3 is to allocate the minimum number of devices required for this job. If the system fails, JES3 is to restart the job on the processor SY1. JES3 is to complete this job by 8 a.m. every Tuesday (Tuesday is day number 3) by adjusting the job's scheduling priority using the installation-defined A-type deadline scheduling parameters.

---

```
/*MAIN ACMAIN=2,USER='58BIRD'
```

If this statement appears in a job entered from any TSO id on any processor in the complex, then the job's sysout data sets would go to TSO id 58BIRD on processor 2.

## **//\*NET Statement**

**Purpose:** Use the **//\*NET** statement to define the dependencies between jobs in a dependent job control (DJC) network. JES3 sets up a net of dependent jobs and executes them in a specific order.

**References:** For more information, see “Dependent Job Control for JES3: The Job Net” on page 3-27.

### **Syntax:**

```
//*NET {NETID|ID}=name[,parameter]...
```

The parameters are:

```
{ABCMP|AC}={NOKP|KEEP}
{ABNORMAL|AB}={D|F|R}
{NORMAL|NC}={D|F|R}
DEVPOOL=( {ANY|NET} [, dev-name,n]... [,SDGxx]... )
DEVRELEASE={YES|NO}
{NETREL|NR}=(netid,jobname)
{NHOLD|HC}=n
{NRCMP|PC}={HOLD|NOHO|FLSH}
{OPHOLD|OH}={NO|YES}
{RELEASE|RL}=(jobname[,jobname]...)
{RELSCHCT|RS}=n
```

- Only one **//\*NET** statement can be coded for each job in a DJC network.
- The **//\*NET** statement must precede any JES3 **//\*PROCESS** statements.

## **Parameter Definition**

### **NETID = name**

Specifies the name of the DJC network for this job. name is 1 to 8 characters; the first character must be alphabetic.

All jobs put into the system with the same NETID name form a DJC network. To add a job to an existing DJC network, specify the NETID name for that job.

### **ABCMP = {NOKP|KEEP}**

Indicates what action JES3 is to take if the job abnormally terminates.

#### **KEEP**

Indicates that the net is to be kept in the system until (1) the job is resubmitted and completes normally or (2) the operator forces the net from the system. Use KEEP to make sure that the net is not purged until the operator takes proper action.

#### **NOKP**

Indicates that JES3 is to purge the net if the job that abnormally terminated has not been resubmitted by the time the other jobs in the net have completed. JES3 purges the net unless successors or subnets are missing.

## JES3: /\*NET

*Note:* If a job abnormally terminates, you can resubmit it to the net, and the net will be retained until the job completes.

**ABNORMAL = {D|F|R}**

**NORMAL = {D|F|R}**

Indicates the action JES3 is to take for this job when any predecessor job completes execution normally or abnormally.

### D

Requests that JES3 decrease this job's NHOLD count, which indicates the number of predecessors for this job. When the NHOLD count goes to zero, JES3 can schedule this job.

### F

Requests that JES3 flush this job and its successors from the system. JES3 cancels the job, prints any output, and cancels all successors presently in the system, regardless of their normal or abnormal specifications. However, JES3 enters into the system all successor jobs that enter after the net has been flushed. To flush these jobs, the operator must cancel the jobs or the net.

### R

Requests that JES3 retain this job in the system and not decrease the NHOLD count. R suspends the job and its successors from scheduling until either the predecessor job is resubmitted or the operator decreases the NHOLD count.

**DEVPOOL = ({ANY|NET},[dev-name,n]...[,SDGxx]...)**

Specifies devices to be dedicated to this DJC net.

The DEVPOOL parameter is recognized only in the first job of a DJC network to enter the system.

### ANY

Indicates that jobs in the net can use any dedicated or undedicated device. JES3 tries to allocate from the dedicated pool before allocating any undedicated devices.

### NET

Indicates that jobs can use only devices dedicated to the net.

### dev-name,n

Specifies the name and number of a dedicated device. You can specify as many device-names with numbers as will fit on one statement.

#### dev-name

specifies (1) a device name defined to JES3 by the installation during initialization or (2) a device-type as specified in the UNIT parameter of an IODEVICE system generation macro instruction. See *SPL: System Generation Reference* for a list of IBM device types.

#### n

Specifies the number of named devices. n is a number from 1 through 32767.

**SDGxx**

Specifies the mass storage system (MSS) staging drive group(s) to be pooled (fenced) for the DJC network. xx is the staging drive group number.

**DEVRELSE = {YES|NO}**

Indicates when devices dedicated to the net are to be released.

The DEVRELSE parameter can be coded in several jobs in the network. The first completing job that specified DEVRELSE = YES causes the devices dedicated to the network to be released.

**YES**

Requests that JES3 release all devices at the end of this job.

**NO**

Requests that JES3 release all devices only when the last job in the net ends.

**NETREL = (netid,jobname)**

Indicates that this job must be executed before the named job in another DJC network can be executed.

The NETREL parameter can be specified only once for each job of a DJC network.

**netid**

Identifies the NETID for the successor job.

**jobname**

Identifies the name on the JOB statement for the successor job.

**NHOLD = n**

Indicates the number of immediate predecessor job completions required before this job can be released for scheduling. n is a number from 0 through 32767.

If you specify NHOLD = 0 or omit the NHOLD parameter, this job has no predecessor jobs. JES3 can schedule it for immediate execution.

If the NHOLD count is incorrect, the following can occur:

- If n is greater than the actual number of predecessor jobs, JES3 does not release this job for execution when all of its predecessor jobs complete execution.
- If n is less than the actual number of predecessor jobs, JES3 prematurely releases the job for execution.

**NRCMP = {HOLD|NOHO|FLSH}**

Indicates that a network job that completed normally is being resubmitted and that JES3 must erase all references to the job before the job reenters the network.

## JES3: /\*NET

### **HOLD**

Indicates that JES3 is to put the job in operator hold status.

### **NOHO**

Indicates that JES3 is to allow the job to be scheduled as system resources become available.

### **FLSH**

Indicates that JES3 is to flush the job.

**OPHOLD = {NO|YES}**

### **NO**

Indicates that the job is to be processed normally without operator intervention.

### **YES**

Indicates that the job is placed in DJC operator hold. This subparameter prevents JES3 from scheduling this job until the operator explicitly releases it from DJC operator hold.

**RELEASE = (jobname[,jobname]...)**

Indicates that this job must be executed before the named the job(s) in this DJC network can be executed.

### **jobname**

Identifies the name on the JOB statement for a successor job. You can specify from 1 to 50 successor jobnames.

RELEASE is the only parameter on the /\*NET statement that can be split and continued on the next statement.

**RELSCHCT = n**

Controls early set up of a dependent job's resources. Set up begins when the NHOLD count becomes less than or equal to n. n is a number from 1 through 32767.

If you specify RELSCHCT=0 or omit the RELSCHCT parameter, JES3 does not set up dependent jobs early.

Do not specify the RELSCHCT parameter:

- For a job that may have catalog dependencies in dependent job control.
- For nonstandard DJC jobs. Nonstandard jobs contain one or more /\*PROCESS statements.

**Examples of the `//*NET` Statement**

```
//*NET NETID=NET01,NHOLD=0,DEVPOOL=(,3330,2)
```

This statement defines a DJC network named NET01. The net contains no predecessor jobs. The DEVPOOL parameter, which must be coded in the first job in the network, requests that JES3 establish a device pool of two 3330s for network NET01.

---

```
//*NET NETID=N1,RL=B,NR=(N2,B2),DEVPOOL=(NET,3330,1)
```

This statement defines a DJC network named N1. This job must be executed before job B, which is in N1, and before job B2, which is in the DJC network named N2. The DEVPOOL parameter specifies that one 3330 is to be dedicated to network N1; jobs in the network must wait for this device if it is in use.

JES

## **//\*NETACCT Statement**

**Purpose:** Use the **//\*NETACCT** statement to specify accounting information that JES3 is to transmit with a job to another node in the network.

**Syntax:**

```
//*NETACCT parameter[,parameter]...
```

The parameters are:

```
PNAME=programmer's-name  
ACCT=nnnnnnnn  
BLDG=nnnnnnnn  
DEPT=nnnnnnnn  
ROOM=nnnnnnnn  
USERID=nnnnnnnn
```

- You must code at least one parameter on the **//\*NETACCT** statement.
- The **//\*NETACCT** statement must immediately follow the first **JOB** statement and precede any **//\*ROUTE XEQ** statements.
- You cannot continue a **//\*NETACCT** statement. If the parameters occupy too much space, code several **//\*NETACCT** statements.
- Enclose any character string that contains embedded blanks or special characters in apostrophes.

### **Parameter Definition**

**PNAME = programmer's-name**

Specifies the programmer's name. The name is 1 to 20 characters.

**ACCT = nnnnnnnn**

Specifies the network account number. nnnnnnnn is 1 to 8 characters.

**BLDG = nnnnnnnn**

Specifies the building number in the network. nnnnnnnn is 1 to 8 characters.

**DEPT = nnnnnnnn**

Specifies the department number in the network. nnnnnnnn is 1 to 8 characters.

**ROOM = nnnnnnnn**

Specifies the room number in the network. nnnnnnnn is 1 to 8 characters.

**USERID = nnnnnnnn**

Specifies the user in the network. nnnnnnnn is 1 to 8 characters.



## **Defaults**

If no **//\*NETACCT** parameter is specified, JES3 uses installation defaults specified at JES3 initialization.

## **Example of the **//\*NETACCT** Statement**

```
//*NETACCT      PNAME=COLLINS,ACCT=D588706,USERID=NXT
```

## JES3: **//\*OPERATOR**

### **//\*OPERATOR Statement**

**Purpose:** Use the **//\*OPERATOR** statement to issue a message to the operator. Columns 1 through 80 are written on the operator console and in the job's hard-copy log when the job enters the JES3 queue.

**Syntax:**

```
//*OPERATOR text
```

### **Example of the **//\*OPERATOR Statement****

```
//*OPERATOR CALL EXT. 641 WHEN THIS JOB STARTS
```

## **//\*PAUSE Statement**

**Purpose:** Use the **//\*PAUSE** statement to halt an input reader temporarily. When you enter a **//\*PAUSE** statement through an input reader, JES3 issues a message and waits for the operator to reply. The system operator must issue a **\*START** command to start the job or a remote work station with console level 15 must send a start message.

The **//\*PAUSE** statement is recognized only when it appears before the first **JOB** statement in an input stream.

The **//\*PAUSE** statement is intended primarily for system checkout and test. It should be issued only by remote work stations.

**Syntax:**

<pre>//*PAUSE  comments</pre>
At least two blanks must follow the word PAUSE before comments.

### **Example of the **//\*PAUSE** Statement**

```
//*PAUSE  THIS IS A TEST.
```

## JES3: **//\*PROCESS**

### **//\*PROCESS Statement**

**Purpose:** Use the **//\*PROCESS** statement to control how JES3 processes a job. A job that contains **//\*PROCESS** statements receives only the JES3 processing specified on the **//\*PROCESS** statements plus certain required processing.

The **//\*PROCESS** statement is useful in testing JCL and programs. For instance, a **//\*PROCESS** statement can make JES3 bypass program execution so that the job's JCL can be checked. Another **//\*PROCESS** statement can make JES3 bypass output processing; then the operator can check by inquiry command whether the job reached execution.

Specifically, the **//\*PROCESS** statement calls a dynamic support program (DSP) in the DSP dictionary. JES3 must be capable of processing the specified DSP.

#### **Syntax:**

```
//*PROCESS dsp  
[parameter[,parameter]...]
```

If the specified DSP requires parameters, they must be listed on the next statement, starting in column 1, ending in column 72, and separated by commas. Parameters must not extend into columns 73 through 80, because these columns are used for sequence numbers in jobs submitted through TSO.

### **Parameter Definition**

#### **dsp**

Specifies the name of the DSP that JES3 is to process. Figure 17-1 lists the valid DSP names and whether parameters can follow.

*Note:* For nonstandard dependent job control (DJC) jobs, a **//\*PROCESS DJC** statement is required only when a **//\*PROCESS MAIN** statement is not included in the job stream.

DSP	DSP Function	Parameters
Standard processing:		
CI	JES3 Converter/Interpreter Service, which interprets the JCL and creates control blocks.	Yes (note 1)
MAIN	Main Service, which processes the program.	No
OUTSERV	Output Service, which processes the job's output.	No
PURGE	Purge service, which purges the job. This is the last function in any job. JES3 automatically creates this DSP.	No
Other functions:		
CBPRNT	Control Block Print	Yes (note 1)
DISPDJC	Display Dependent Job Control	Yes (note 1)
DISPLAY	Display Job Queues	Yes (note 1)
DJC	Invoke Dependent Job Control Updating	No
DR	Disk Reader	Yes (note 2)
ISDRVR	Input Service Driver (JES3 Control Statement Processing)	Yes (Qualified ddname of input file)
JESNEWS	Use JESNEWS Facility	Yes (note 2)
xxx	User-written DSP	(note 3)
<b>Notes:</b> 1. See <i>JES3 Diagnosis</i> 2. See <i>JES3 Commands</i> 3. See <i>JES3 User Modifications and Macros</i>		

Figure 17-1. Table of Allowable DSPs for PROCESS Statements

### Location in the JCL

- The **//\*PROCESS** statements must immediately follow the **JOB** statement.
- Place all the **//\*PROCESS** statements immediately following each other, separated only by parameter statements.
- JES3 processes the **//\*PROCESS** statements in the order in which they appear in the input stream.
- If a **//\*NET** statement is specified for the job, the **//\*NET** statement must precede all **//\*PROCESS** statements.
- **//\*PROCESS** statements must precede their associated **EXEC** statements.
- Input service error messages that indicate a job is to be scheduled for interpreter processing before being purged are scheduled for an interpreter DSP only if the interpreter DSP is specified on the first **//\*PROCESS** statement.

## JES3: **//\*PROCESS**

### Examples of the **//\*PROCESS** Statement

```
//EXAM1 JOB
//*PROCESS CI
//*PROCESS MAIN
//*PROCESS OUTSERV
//S1 EXEC PGM=ANY
.
.
JCL statements
.
```

This example shows how to submit a simple job via **//\*PROCESS** statements. It executes in the same way as a standard job without JES3 control statements. The four standard scheduler elements are created for the job: CI, MAIN, OUTSERV, and PURGE.

---

```
//EXAM2 JOB
//*PROCESS CI
//*PROCESS MAIN
//*PROCESS OUTSERV
//*PROCESS PLOT
//*ENDPROCESS
//S1 EXEC PGM=ANY
//DD1 DD UNIT=34209,DISP=(NEW,KEEP)
.
.
JCL statements
.
```

This example shows how to use a user-written DSP. PLOT is a user-written DSP and is to be executed after output service has completed. Note that PURGE is not specified; JES3 automatically creates this DSP.

---

```
//EXAM3 JOB
//*PROCESS OUTSERV
//*FORMAT PR,DDNAME=DS1,COPIES=5
//*DATASET DDNAME=DS1
.
.
data
.
.
//*ENDDATASET
```

This example uses JES3 output service and the **//\*DATASET** statement. Five copies of data set DS1 are printed on any local printer.

## **ROUTE XEQ Statement**

**Purpose:** Use the **ROUTE XEQ** statement to identify the network node where the job is to execute.

**Syntax:**

```
ROUTE XEQ nodename[.vmguestid]
```

Do not imbed blanks in the nodename or vmguestid parameters.

### **Parameter Definition**

**nodename**

Indicates the node. The nodename identifies an MVS JES2 system, an MVS JES3 (global) system, a VSE POWER node, or a VM system.

Nodename should not specify the local node; if it does, the job executes locally.

**.vmguestid**

Identifies a guest system running in a virtual machine (VM), for example, an MVS system running under VM.

*Note:* Do not specify a work station or terminal in this parameter.

### **Location in the JCL**

- Place the **ROUTE XEQ** statement after a valid **JOB** statement for the submitting location.
- Place the **ROUTE XEQ** statement after any **NETACCT** statements.
- Place the **ROUTE XEQ** statement immediately before a valid **JOB** statement for the executing location.

*Note:* TSO users must code **NJB** in place of **JOB** on the second **JOB** statement, which is the **JOB** statement for the executing location.

## JES3: **//\*ROUTE XEQ**

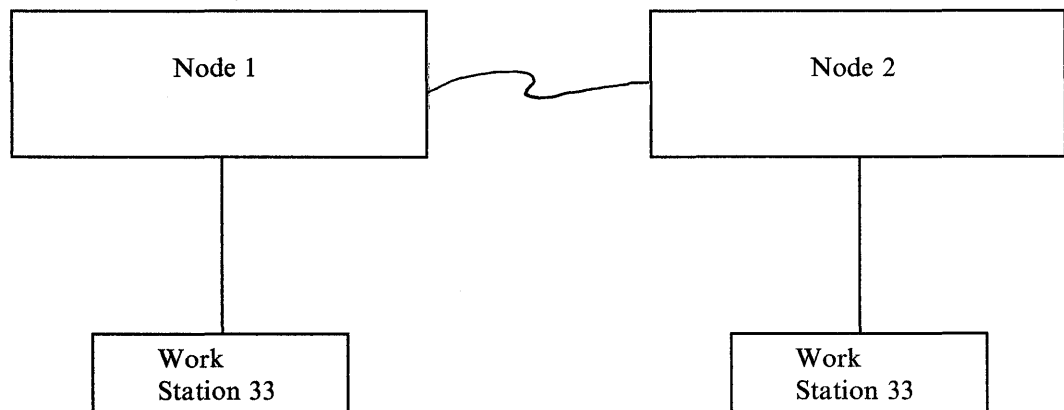
### Example of the **//\*ROUTE XEQ** Statement

```
//JOBN1 JOB options ...
//*ROUTE XEQ 2
//JOBN2 JOB options ...
//STEP1 EXEC PGM=REPORTER
//DD1 DD SYSOUT=A,DEST=N1R33
//DD2 DD SYSOUT=A,DEST=N2R33
//DD3 DD SYSOUT=B,DEST=R33
//DDIN DD *
.
.
data
.
/*
```

In this example, JOB statement JOBN1 is entered through the JES3 system at node 1. The **//\*ROUTE XEQ** statement tells JES3 to send the incoming job stream to node 2 until the **/\*** delimiter statement. JOB statement JOBN2 and all following statements until the delimiter are read and executed by the system at node 2.

The system output data sets are sent to two work stations:

- Sysout data set DD1 is produced at work station 33 attached to node 1, the submitting node.
- Sysout data set DD2 is produced at work station 33 attached to node 2, the executing node.
- Sysout data set DD3 is produced at work station 33, which is also attached to node 2. Because no node is specified, the executing node is assumed.





## /\*SIGNOFF Statement

**Purpose:** Use the /\*SIGNOFF statement to tell JES3 to terminate a remote job stream processing session. At the completion of the current print and/or punch input streams, JES3 disconnects the remote work station from the system. If JES3 is reading jobs from the station when the output completes, JES3 disconnects the station when the input is completed.

Both systems network architecture (SNA) and binary synchronous communication (BSC) remote work stations use the /\*SIGNOFF statement.

**References:** For more information on the /\*SIGNOFF command, see *SPL: JES3 Initialization and Tuning*.

**Syntax:**

```
/*SIGNOFF
```

Note that, unlike other JES3 statements, this statement starts with only one slash.

### Example of the /\*SIGNOFF Statement

```
/*SIGNOFF
```

This statement requests that JES3 terminate a remote job stream processing session.

## JES3: /\*SIGNON

### /\*SIGNON Statement

**Purpose:** Use the /\*SIGNON statement to tell JES3 to begin a remote job stream processing session. The /\*SIGNON statement can override the remote identification number normally assigned to the remote work station. This statement is optional for all work stations except non-multi-leaving remote stations on a switched line.

Systems network architecture (SNA) remote work stations must use the LOGON command instead of the /\*SIGNON statement to notify JES3 of a connection request.

#### Syntax:

```
/*SIGNON work-station-name {A|(blank)} {R|(blank)} passwd1 passwd2
```

- Note that, unlike other JES3 statements, this statement starts with only one slash.
- All the fields in this statement must appear in fixed locations, as follows:

Column	Contents
16-20	Work-station-name, beginning in 16
22	A or a blank
23	R or a blank
25-32	Password1, beginning in 25
35-42	Password2, beginning in 35

### Parameter Definition

#### work-station-name

Specifies the name of the remote work station. The work-station-name is 1 to 5 characters and must have been defined on a JES3 RJPTERM initialization statement.

#### A

Indicates an automatic reader. A can be coded only when the work station is a programmable terminal. Leave this column blank if you do not want to specify an automatic reader.

#### R

Indicates that print or punch output can be suspended if the needed device is not ready. R can be coded only when the work station is a nonprogrammable terminals. Leave this column blank if you do not want to specify the R option.

#### password1

Specifies the password for the remote job processing (RJP) line. password1 is 1 to 8 characters and must have been defined on a JES3 RJPLINE initialization statement.

#### password2

Specifies the password for the work station. password2 is 1 to 8 characters and must have been defined on a JES3 RJPTERM initialization statement.

**Example of the /\*SIGNON Statement**

```
/*SIGNON      QUIN  A  PSWD1      PSWD2
```

This statement requests that remote work station QUIN begin a remote job stream processing session. The value A in column 22 specifies an automatic reader for the programmable terminal. PSWD1, beginning in column 25, is the password assigned to a dial line. PSWD2, beginning in column 35, is the password assigned to the remote work station.



## Chapter 18. Reference Tables

The first table summarizes the DD statement parameters required to perform the following functions:

- Create
  - A data set on a unit record device (card punch or printer)
  - A data set on a system output device
  - A data set on magnetic tape
  - A data set on a direct access device
  - A data set on a Mass Storage System
  - A subsystem data set
- Retrieve
  - A data set from a unit record device (card punch or printer)
  - A data set from the input stream
  - A passed data set (magnetic tape or direct access)
  - A cataloged data set (magnetic tape or direct access)
  - A kept data set (magnetic tape or direct access)
- Extend
  - A passed data set (magnetic tape or direct access)
  - A cataloged data set (magnetic tape or direct access)
  - A kept data set (magnetic tape or direct access)

Also included are tables for:

- Retrieving or extending an indexed sequential data set
- Area arrangement of indexed sequential data sets
- Mutually exclusive DD parameters
- Disposition processing
- Direct access capacities
- Track capacities
- The JOB statement
- The EXEC statement
- The DD statement
- The OUTPUT statement

## DD Parameters for Creating Data Set

Device	Parameter Type	Parameter	Comments
Unit Record Devices	Location of the Data Set	UNIT	Required
	Data Attributes	DCB	Optional
	Special Processing Options	UCS	Optional (for a printer with the universal character set feature)
		FCB	Optional (for a 3211 or 3800 Printing Subsystem if forms control information is to be specified)
		FREE	Optional
		DUMMY	Optional
		COPIES	Optional
		CHARS	Optional (for the 3800 Printing Subsystem)
		BURST	Optional (for the 3800 Printing Subsystem)
		FLASH	Optional (for the 3800 Printing Subsystem)
MODIFY	Optional (for the 3800 Printing Subsystem)		
System Output Devices	Location of the Data Set	SYSOUT	Required. Specifies the output class
	Data Attributes	DCB	Optional
	Special Processing Options	OUTLIM	Optional
		FREE	Optional
		DEST	Optional
		DSID	Required for output to a 3540 diskette
		HOLD	Optional
		UCS	Optional (for a printer with the universal character set feature)
		FCB	Optional (for a 3211 or 3800 Printing Subsystem if forms control information is to be specified)
		COPIES	Optional
		CHARS	Optional (for the 3800 Printing Subsystem)
		BURST	Optional (for the 3800 Printing Subsystem)
		FLASH	Optional (for the 3800 Printing Subsystem)
		MODIFY	Optional (for the 3800 Printing Subsystem)
Magnetic Tape	Data Set Information	DSNAME (or DSN)	Required if the data set is to be cataloged or used by a later job
		DISP	Required if the data set is to be cataloged, used by a later step in this job, or used by another job
	Location of the Data Set	UNIT	Required unless you request (with the VOLUME parameter) the same volume used for an earlier data set in your job
		VOLUME (or VOL)	Required if you want a specific volume. If you do not use this parameter you will get a scratch tape
		LABEL	Required if you do not want to use IBM standard labels for the data set
	Data Attributes	DCB	Optional
	Special Processing Options	DUMMY	Optional
		CHKPT	Optional
		FREE	Optional
PROTECT		Optional	

Figure 18-1 (Part 1 of 2). DD Parameters for Creating a Data Set

## DD Parameters for Creating Data Set

Device	Parameter Type	Parameter	Comments
Direct Access Devices	Data Set Information	DSNAME (or DSN)	Required if the data set is to be cataloged or used by a later job
		DISP	Required if the data set is to be cataloged, used by a later step in this job, or used by another job
	Location of the Data Set	UNIT	Required unless you request (with the VOLUME parameter) the same volume used for an earlier data set in your job
		VOLUME (or VOL)	Required if you want a specific volume or multiple volumes. If you do not use this parameter your data set will be allocated on any suitable volume
		LABEL	Required if you want the data set to have both IBM standard and user labels
	Data Attributes	DCB	Optional
	Size of the Data Set	SPACE	SPACE must be used for ISAM data sets
	Special Processing Options	DUMMY	Optional
		DYNAM	Optional
		FREE	Optional
		PROTECT	Optional
Mass Storage System (MSS)	Data Set Information	DSNAME (or DSN)	Required if the data set is to be cataloged or used by another job
		DISP	Required if the data set is to be cataloged, used by a later step in the job, or used by another job
	Location of the Data Set	UNIT	Required unless you request (with the VOLUME parameter) the same volume used for an earlier data set in your job
		VOLUME (or VOL)	Required for specific volume requests. Use MSVGP instead of VOL=SER if a nonspecific volume in a specific MSS volume group is desired. If neither is coded, the system will select an already mounted 3330V volume (storage or public) unless PRIVATE is coded
		LABEL	Required if you want the data set to have both IBM standard and user labels
		MSVGP	Required if a nonspecific volume in a specific MSS volume group is required
	Data Attributes	DCB	Optional
	Size of the Data Set	SPACE	Required unless MSVGP is coded
	Special Processing Options	DUMMY	Optional
		DYNAM	Optional
		FREE	Optional
PROTECT		Optional	
Subsystem Data Set	Location of the Data Set	SUBSYS	Required. Specifies the subsystem and the subsystem-defined parameters
	Data Attributes	DCB	Optional
	Special Processing Options	DUMMY	Optional
		FREE	Optional

Figure 18-1 (Part 2 of 2). DD Parameters for Creating a Data Set

## DD Parameters for Retrieving Data Set

Data Set	Parameter Type	Parameter	Comments
Unit Record	Location of the Data Set	UNIT	Required
	Data Attributes	DCB	Optional
	Special Processing Options	DUMMY	Optional
FREE		Optional	
Input Stream	Location of the Data Set	*	You must specify one of these parameters
		DATA	
	Data Attributes	DCB	Optional
	Special Processing Options	DLM	Optional
FREE		Optional	
Associated Data Set	Location of the Data Set	*	You must specify one of these parameters
		DATA	
	Data Attributes	DCB	Optional
	Data Set Information	DSID	Required for 3540 associated data sets
		VOL = SER	Optional for 3540 associated data sets
Passed Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Required only if you want more units
		LABEL	Required only if the data set does not have IBM standard labels
	Data Attributes	DCB	Optional
	Special Processing Options	FREE	Optional
		CHKPT	Optional
DUMMY		Optional	
Cataloged Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Optional
		VOLUME	May be required if you want to begin processing with a volume after the first
		LABEL	Required only if the data set does not have IBM standard labels
	Data Attributes	DCB	Optional
	Special Processing Options	DUMMY	Optional
		DYNAM	Optional
		FREE	Optional
CHKPT		Optional	
Kept Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Required
		VOLUME	Required
		LABEL	Required only if the data set does not have IBM standard labels
	Data Attributes	DCB	Optional
	Special Processing Options	DUMMY	Optional
		DYNAM	Optional
FREE		Optional	
CHKPT		Optional	

Figure 18-2. DD Parameters for Retrieving a Data Set



## DD Parameters for Extending Data Set

Data Set	Parameter Type	Parameter	Comments
Passed Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Required only if you want more units
		VOLUME	Required only if you need more volumes
		LABEL	Required only if the data set does not have IBM standard labels
	Size of the Data Set	SPACE	Required only if you want to override the secondary quantity
	Data Attributes	DCB	May be required if data set does not have IBM standard labels
	Special Processing Options	FREE	Optional
		CHKPT	Optional
		DUMMY	Optional
Cataloged Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Optional
		VOLUME	Required only if you need more volumes
		LABEL	Required only if the data set does not have IBM standard labels
	Size of the Data Set	SPACE	Required only if you want to override the secondary quantity
	Data Attributes	DCB	Required only if the data set does not have IBM standard labels
	Special Processing Options	DUMMY	Optional
		FREE	Optional
		CHKPT	Optional
Kept Data Set	Data Set Information	DSNAME	Required
		DISP	Required
	Location of the Data Set	UNIT	Required
		VOLUME	Required
		LABEL	Required only if data set does not have IBM standard labels
	Size of the Data Set	SPACE	Required only if you want to override the secondary quantity
	Data Attributes	DCB	Required only if the data set does not have IBM standard labels
	Special Processing Options	DUMMY	Optional
		DYNAM	Optional
		FREE	Optional
CHKPT		Optional	

Figure 18-3. DD Parameters for Extending a Data Set

## DD Parameters for ISAM

Area	Parameter Type	Parameter	Comments	
Index (used only if index area not on same device type as prime area) (First DD statement)	Data Set Information	DSNAME	Required. You must code the same value as in second DD statement.	
		DISP	Required. You must code the same value as in second DD statement.	
	Location of the data set	UNIT	Required	
		VOLUME	Required	
Data Attributes	DCB	Required		
	Prime and Overflow; or Index, Prime, and Overflow; or Index and Prime (required) (Second DD statement)	Data Set Information	DSNAME	Required
		DISP	Required. Specifies whether you are retrieving the data set.	
Location of the data set		UNIT	Required unless it is a passed data set with all three areas on one volume.	
		VOLUME	Same requirement as UNIT. If used, code volumes in order they were defined.	
Data Attributes	DCB	Required		
Overflow (used only if overflow area not on same device type as prime area) (Third DD Statement)	Data Set Information	DSNAME	Required. You must code the same value as in second DD statement.	
		DISP	Required. You must code the same value as in the second DD statement.	
	Location of the data set	UNIT	Required	
		VOLUME	Required	
Data Attributes	DCB	Required		

Figure 18-4. DD Parameters for Retrieving or Extending an Indexed Sequential Data Set

## Area Arrangement of ISAM

CRITERIA			RESTRICTIONS ON DEVICE TYPES AND NUMBER OF DEVICES REQUESTED	RESULTING ARRANGEMENT OF AREAS
1. Number of DD statements	2. Area defined on a DD statement	3. Index size coded?		
3	INDEX PRIME OVFLOW	-	NONE	Separate index, prime, and overflow areas.
2	INDEX PRIME	-	None	Separate index and prime area. <sup>1</sup>
2	PRIME OVFLOW	No	None	Separate prime and overflow areas. An index area is at the end of the overflow area.
2	PRIME OVFLOW	Yes	The statement defining the prime area cannot request more than one device.	Separate prime and overflow areas. An index area is embedded in the prime area.
1	PRIME	No	None	Prime area with index area at its end. <sup>2</sup>
1	PRIME	Yes	Cannot request more than one device.	Prime area with embedded index area. <sup>2</sup>
<sup>1</sup> If both areas are on volumes that correspond to the same device type, an overflow area is established if one of the cylinders allocated for the index area is only partially used. The overflow area is established in the unused portion of that cylinder.				
<sup>2</sup> If the index area is at least one cylinder and the unused portion of the index area is less than one cylinder, the unused portion is used as an overflow area. For a one-cylinder data set, the index is not assigned a whole cylinder; therefore, no overflow area is provided.				

**Figure 18-5. Area Arrangement of Indexed Sequential Data Sets**

# Mutually Exclusive DD Parameters

The table is a triangular grid where the horizontal axis represents the DD parameters and the vertical axis represents the same parameters. The parameters listed on the right side of the grid are: \*, AMP, BURST, CHARS, CHKPT, COPIES, DATA, DCB, DDNAME, DEST, DISP, DLM, DSID, DSNNAME, DUMMY, DYNAM, FCB, FLASH, FREE, HOLD, LABEL, MODIFY, MSVGP, OUTLIM, OUTPUT, PROTECT, QNAME, SPACE, SYBSYS, SYSOUT, TERM, UCS, UNIT, VOLUME. The grid shows shaded squares at the intersections of parameters that cannot be coded together. The shaded squares form a triangular pattern, indicating that each parameter is mutually exclusive with itself and with all other parameters listed.

**Legend:** This table shows which DD parameters cannot be coded together. At the intersection of the horizontal and vertical columns, the square will be shaded if the parameters are mutually exclusive and white if they can be coded together on the same DD statement.

For example, to see if DISP and SYSOUT are mutually exclusive, look down the column marked DISP and across the column marked SYSOUT. In this case, they are mutually exclusive.

As indicated by the table, each DD parameter is mutually exclusive with itself; that is, it cannot appear twice on the same DD statement.

**Figure 18-6. Table of Mutually Exclusive DD Parameters**

Table 18-7. Disposition Processing Table

Status	Requested Disposition	Conditional Disposition	or Normal End of Step <sup>1</sup>	Program canceled or abnormally terminated	Subsequent data set allocation for same step failed	End of Job
NEW or MOD <sup>2</sup>	none	none	deleted	deleted	deleted	
	KEEP	none	kept	kept	deleted	
	DELETE	none	deleted	deleted	deleted	
	CATLG	none	cataloged	cataloged	deleted	
	PASS	none	passed	passed	passed	deleted
	PASS	any	passed	passed	passed	deleted <sup>5</sup>
	any except PASS	KEEP	requested disposition	kept	deleted	
	any except PASS	DELETE	requested disposition	deleted	deleted	
	any except PASS	CATLG	requested disposition	cataloged	deleted	
OLD or MOD or SHR	none	none	kept	kept	kept	
	KEEP	none	kept	kept	kept	
	DELETE	none	deleted	deleted	kept	
	CATLG	none	cataloged <sup>3</sup>	cataloged <sup>3</sup>	kept	
	UNCATLG	none	uncataloged	uncataloged	kept	
	PASS	none	passed	passed	passed	kept
	PASS	any	passed	passed	passed	kept <sup>6</sup>
	any except PASS	KEEP	requested disposition	kept	kept <sup>4</sup>	
	any except PASS	DELETE	requested disposition	deleted	kept <sup>4</sup>	
any except PASS	CATLG	requested disposition	cataloged <sup>3</sup>	kept <sup>4</sup>		
any except PASS	UNCATLG	requested disposition	uncataloged	kept <sup>4</sup>		

**Footnotes:**

- <sup>1</sup> See list of exceptions in right-hand column.
- <sup>2</sup> If volume information is not available to the system, a MOD data set is considered to be a new data set.
- <sup>3</sup> If volumes were added to a data set for which unit and volume information was retrieved from the catalog, the data set is actually recataloged.
- <sup>4</sup> If the step was attempting to receive a passed data set which was new when initially passed, the data set is deleted.
- <sup>5</sup> If any job steps reached abnormal termination, the conditional disposition will be processed. Otherwise, the data set is deleted.
- <sup>6</sup> If any job steps reached abnormal termination, the conditional disposition will be processed. Otherwise, the data set is kept if it was old when initially passed in the job, or deleted if it was new when originally passed in the job.

**List of Exceptions:**

- When a nontemporary data set is passed and the receiving step does not assign it a disposition, the system will, upon termination of this step, do one of two things. If the data set was new when it was initially passed, it will be deleted. If the data set was old when initially passed, it will be kept. Temporary data sets are deleted.
- If automatic step restart is to occur, all data sets in the restart step with a status of OLD are kept. All data sets in the restart step with a status of NEW are deleted.
- If automatic checkpoint restart is to occur, all data sets currently in use by the job are kept.
- If a data set is assigned a temporary name or no name, a conditional disposition other than DELETE is invalid. The system assumes DELETE.
- If the data set is not allocated, then no action is taken.

## Direct Access Capacities

Device	Storage Medium	Cylinders	Tracks Per Cylinder	Bytes Per Track	Bytes Per Cylinder	Bytes Per Device (in millions)
2314/2319 (each volume)	Disk	200	20	7,294	145,880	29.17
2305 Model 1	Disk	48	8	14,136	113,088	5.4
2305 Model 2	Disk	96	8	14,660	117,280	11.3
3330	Disk	404	19	13,030	247,570	100
3330 Mod II	Disk	808	19	13,030	247,570	200
3340/3344	Disk	696 (70-megabytes) 348 (35-megabytes)	12	8,368	100,416	69.8 (70-megabytes) 34.9 (35-megabytes)
3350	Disk	555	30	19,069	572,070	317.5
3375	Disk	959	12	35,616	427,392	409.8
3380	Disk	885	15	47,476	712,140	630.2
<p>Note: 3344 pertains only to the 70-megabyte 3340</p>						

Figure 18-8. Direct Access Capacities

# Track Capacities

Physical Records per Track	Maximum Bytes per Record Formatted without Keys							
	2314/ 2319	2305-1	2305-2	3330/ 3330 Mod II	3340/ 3344	3350	3375	3380 <sup>2</sup>
1	7294	14136	14660	13030	8368	19069	35616 <sup>1</sup>	47476 <sup>1</sup>
2	3520	6852	7231	6447	4100	9442	17600	23476
3	2298	4424	4754	4253	2678	6233	11616	15476
4	1693	3210	3516	3156	1966	4628	8608	11476
5	1332	2480	2773	2498	1540	3665	6816	9076
6	1092	1996	2278	2059	1255	3024	5600	7476
7	921	1648	1924	1745	1052	2565	4736	6356
8	793	1388	1659	1510	899	2221	4096	5492
9	694	1186	1452	1327	781	1954	3616	4820
10	615	1024	1287	1181	686	1740	3200	4276
11	550	892	1152	1061	608	1565	2880	3860
12	496	782	1040	962	544	1419	2592	3476
13	450	688	944	877	489	1296	2368	3188
14	411	608	863	805	442	1190	2176	2932
15	377	538	792	742	402	1098	2016	2676
16	347	478	730	687	366	1018	1856	2484
17	321	424	676	639	335	947	1728	2324
18	298	376	627	596	307	884	1600	2164
19	276	334	584	557	282	828	1504	2004
20	258	296	544	523	259	777	1408	1876
21	241	260	509	491	239	731	1312	1780
22	226	230	477	463	220	690	1248	1684
23	211	200	448	437	204	652	1152	1588
24	199	174	421	413	188	617	1088	1492
25	187	150	396	391	174	585	1056	1396
26	176	128	373	371	161	555	992	1332
27	166	106	352	352	149	528	928	1268
28	157	88	332	335	137	502	896	1204
29	148	70	314	318	127	478	832	1140
30	139	52	297	303	117	456	800	1076
<p><sup>1</sup> Standard access methods support records up to 32K data length; only EXCP supports records greater than 32K data length.</p> <p><sup>2</sup> For the 3380, the value is the data length. To obtain the value, the length is rounded up to a multiple of 32, then 12 bytes are subtracted.</p>								

Figure 18-9 (Part 1 of 2). Track Capacities

# Track Capacities

Physical Records per Track	Maximum Bytes per Record Formatted with Keys							
	2314/2319	2305-1	2305-2	3330/3330 Mod II	3340/3344	3350	3375 <sup>2</sup>	3380 <sup>2</sup>
1	7249	13934	14569	12974	8293	18987	35456 <sup>1</sup>	47240 <sup>1</sup>
2	3476	6650	7140	6391	4025	9360	17440	23240
3	2254	4222	4663	4197	2603	6151	11456	15240
4	1649	3008	3425	3100	1891	4546	8448	11240
5	1288	2278	2682	2442	1465	3583	6656	8840
6	1049	1794	2187	2003	1180	2942	5440	7240
7	877	1446	1833	1689	977	2483	4576	6120
8	750	1186	1568	1454	824	2139	3936	5256
9	650	984	1361	1271	706	1872	3456	4584
10	571	822	1196	1125	611	1658	3040	4040
11	506	690	1061	1005	533	1483	2720	3624
12	452	580	949	906	469	1337	2432	3240
13	407	486	853	821	414	1214	2208	2952
14	368	406	772	749	367	1108	2016	2696
15	333	336	701	686	327	1016	1856	2440
16	304	276	639	631	291	936	1696	2248
17	277	222	585	583	260	865	1568	2088
18	254	174	536	540	232	802	1440	1928
19	233	132	493	501	202	746	1344	1768
20	215	94	453	467	184	695	1248	1640
21	198	58	418	435	164	649	1152	1544
22	183		386	407	145	608	1088	1448
23	168		357	381	129	570	992	1352
24	156		330	357	113	535	928	1256
25	144		305	335	99	503	896	1160
26	133		282	315	86	473	832	1096
27	123		261	296	74	446	768	1032
28	114		241	279	62	420	736	968
29	105		223	262	52	396	672	904
30	96		206	247	42	374	640	804

<sup>1</sup> Standard access methods support records up to 32K data length; only EXCP supports records greater than 32K data length.

<sup>2</sup> For the 3380, the value is the key length plus a data length. To obtain the value, both lengths are rounded up to multiples of 32, then 24 bytes are subtracted.

Figure 18-9 (Part 2 of 2). Track Capacities





# EXEC Statement

The EXEC Statement				
//Name	Operation	Operand	P/K	Comments
//[stepname]	EXEC	ACCT[.procstepname] = (accounting-information)	K	Specifies accounting information for step.
		ADDRSPC [.procstepname] = $\left\{ \begin{array}{l} \text{VIRT} \\ \text{REAL} \end{array} \right\}$	K	Requests virtual or real storage.
		COND [.procstepname] = ((code,operator[,stepname][.procstepname]) [, (code,operator[,stepname][.procstepname]) . . . [,EVEN) ] [,ONLY) ]	K	Specifies test for a return code. Note: COND is supported only by JES2.
		DPRTY [.procstepname] = ((value1)[,value2])	K	Assigns a dispatching priority to the address space for the step.
		DYNAMNBR [.procstepname] = n	K	Tells the system to hold resources in anticipation of reuse.
		PARM [.procstepname] = information	K	Passes variable information to a program at execution time.
		PERFORM [.procstepname] = n	K	Specifies the performance group for the job step.
		PGM = $\left\{ \begin{array}{l} \text{program-name} \\ *.\text{stepname}.\text{ddname} \\ *.\text{stepname}.\text{procstepname}.\text{ddname} \end{array} \right\}$	P	Identifies program to be executed.
		[PROC=] procedure name	P	Identifies a cataloged or in-stream procedure to be called and executed.
		RD [.procstepname] = $\left\{ \begin{array}{l} \text{R} \\ \text{RNC} \\ \text{NC} \\ \text{NR} \end{array} \right\}$	K	Specifies how a job step is to be restarted.
		REGION [.procstepname] = valueK	K	Specifies amount of space the step requires.
		TIME [.procstepname] = $\left\{ \begin{array}{l} [minutes][,seconds] \\ 1440 \end{array} \right\}$	K	Specifies a processor time limit for the job.

**Legend:**

- P Positional parameter. (Positional parameters must precede keyword parameters)
- K Keyword parameter.
- { } Choose one.
- [ ] Optional; if more than one line is enclosed, choose one or none.

**Figure 18-11. The EXEC Statement**

The DD Statement				
//Name	Operation	Operand	P/K	Comments
// [ddname procstepname. ddname]	DD	*	P	Begins an in-stream data set.
		ACCODE = access-code	K	Specifies or changes an accessibility code.
		$\left. \begin{array}{l} \text{AMORG} \\ \text{BUFND} = \text{number} \\ \text{BUFNI} = \text{number} \\ \text{BUFSP} = \text{bytes} \\ \text{CROPS} = \left. \begin{array}{l} \text{RCK} \\ \text{NCK} \\ \text{NRE} \\ \text{NRC} \end{array} \right\} \\ \text{AMP} = \left. \begin{array}{l} \text{OPTCD} = \left. \begin{array}{l} \text{I} \\ \text{L} \\ \text{IL} \end{array} \right\} \\ \text{RECFM} = \left. \begin{array}{l} \text{F} \\ \text{FB} \\ \text{V} \\ \text{VB} \end{array} \right\} \\ \text{STRNO} = \text{number} \\ \text{SYNAD} = \text{modulename} \\ \text{TRACE} \end{array} \right\}$	K	Completes the access method control block (ACB) for VSAM data sets.
		BURST = $\left. \begin{array}{l} \{ \text{YES Y} \} \\ \{ \text{NO N} \} \end{array} \right\}$	K	Specifies whether or not output is to go to the Burster-Trimmed-Stacker of the 3800.
		CHARS = $\left\{ \left. \begin{array}{l} \{ \text{table-name[, table-name] . . .} \} \\ \{ \{ \text{DUMP} \} [, \text{table-name} . . .] \} \end{array} \right\}$	K	Specifies character-arrangement table(s) for a 3800. Note: DUMP is supported only by JES3.
		CHKPT = EOVS	K	Requests checkpoint at end of volume.
		CNTL = $\left\{ \begin{array}{l} *.\text{label} \\ *.\text{stepname}.\text{label} \\ *.\text{stepname}.\text{procstepname}.\text{label} \end{array} \right\}$	K	References CNTL/ENDCNTL group.
		COPIES = (nnn[, (group-value[, group-value] . . .)])	K	Requests multiple copies (and grouping, for the 3800 only) of the output data set.
		DATA	P	Begins an in-stream data set.
		$\left[ \begin{array}{l} \text{DCB} = (\text{subparameter[, subparameter] . . .}) \\ \text{DCB} = \left( \begin{array}{l} \text{dsname} \\ *.ddname \\ *.stepname.ddname \\ *.stepname.procstepname.ddname \end{array} \right) \left[ \text{[, subparameter] . . .} \right] \end{array} \right]$	K	Completes the data control block (used for all data sets except VSAM).
		DDNAME = ddname	K	Postpones defining a data set.
		$\text{DEST} = \left\{ \begin{array}{l} \left. \begin{array}{l} \text{JES2} \\ \text{LOCAL} \\ \text{name} \\ \text{Nnn} \\ \text{NnnRmmm} \\ \text{Rnnnn} \\ \text{RMnnnn} \\ \text{RMTnnn} \\ \text{Unnn} \\ (\text{node,userid}) \end{array} \right\} \left\{ \begin{array}{l} \text{JES3} \\ \text{ANYLOCAL} \\ \text{device-name} \\ \text{device-address} \\ \text{group-name} \\ \text{node name} \\ (\text{node,userid}) \end{array} \right\} \end{array} \right\}$	K	Specifies a destination for a system output data set.

Table 18-12

Figure 18-12 (Part 1 of 3). The DD Statement

# DD Statement

The DD Statement (con't)																																				
//Name	Operation	Operand	P/K	Comments																																
		DISP = ( <table border="0"> <tr> <td>[NEW]</td> <td>[DELETE]</td> <td>[DELETE]</td> </tr> <tr> <td>[OLD]</td> <td>[KEEP]</td> <td>[KEEP]</td> </tr> <tr> <td>[SHR]</td> <td>[PASS]</td> <td>[CATLG]</td> </tr> <tr> <td>[MOD]</td> <td>[CATLG]</td> <td>[UNCATLG]</td> </tr> <tr> <td></td> <td>[UNCATLG]</td> <td>[UNCATLG]</td> </tr> </table> )	[NEW]	[DELETE]	[DELETE]	[OLD]	[KEEP]	[KEEP]	[SHR]	[PASS]	[CATLG]	[MOD]	[CATLG]	[UNCATLG]		[UNCATLG]	[UNCATLG]	K	Assigns a status, normal termination disposition, abnormal termination disposition to the data set																	
[NEW]	[DELETE]	[DELETE]																																		
[OLD]	[KEEP]	[KEEP]																																		
[SHR]	[PASS]	[CATLG]																																		
[MOD]	[CATLG]	[UNCATLG]																																		
	[UNCATLG]	[UNCATLG]																																		
		DLM = delimiter	K	Assigns delimiter other than *																																
		DSID = (id, V)	K	Identifies an input or output data set for diskette.																																
		{DSNAME} = { <table border="0"> <tr><td>dsname</td></tr> <tr><td>dsname(member-name)</td></tr> <tr><td>dsname(generation-number)</td></tr> <tr><td>dsname(area-name)</td></tr> <tr><td>&amp;&amp;dsname</td></tr> <tr><td>&amp;&amp;dsname(member-name)</td></tr> <tr><td>&amp;&amp;dsname(area-name)</td></tr> <tr><td>*.ddname</td></tr> <tr><td>*.stepname.ddname</td></tr> <tr><td>*.stepname.procstepname.ddname</td></tr> <tr><td>NULLFILE</td></tr> </table> }	dsname	dsname(member-name)	dsname(generation-number)	dsname(area-name)	&&dsname	&&dsname(member-name)	&&dsname(area-name)	*.ddname	*.stepname.ddname	*.stepname.procstepname.ddname	NULLFILE	K	Assigns a name to a new data set or identifies an existing data set.																					
dsname																																				
dsname(member-name)																																				
dsname(generation-number)																																				
dsname(area-name)																																				
&&dsname																																				
&&dsname(member-name)																																				
&&dsname(area-name)																																				
*.ddname																																				
*.stepname.ddname																																				
*.stepname.procstepname.ddname																																				
NULLFILE																																				
		DUMMY	P	Bypasses I/O operations on the data set (BSAM and QS)																																
		DYNAM	P	Specifies that the system should hold a resource in anticipation of reuse.																																
		FCB = (fcb-name [ALIGN,VERIFY])	K	Specifies the forms control buffer, carriage control data-protection image.																																
		FLASH = {(overlay name[,count]) NONE}	K	Identifies the forms overlays to be used on the 3800.																																
		FREE = {END, CLOSE}	K	Specifies deallocation action at data set close																																
		HOLD = {[YES Y] [NO N]}	K	Tells the system to hold system output data set released by operator.																																
		LABEL = ((data-set-seq-#) <table border="0"> <tr><td>[.SL]</td><td>[PASSWORD]</td><td>[IN]</td><td>[EXPDT = yyddd]</td></tr> <tr><td>[.SUL]</td><td>[NOPWREAD]</td><td>[OUT]</td><td>[RETPD = nnnn]</td></tr> <tr><td>[.AL]</td><td></td><td></td><td></td></tr> <tr><td>[.AUL]</td><td></td><td></td><td></td></tr> <tr><td>[.NSL]</td><td></td><td></td><td></td></tr> <tr><td>[.NL]</td><td></td><td></td><td></td></tr> <tr><td>[.BLP]</td><td></td><td></td><td></td></tr> <tr><td>[.LTM]</td><td></td><td></td><td></td></tr> </table> )	[.SL]	[PASSWORD]	[IN]	[EXPDT = yyddd]	[.SUL]	[NOPWREAD]	[OUT]	[RETPD = nnnn]	[.AL]				[.AUL]				[.NSL]				[.NL]				[.BLP]				[.LTM]				K	Supplies label information for tape data set.
[.SL]	[PASSWORD]	[IN]	[EXPDT = yyddd]																																	
[.SUL]	[NOPWREAD]	[OUT]	[RETPD = nnnn]																																	
[.AL]																																				
[.AUL]																																				
[.NSL]																																				
[.NL]																																				
[.BLP]																																				
[.LTM]																																				
		MODIFY = (module name[,trc])	K	Specifies a copy modification module to be used on the 3800 and which CHARACTER name is to be used.																																

Figure 18-12 (Part 2 of 3). The DD Statement

The DD Statement (con't)				
Name	Operation	Operand	P/K	Comments
		MSVGP = { (id[, ddname]) SYSGROUP }	K	Requests a mass storage group for a mass storage system (MSS) device.
		OUTLIM = number	K	Limits the number of logical records in the system output data set.
		OUTPUT = { (*.name[, *.name] . . . ) { (*.stepname.name[, *.stepname.name] . . . ) { (*.stepname.procstepname.name[, *.stepname.procstepname.name] . . . ) } } }	K	Identifies OUTPUT JCL statement(s) to use in processing this data set.
		PROTECT = YES	K	Requests RACF protection for tape volumes or for direct access data sets.
		QNAME = procname[.tcamname]	K	Specifies the name of a TPROCESS macro that defines a destination queue for messages received via TCAM.
		SPACE = ( { TRK CYL blocklength } , (primary-quantity [secondary-quantity] [directory] ) [.RLSE] [CONTIG MXIG ALX] [ROUND] )	K	Assigns space on a direct access volume for a new data set.
		SPACE = ( ABSTR, (primary-quantity, address [directory] ) [.index] )	K	Assigns specific tracks on a direct access volume for a new data set.
		SUBSYS = (subsystem-name[, subsystem-subparameter] . . . )	K	Specifies the subsystem to process the data set.
		SYSOUT = (class-name [writer-name] [form-name] [.code-name] )	K	Indicates a system output data set and assigns it to an output class.
		TERM = TS	K	Indicates that the data set is coming from or going to a terminal for a TSO user.
		UCS = (character-set-code [FOLD] [, VERIFY]) [.]	K	Requests a special character set.
		UNIT = ( [device-address] [unit-count] [device-type] [P] [, DEFER] ) [.group-name] )	K	Asks the system to place the data set on a device.
		UNIT = AFF = ddname		
		{ VOLUME } = ([PRIVATE] [, RETAIN] [, volume-seq-number] [, volume-count] [SER = (serial-number[, serial-number] . . . ) REF = dsname REF = *.ddname REF = *.stepname.ddname REF = *.stepname.procstepname.ddname] [, ] )	K	Identifies the volume(s) on which a data set resides or will reside.

itional parameter. (Positional parameters must precede /word parameters)  
/word parameter.

{ } Choose one.  
[ ] Enclosing subparameter, indicates that subparameter is optional; if more than one line is enclosed, choose one or more.

Tables

# OUTPUT JCL Statement

The OUTPUT Statement				
//Name	Operation	Operand	P/K	Comments
//name	OUTPUT	<p>BURST = {            {YES Y}            {NO N}}</p> <p>CHARS = {            {(table-name[, table-name] . . . )            STD            (DUMPI[, table-name] . . . )}</p> <p>CKPTLINE=nnnnn</p> <p>CKPTPAGE=nnnnn</p> <p>CKPTSEC=nnnnn</p> <p>CLASS = {            *            class-name}</p> <p>COMPACT = compaction-table-name</p> <p>CONTROL = {            PROGRAM            SINGLE            DOUBLE            TRIPLE}</p> <p>COPIES = (nnn[, group-value[ group-value] . . . ))</p> <p>DEFAULT = {            {YES Y}            {NO N}}</p> <p>DEST = {            JES2            LOCAL            name            Nnnnn            NnnRmmmm            NnnnRmmm            NnnnnRmm            nodename.userid            Rnnnn            RMnnnn            RMTnnnn            Unnn            JES3            device-name            group-name            LOCAL            nodename            nodename.remote            (type)}</p> <p>FCB = fcb-name</p>	<p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p> <p>K</p>	<p>Specifies whether or not output is to go to the Burster-Trimmer-Stacker of the 3800.</p> <p>Specifies a character-arrangement table(s) for a 3800. Note: STD and DUMP are supported only by JES3.</p> <p>Specifies the maximum number of lines in a logical page.</p> <p>Specifies the number of logical pages to be printed or transmitted before a checkpoint.</p> <p>Specifies the number of seconds that may elapse between checkpoints.</p> <p>Assigns the system output data set to an output class.</p> <p>Specifies a compaction table for sending this system output data set to an SNA remote terminal.</p> <p>Specifies forms control.</p> <p>Specifies the number of copies to be printed.</p> <p>Specifies that this OUTPUT statement can or cannot be implicitly referenced by a sysout DD statement.</p> <p>Specifies a destination for the data set.</p> <p>Specifies the forms control buffer, carriage control tape, or data-protection image.</p>
<p><b>Legend:</b></p> <p>P Positional parameter. (Positional parameters must precede keyword parameters)      {} Choose one.</p> <p>K Keyword parameter.      [] Enclosing subparameter, indicates that subparameter is optional if more than one line is enclosed, choose one or more.</p>				

Figure 18-13 (Part 1 of 2). The OUTPUT JCL Statement

# OUTPUT JCL Statement

The OUTPUT Statement (con't)				
//Name	Operation	Operand	P/K	Comments
		FLASH = { ((overlay-name)[,count]) NONE STD }	K	Identifies the forms overlay to be used on the 3800. Note: STD is supported only by JES3.
		FORMDEF = membername	K	Identifies a library member to be used by PSF for printing on a 3800.
		FORMS = { form-name STD }	K	Identifies the forms for printing or punching.
		GROUPID = output-group	K	Specifies that the system output data set belongs to an output group.
		INDEX = nn	K	Sets the left margin for output on a 3211 printer with the indexing feature.
		JESDS = { ALL JCL LOG MSG }	K	Specifies the processing options for the job's system data sets.
		LINDEX = nn	K	Sets the right margin for output on a 3211 printer with the indexing feature.
		LINECT = nnn	K	Specifies the maximum number of lines to be printed on each output page. Note: LINECT is supported only by JES2.
		MODIFY = ((module-name)[,trc])	K	Specifies a copy-modification module to be used on the 3800 and which CHARS table-name is to be used.
		PAGEDEF = membername	K	Identifies a library member to be used by PSF for printing on a 3800.
		PIMSG = { (YES Y) (NO N) }	K	Indicates that messages from a functional subsystem should or should not be printed.
		PRMODE = { LINE PAGE process-mode }	K	Identifies the process mode required to print this system output data set. Note: Process-mode is supported only by JES2.
		PRTY = nnn	K	Specifies the priority at which the system output data set enters the output queue.
		THRESHLD = limit	K	Specifies the maximum output to be printed as one unit of work.
		TRC = { (YES Y) (NO N) }	K	Specifies whether the system output data set contains TRC codes or not.
		UCS = character-set-code	K	Requests a special character set.
		WRITER = name	K	Names an external writer to process the output data set.

**Legend:**

P	Positional parameter. (Positional parameters must precede keyword parameters)	{ }	Choose one.
K	Keyword parameter.	[ ]	Enclosing subparameter, indicates that subparameter is optional if more than one line is enclosed, choose one or more.

Figure 18-13 (Part 2 of 2). The OUTPUT JCL Statement





## Index

- ..., use 2-3
- .., use 2-3, 2-18
- ++ (in-stream procedure) 9-14
- ++\* (in-stream procedure) 9-14
- +/ (in-stream procedure) 9-14
- | use 2-2
- & use 2-11
- && identifying a temporary data set 7-4
- &&dsname
  - for indexed sequential data sets 12-70
  - for partitioned data sets 12-70
  - in DSNAMES parameter 12-70
- { } braces, use 2-2
- \$A operator command (JES2) 16-2
- \$B operator command (JES2) 16-2
- \$C operator command (JES2) 16-2
- \$D operator command (JES2) 16-2
- \$E operator command (JES2) 16-2
- \$F operator command (JES2) 16-2
- \$G operator command (JES2) 16-2
- \$H operator command (JES2) 16-2
- \$I operator command (JES2) 16-2
- \$L operator command (JES2) 16-2
- \$M operator command (JES2) 16-2
- \$N operator command (JES2) 16-2
- \$O operator command (JES2) 16-2
- \$P operator command (JES2) 16-2
- \$R operator command (JES2) 16-2
- \$S operator command (JES2) 16-2
- \$T operator command (JES2) 16-2
- \$TR operator command (JES2) 16-2
- \$VS operator command (JES2) 16-2
- \$Z operator command (JES2) 16-2
- \*
  - DD statement parameter 12-6
  - in /\*JOBPARM statement 16-6
  - in /\*OUTPUT statement 16-14
  - in PGM parameter 11-21
  - in RESTART parameter 10-34
  - restriction for cataloged and in-stream procedures 9-1
- \* DD parameter
  - data in input stream 12-6
  - description 12-6
  - examples 12-7
  - relationship to other parameters 12-6
  - syntax 12-6
  - unread records 12-7
  - use of 8-13
- \*.ddname
  - in DCB parameter 12-31
  - in DSNAMES parameter 12-71
  - in VOLUME parameter 12-139
- \*.label
  - in CNTL parameter 12-23
- \*.name
  - in OUTPUT parameter 12-104
- \*.stepname.ddname
  - in DCB parameter 12-32
  - in DSNAMES parameter 12-71
  - in PGM parameter 11-21
  - in VOLUME parameter 12-139
- \*.stepname.label 12-23
  - in CNTL parameter 12-23
- \*.stepname.name 12-104
  - in OUTPUT parameter 12-104
- \*.stepname.procstepname.ddname 12-104
  - in DCB parameter 12-31
  - in DSNAMES parameter 12-71
  - in OUTPUT parameter 12-104
  - in PGM parameter 11-21
  - in VOLUME parameter 12-139
- \*.stepname.procstepname.label 12-23
  - in CNTL parameter 12-23
- \*\*\* on output listing 9-13
- [] brackets, use 2-2
- /\*
  - for delimiter statement 15-7
  - for internal reader 7-52
- /\*DEL 7-53
- /\*EOF 7-52
- /\*JOBPARM Statement
  - description 16-4
  - example 16-8
  - introduction 1-3
  - overrides 16-7
  - parameter definition 16-4
  - syntax 16-4
- /\*MESSAGE Statement
  - description 16-9
  - example 16-9
- /\*NETACCT Statement
  - description 16-10
  - example 16-10
  - introduction 1-3
  - parameter definition 16-10
  - use of 3-4
- /\*NOTIFY Statement
  - description 16-11
  - examples 16-12
  - introduction 1-3
  - overrides 16-11
  - parameter definition 16-11
  - syntax 16-11
  - use of 3-18
- /\*OUTPUT Statement
  - description 16-13
  - example 16-21
  - example of job and step-level statements 14-1
  - parameter definition 16-14
  - syntax 16-14
  - use of 7-44-7-62

# Index

- /\*PRIORITY** statement
  - description 16-22
  - example 16-23
  - introduction 1-3
  - job selection 5-1
  - parameter definition 16-22
  - use of 5-20
  - using 5-1
- /\*PURGE** 7-53
- /\*ROUTE** statement
  - description 16-24
  - examples 16-26
  - introduction 1-4
  - parameter definition 16-24
  - restriction for RMT 16-25
  - routing output 3-8
  - syntax 16-24
- /\*SCAN** 7-53
- /\*SETUP** statement
  - description 16-28
  - example 16-28
  - parameter definition 16-28
  - use of 3-24
- /\*SIGNOFF** statement
  - for JES2 16-29
    - description 16-29
    - example 16-29
  - for JES3 17-49
    - description 17-49
    - use of 3-23
  - use of 3-21
- /\*SIGNON** statement
  - for JES2 16-30
    - description 16-30
    - examples 16-31
  - for JES3 17-50
    - description 17-50
    - example 17-51
    - parameter definition 17-50
    - syntax 17-50
    - use of 3-22
  - use of 3-21
- /\*XEQ** statement
  - description 16-32
  - example 16-32
  - introduction 1-4
  - parameter definition 16-32
- /\*XMIT** statement
  - description 16-33
  - introduction 1-4
  - parameter definition 16-33
- /\*DATASET** statement
  - description 17-5
  - examples 17-6
  - for binary input 3-26
  - introduction 1-5
  - parameter definition 17-5
  - syntax 17-5
- /\*ENDDATASET** statement
  - description 17-7
  - example 17-7
  - introduction 1-5
- /\*ENDPROCESS** statement
  - description 17-8
  - example 17-8
- ABCMP parameter on JES3 **/\*NET** statement 17-35
- ABEND dumps
  - See abnormal termination dump
- ABNORMAL parameter on JES3 **/\*NET** statement 17-36
  - use of 3-28
- abnormal termination dump
  - See also dump, abnormal termination
  - See also SYSABEND, SYSUDUMP, and SYSMDUMP DD statements
  - requesting 8-6
- absolute track technique for indexed sequential data sets (ISAM) 8-21
- ABSTR subparameter of SPACE parameter 12-115
  - See also SPACE parameter
  - use of to assign specific tracks 7-42
  - use with ISAM data sets 8-21
- accessing TCAM messages 12-111
- ACCODE parameter on DD statement
  - default 12-9
  - definition 12-9
  - description 12-9
  - examples 12-10
  - syntax 12-9
- accounting information for a network 3-4
  - See also NETACCT statement for JES2 or JES3
- accounting information for JES2 3-4
- accounting information on JOB statement 3-2
  - See also Accounting Information Parameter
- Accounting Information Parameter
  - JES2 format 10-4
    - example 10-6
    - subparameter definition 10-4-10-5
    - syntax 10-4
  - on JOB statement 10-3
    - examples 10-5
    - subparameter definitions 10-4
    - syntax 10-3
    - use of 3-2, 3-4
- ACCT parameter on EXEC statement
  - coding special characters 4-11
  - examples 11-5
  - subparameter definition 11-5
  - syntax 11-4
  - using 4-11
- ACCT parameter on JES3 **/\*NETACCT** statement 17-40
- ACMAIN parameter on JES3 **/\*MAIN** statement 17-24
- adding DD statements to a procedure 9-8
- adding members to a private library 8-2
- adding OUTPUT JCL statements to a procedure 9-10
- address subparameter of SPACE parameter

- assigning specific tracks 7-42
- description 12-115
- address, unit
  - on UNIT parameter 12-130
  - specifying unit information 7-28
- ADDRSPC Parameter
  - See also REGION Parameter
  - on EXEC statement 11-6
    - examples 11-7
    - override 11-6
    - relationship to REGION parameter 11-6
    - subparameter definition 11-6
    - syntax 11-6
    - use of 5-24
    - use of with cataloged procedures 5-24
  - on JOB statement 10-7
    - examples 10-8
    - subparameter definition 10-7
    - syntax 10-7
    - use of 5-24
  - requesting storage 5-23
- AFF subparameter of UNIT parameter
  - description 12-132
  - requesting unit affinity 7-32
- affinity between units and volumes 7-32
  - See also unit and volume affinity
- AL subparameter of LABEL parameter
  - ISO/ANSI Version 1 labels 12-90
  - ISO/ANSI/FIPS Version 3 labels 12-90
- ALIGN subparameter of FCB parameter 12-78
- alignment of forms 12-78
- ALL subparameter on JES3 **//\*MAIN** statement 17-27
  - See also FETCH parameter on JES3 **//\*MAIN** statement
- allocating data sets dynamically 4-12, 6-3
  - See also DYNAMNBR parameter
- allocation
  - absolute track 8-21
  - devices eligible 7-24
  - devices, for JES3
    - See also SETUP parameter on JES3 **//\*MAIN** statement
    - See also types of JES3 setup
    - for existing data sets 6-4
    - for new nonspecific data sets 6-4
  - dynamic 4-12, 6-3
    - See also DYNAM parameter on DD statement
    - See also DYNAMNBR parameter
  - nonspecific 8-21
    - See also SPACE parameter
- allocation messages
  - JES3 3-17
  - MVS 3-14
- allocation/termination messages
  - See allocation messages
- allowing for changes in cataloged and in-stream procedures 9-3
- alphanumeric character sets 2-10, 12-128, 14-54
- ALX subparameter of SPACE parameter 12-115
  - See also SPACE parameter
- AMORG subparameter of AMP parameter 12-12
- AMP parameter on DD statement
  - description 12-11
  - examples 12-15
  - invalid DD names 12-15
  - relationship to other parameters 12-14
  - subparameter definition 12-12-12-14
  - syntax 12-11
  - use of VSAM data sets 8-14
- ampersand
  - as special character 2-1
  - identifying a symbolic parameter 2-6
  - identifying temporary data set 2-10, 7-5
- AN character set
  - for 1403 12-128, 14-54
  - for 3203 Model 5 12-128, 14-54
- ANY subparameter (JES2)
  - See SYSAFF parameter on **/\*JOBPARM** statement
- ANYLOCAL subparameter
  - of DEST parameter
    - on DD statement 12-54
- ANYLOCAL subparameter of DEST parameter
  - on JES3 **//\*FORMAT PR** statement 17-13
  - on JES3 **//\*FORMAT PU** statement 17-20
- APG
  - See assigning a dispatching priority
- apostrophes
  - use of with special characters 2-11
  - use of with the DSNAME parameter 7-7
- area arrangement of an indexed sequential data set 8-22
- areas of a temporary indexed sequential data set
  - See temporary data set
- areas of indexed sequential data set 8-18
  - See also indexed sequential data set
- ASCII magnetic tape
  - See DCB parameter on DD statement
  - See LABEL parameter on DD statement
- ASCII, requesting translation 7-9
- assigning a dispatching priority
  - See also DPRTY parameter on EXEC statement to job steps 5-21
- assigning a job to a job class
  - in JES2 5-19
  - in JES3 5-19
- assigning a priority to a job
  - in JES2 5-20
  - in JES3 5-20
- assigning a value to a symbolic parameter 2-17
- assigning default values to symbolic parameters 2-16
- assigning specific tracks 7-42
- assigning system output data sets to output classes 7-51
- assigning values to and nullifying symbolic parameters 2-16
- associated data sets 7-6
  - See also DSID parameter on DD statement
- attributes, DCB
  - See DCB parameter on DD statement

# Index

AUL subparameter of LABEL parameter (ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels and user labels) 12-91

automatic priority group

See assigning a dispatching priority

automatic restart

See also RD Parameter

for JES2 5-27

for JES3 5-27

automatic step restart

See automatic restart

average block length space request

See blocklength subparameter of SPACE parameter

All character set (3211) 12-128, 14-54

B parameter (JES2)

See BURST parameter on /\*JOBPARM statement

See BURST parameter on /\*OUTPUT statement

b use

See blank, use in control statements

backward references

copying a data set name 7-6

DCB parameter 12-31

DDNAME parameter 12-49

DSNAME parameter 12-68

DUMMY parameter 12-73

introduction 2-13

PGM parameter 11-21

to VIO data sets 8-11

VOLUME parameter 12-135

basic direct access method

See BDAM subparameters of DCB parameter

basic indexed sequential access method

See BISAM subparameters of DCB parameter

basic partitioned access method

See BPAM subparameters of DCB parameter

basic sequential access method

See BSAM subparameters of DCB parameter

basic telecommunications access method

See BTAM subparameters of DCB parameter

BDAM subparameters of DCB parameter 12-36-12-48

BFALN subparameter on DCB parameter 12-36

BFTEK subparameter on DCB parameter 12-36

binary input

See reading column binary input

BISAM subparameters of DCB parameter 12-36-12-48

blank, use in control statements 2-10

BLDG parameter on JES3 /\*NETACCT statement 17-40

BLKSIZE subparameter on DCB parameter 12-36

coded with \* parameter 12-6

coded with DATA parameter 12-29

using with DDNAME parameter 12-49

with QNAME parameter 12-111

blocklength subparameter of SPACE parameter

See also SPACE parameter

description 12-113

use of 7-41

blocks, directory

See directory space

blocksize

See BLKSIZE subparameter on DCB parameter

BLP subparameter of LABEL parameter 12-91

BPAM subparameters of DCB parameter 12-36-12-48

braces in control statements 2-2

brackets in control statements 2-2

BSAM subparameters of DCB parameter 12-36-12-48

BSC communications link

See point-to-point communications link

BSC RJE for JES2 3-20

BTAM subparameters of DCB parameter 12-36-12-48

buffers

boundary

See BFALN subparameter on DCB parameter

for all lines

See BUFSIZE subparameter on DCB parameter

for one line

See BUFMAX subparameter on DCB parameter

for receiving operation

See BUFIN subparameter on DCB parameter

for sending operation

See BUFOUT subparameter on DCB parameter

length of

See BUFL subparameter on DCB parameter

number of

See BUFNO subparameter on DCB parameter

offset

See BUFOFF subparameter on DCB parameter

type

See BFTEK subparameter on DCB parameter

BUFIN subparameter on DCB parameter 12-36

BUFL subparameter on DCB parameter 12-36

with QNAME parameter 12-111

BUFMAX subparameter on DCB parameter 12-36

BUFND subparameter of AMP parameter 12-12

BUFNI subparameter of AMP parameter 12-12

BUFNO subparameter on DCB parameter 12-37

coded with \* parameter 12-6

coded with DATA parameter 12-29

using with DDNAME parameter 12-49

BUFOFF subparameter on DCB parameter 12-37

BUFOUT subparameter on DCB parameter 12-37

BUFSIZE subparameter on DCB parameter 12-37

BUFSP subparameter of AMP parameter 12-12

BURST parameter

on /\*JOBPARM Statement 16-4

on /\*OUTPUT statement 16-14

on DD statement 12-16

default 12-16

example 12-17

relationship to JES 12-17

relationship to other parameters 12-17

subparameter definition 12-16

syntax 12-16

on OUTPUT JCL statement 14-6

BURST Parameter on OUTPUT JCL statement

bursting of output 7-62

bypassing data set allocation

See defining a dummy data set

- bypassing disposition processing 7-19
- bypassing I/O operations
  - See DUMMY parameter on DD statement
- bypassing job initiation 3-25
- bypassing job steps
  - See COND Parameter
- bypassing label processing
  - See BLP subparameter of LABEL parameter
- bytes
  - per cylinder 18-10
  - per device 18-10
  - per record with keys 18-12
  - per record without keys 18-11
  - per track 18-10
- BYTES parameter
  - on /\*JOBPARM statement 16-5
- BYTES parameter on JES3 /\*MAIN statement
  - description 17-24
  
- C operator command (JES3)
  - See CANCEL operator command
- C parameter (JES2)
  - See CARDS parameter on /\*JOBPARM statement
  - See FCB parameter on /\*OUTPUT statement
- C subparameter of MODE parameter 17-5
- C subparameter of STACKER parameter (JES3) 17-16
- C subparameter on JES3 /\*MAIN statement
  - See CANCEL subparameter on JES3 /\*MAIN statement
- calculating dispatching priority 5-21
- CALL operator command (JES3) 17-3
- CANCEL operator command
  - for JCL 15-2
  - for JES3 17-3
- CANCEL subparameter on JES3 /\*MAIN statement
  - 17-25, 17-27, 17-29
  - See also CARDS parameter
  - See also FAILURE parameter on JES3 /\*MAIN statement
- capacities of direct access devices 18-10
- card forms (JES3)
  - See FORMS parameter on /\*FORMAT PU statement (JES3)
- CARDS parameter
  - on /\*JOBPARM statement 16-5
  - on JES3 /\*MAIN statement 17-24
- CARRIAGE parameter on JES3 /\*FORMAT PR statement 17-10
- carriage-tape-name subparameter on JES3 /\*FORMAT PR statement 17-10
- cataloged and in-stream procedures
  - See also JOBLIB DD statement
  - calling 9-1, 9-3
  - changes, allowing for 9-3
  - DD statement parameters, modifying 9-6
  - DD statements, adding 9-8
  - definition 9-1
  - EXEC statement parameters, modifying 9-4
  - identifying (in-stream procedure) 9-2
  - identifying on an output listing 9-13
  - introduction 1-7
  - modifying 9-4
  - OUTPUT JCL statement parameters, modifying 9-9
  - OUTPUT JCL statements, adding 9-10
  - passing information 4-8
  - placing in a procedure library 9-2
  - restrictions on contents of 9-1
  - selecting a cataloged procedure 4-4
  - symbolic parameters with 9-2
  - symbolic parameters, example 2-15, 2-21
  - symbolic parameters, use 2-15-2-21
  - using 9-3
  - writing 9-1
- cataloged procedure library, selecting 4-7
  - See also PROC parameter, on EXEC statement
- cataloging a data set 7-16
- CATLG subparameter of DISP parameter
  - description 12-59
  - use of 7-14
- caution concerning leading and trailing commas 2-19
- cautions, for concatenated data sets 2-9
- cccc subparameter (JES2)
  - See SYSAFF parameter on /\*JOBPARM statement
- changing a user's RACF password 10-21
- channel program, active
  - requesting real storage 5-23
- CHAR Parameter on OUTPUT JCL statement
- character arrangement tables 12-95
  - See also CHARS parameter
  - See also MODIFY parameter
- character arrangements, 3800 printing subsystem, requesting 7-60
- character control, requesting 7-58
- character set codes 12-128, 14-54
- character sets 2-10
- character sets, requesting 7-59
  - See also CHARS parameter
- character-set-code subparameter of UCS parameter 12-127
- CHARS parameter
  - on /\*OUTPUT statement 16-14
  - on DD statement 12-18
  - description 12-18
  - examples 12-20
  - high-density dumps on OUTPUT JCL statement 14-8
  - relationship to JES 12-20
  - relationship to other parameters 12-19
  - relationships 12-79
  - subparameter definition 12-18
  - syntax 12-18
  - on JES3 /\*FORMAT PR statement 17-11
  - on OUTPUT JCL statement 14-7
  - use of 7-60
- checkid subparameter of RESTART parameter 10-34
- checkpoint data set 5-26
  - See also SYSCHK DD statement
  - See also SYSCKEOV DD statement

# Index

- checkpoint restart 5-26
  - See also RD Parameter
  - See also SYSCHK DD statement
  - See also SYSCKEOV DD statement
  - for generation data sets 5-28
- checkpoint/restart facility 5-26
  - See also RD Parameter
  - See also RESTART Parameter on JOB statement
  - See also SYSCHK DD statement
  - See also SYSCKEOV DD statement
- CHKPT macro instruction 5-26
  - See also CHKPT parameter on DD statement
  - use of 5-26
  - with RD parameter 10-28
- CHKPT parameter on DD statement
  - See also CHKPT macro instruction
  - description 12-21
  - overrides 12-21
  - relationship to other parameters 12-21
  - subparameter definition 12-21
  - syntax 12-21
- CHNGDUMP operator command (JCL) 15-2
- CHNSIZE parameter
  - on JES3 **/\*FORMAT PR** statement 17-11
  - on JES3 **/\*FORMAT PU** statement 17-19
- CKPTLINE Parameter on OUTPUT JCL statement
- CKPTLNS parameter
  - on **/\*OUTPUT** statement 16-14
- CKPTPAGE Parameter on OUTPUT JCL statement
- CKPTPGS parameter
  - on **/\*OUTPUT** statement 16-15
- CKPTSEC Parameter on OUTPUT JCL statement
- class
  - See CLASS Parameter
- CLASS initialization statement (JES3) 5-4
- CLASS Parameter
  - on JES3 **/\*DATASET** statement 17-6
  - on JES3 **/\*MAIN** statement 17-25
    - use of 5-19
  - on JOB statement 10-9
    - defaults 10-9
    - example 10-9
    - override 10-9
    - subparameter definition 10-9
    - syntax 10-9
  - on OUTPUT JCL statement 14-13
  - use of 5-1
- CLASS Parameter on OUTPUT JCL statement
- class-name subparameter
  - in SYSOUT parameter 12-121
  - on JES3 **/\*MAIN** statement 17-25
  - See also CLASS Parameter
- class-name subparameter of MSGCLASS parameter
  - description 10-14
  - use of 3-14
- CLOSE macro instruction
  - LEAVE option 12-85
  - REREAD option 12-85
  - with SPACE parameter 12-114
- CLOSE subparameter of FREE parameter
  - description 12-84
  - use with dynamic allocation 4-12, 6-3
- CNTL parameter
  - examples 12-23
  - subparameter definition 12-23
  - syntax 12-23
- CNTL Statement
  - description 15-5
  - example 15-6
  - syntax 15-5
- code field in **/\*OUTPUT** statement 16-14
- CODE subparameter on DCB parameter 12-37
- code-name subparameter
  - in SYSOUT parameter 12-121
- coding
  - consecutive periods in control statements 2-3, 2-18
  - in DLM parameter 12-64
  - in DSNAME parameter 12-68
  - JES2 statements 2-8
  - JES3 statements 2-8
  - special characters 2-10
  - symbolic parameters 2-6
- coding conventions
  - considerations for network jobs 3-6
  - for JCL statements 2-1
  - for JES2 control statements 2-8
  - for JES3 control statements 2-8
  - guide to using job and step control statements 2-23
  - the DUMMY parameter 8-8
- Coding JES2 Control Statements
  - /\*JOBPARM** Statement 16-4
  - /\*MESSAGE** Statement 16-9
  - /\*NETACCT** Statement 16-10
  - /\*NOTIFY** Statement 16-11
  - /\*OUTPUT** Statement 16-13
  - /\*PRIORITY** statement 16-22
  - /\*ROUTE** statement 16-24
  - /\*SETUP** statement 16-28
  - /\*SIGNOFF** statement 16-29
  - /\*SIGNON** statement 16-30
  - /\*XEQ** statement 16-32
  - /\*XMIT** statement 16-33
  - coding conventions 2-8
  - Command statement 16-2
  - general rules for coding 16-1
  - introduction 1-3
- Coding JES3 Control Statements
  - /\*SIGNOFF** statement 17-49
  - /\*SIGNON** statement 17-50
  - /\*DATASET** statement 17-5
  - /\*ENDDATASET** statement 17-7
  - /\*ENDPROCESS** statement 17-8
  - /\*FORMAT PR** statement 17-9
  - /\*FORMAT PU** statement 17-18
  - /\*MAIN** statement 17-23
  - /\*NET** statement 17-35
  - /\*NETACCT** statement 17-40
  - /\*OPERATOR** statement 17-42
  - /\*PAUSE** statement 17-43
  - /\*PROCESS** statement 17-44

- coding conventions 2-8
- Command statement 17-3
- general rules for coding 17-1
- introduction 1-5
- XEQ statement 17-47
- coding special DD statements
  - description 13-1
  - general syntax 13-1
- coding the DD statement
  - description 12-1
  - examples of ddnames 12-5
  - general syntax 12-1
  - input stream data 8-13
  - introduction 1-2
  - maximum per job 12-1
  - OUTPUT parameter 12-103
  - parameters, keyword 12-3
    - ACCODE parameter 12-9
    - AMP parameter 12-11
    - BURST parameter 12-16
    - CHARS parameter 12-18
    - CNTL parameter 12-23
    - COPIES parameter 12-25
    - DCB parameter 12-31
    - DDNAME parameter 12-49
    - DEST parameter 12-53
    - DLM parameter 12-64
    - DSID parameter 12-66
    - DSNAME parameter 12-68
    - FCB parameter 12-77
    - FLASH parameter 12-81
    - FREE parameter 12-84
    - HOLD parameter 12-87
    - LABEL parameter 12-89
    - MODIFY parameter 12-95
    - MSVGP parameter 12-98
    - OUTLIM parameter 12-101
    - PROTECT parameter 12-109
    - QNAME parameter 12-111
    - SPACE parameter 12-112
    - SUBSYS parameter 12-117
    - SYSOUT parameter 12-120
    - TERM parameter 12-125
    - UCS parameter 12-127
    - UNIT parameter 12-130
    - VOLUME parameter 12-135
  - parameters, positional 12-3
    - \* parameter 12-6
    - DATA parameter 12-28
    - DUMMY parameter 12-73
    - DYNAM parameter 12-76
  - use of 6-9
- coding the EXEC statement
  - cataloged procedure, use with 9-1
  - description 11-1, 18-14
  - examples 11-3
  - introduction 1-2
  - modifying parameters on 9-4
  - name field 11-1
  - parameters, keyword 11-2
    - ACCT parameter 11-4
    - ADDRSPC parameter 11-6
    - COND parameter 11-8
    - DPRTY parameter 11-13
    - DYNAMNBR parameter 11-15
    - PARM parameter 11-17
    - PERFORM parameter 11-19
    - RD parameter 11-26
    - REGION parameter 11-30
    - TIME parameter 11-32
  - parameters, positional 11-2
    - PGM parameter 11-21
    - PROC parameter 11-24
    - Procedure name parameter 11-24
  - restriction on changing PGM parameter 9-4
  - using, example of 4-1
- Coding the JOB statement
  - description 10-1
  - examples of 10-2
  - parameters, keyword 10-2
    - ADDRSPC Parameter 10-7
    - CLASS Parameter 10-9
    - COND Parameter 10-10
    - GROUP Parameter 10-12
    - MSGCLASS Parameter 10-14
    - MSGLEVEL Parameter 10-16
    - NOTIFY Parameter 10-18
    - PASSWORD Parameter 10-20
    - PERFORM Parameter 10-22
    - PRTY Parameter 10-26
    - RD Parameter 10-28
    - REGION Parameter 10-31
    - RESTART Parameter 10-33
    - TIME Parameter 10-36
    - TYPRUN Parameter 10-39
    - USER Parameter 10-41
  - parameters, positional 10-2
    - accounting information parameter 10-3
    - Programmer's Name Parameter 10-24
- coding the OUTPUT JCL statement
  - description 14-1
  - introduction 1-2
  - overrides 14-4
  - relationship to `//*FORMAT` 14-5
  - relationship to the SYSOUT parameter 14-4
  - syntax 14-2
- column binary input, reading 3-26
- comma
  - continuing control statements 2-6
  - how to code 2-1, 2-2
  - leading and trailing commas, caution 2-19
  - use with symbolic parameters 2-19
- COMMAND Statement
  - for JCL 15-2
    - description 15-2
    - example 15-3
    - syntax 15-2
  - for JES2 16-2
    - description 16-2
    - examples 16-3

# Index

- parameter definition 16-2
- for JES3 17-3
  - commands, list of 17-3
  - description 17-3
  - examples 17-4
  - syntax 17-3
- COMMENT JCL Control Statement
  - description 15-4
  - example 15-4
  - introduction 1-1
  - relationship to MSGLEVEL parameter 15-4
- comments field in control statements 2-3
- comments field, continuing 2-7
- communications link, BSC
  - See point-to-point communications link
- COMPACT parameter
  - on /\*OUTPUT statement 16-15
  - on JES3 /\*FORMAT PR statement 17-12
  - on JES3 /\*FORMAT PU statement 17-20
  - on OUTPUT JCL statement 14-15
- COMPACT Parameter on OUTPUT JCL statement
- compaction-table-name subparameter 17-12
  - on JES3 /\*FORMAT PR statement 17-12
  - on JES3 /\*FORMAT PU statement 17-20
- completion codes
  - See COND Parameter
- concatenating data sets
  - cautions 2-9
    - with DUMMY parameter 2-9
    - with RDJFCB macro 2-9
  - introduction 2-9
- concatenating private libraries 8-4
- concatenation cautions 2-9
- concurrent use, data set 7-19
- COND Parameter
  - on EXEC statement 11-8
    - Cautions when using 11-10
    - examples 5-9, 5-13, 11-12
    - override 11-10
    - Subparameter definition 11-9
    - syntax 11-8
    - use of 5-7
    - use of with cataloged procedures 5-13
    - use of, to force step execution 5-15
  - on JOB statement 10-10
    - examples 5-9, 10-11
    - subparameter definition 10-10
    - syntax 10-10
    - use of 5-5
- conditional disposition of data sets 7-13
  - See also DISP parameter on DD statement
- conditionally executing job steps 5-5
  - See also COND Parameter
  - JES3 setup, for 5-5
  - testing return codes 5-5
- CONTIG subparameter of SPACE parameter
  - See also contiguous space
  - See also SPACE parameter
  - description 12-114
  - use of 8-21
- contiguous space
  - See also CONTIG subparameter of SPACE parameter
  - See also SPACE parameter
  - for indexed sequential data sets 8-21
  - requesting 7-39
- continuing comments 2-7
- continuing control statements 2-6
- continuing parameters 2-7
- continuing the comments field 2-7
- continuing the parameter field 2-7
- control of nontemporary data set
  - exclusive control 7-19
  - shared control 7-19
- CONTROL parameter on JES3 /\*FORMAT PR statement 17-12
- CONTROL Parameter on OUTPUT JCL statement
- control printing
  - bursting of output 7-62
  - copy modification 7-58
  - delaying 7-55
  - forms control 7-60
  - forms overflow processing (JES3) 7-56
  - forms overlay 7-62
  - limiting output records (JES2) 7-55
  - limiting output records (JES3) 7-56
  - multiple copies (JES2) 7-57
  - multiple copies (JES3) 7-58
  - page overflow processing (JES2) 7-56
  - printer form and character control 7-58
  - printer spacing (JES3) 7-56
  - special character sets 7-59
  - suppressing 7-55
- control program (JES3) 5-4
  - See also TYPE parameter on JES3 /\*MAIN statement
- control statement fields
  - comments field 2-4
  - identifier field 2-3
  - name field 2-4
  - operation field 2-4
  - parameter field 2-4
- control statements
  - See specific control statement
- controlling job execution node using JES3
  - networking 3-11
- controlling output destination (JES2)
  - See controlling output destination in a JES2 network
- controlling output destination in a JES2 network 3-7
- controlling output destination using JES3 3-12
- COPIES parameter
  - on /\*JOBPARM statement 16-5
  - on /\*OUTPUT statement 16-15
  - on JES3 /\*FORMAT PR statement 17-12
  - on JES3 /\*FORMAT PU statement 17-20
  - on SYSOUT DD statement 12-25
    - default 12-26
    - description 12-25
    - examples 12-27
    - relationship to JES 12-27



- relationship to other parameters 12-26
- restriction with UNIT parameter 12-27
- subparameter definition 12-25
- syntax 12-25
- requesting multiple copies (JES2) 7-57
- requesting multiple copies (JES3) 7-58
- COPIES Parameter on OUTPUT JCL statement
- copy input deck and bypass job initiation 3-25
- copy modification module 7-58
  - See also MODIFY parameter
- copy modification, requesting 7-58
  - See also MODIFY parameter
- COPY subparameter of TYPRUN parameter 10-40
- COPYG parameter
  - on /\*OUTPUT statement 16-16
- copying JCL input without execution in JES2 3-25
- copying the data set name from an earlier DD statement 7-6
- count subparameter of FLASH parameter
  - on /\*OUTPUT statement 16-18
  - on DD statement 12-81
  - on JES3 /\*FORMAT PR statement 17-14
- CPRI subparameter on DCB parameter 12-37
  - with OUTLIM parameter 12-101
- CPU time limit
  - See TIME Parameter
- creating
  - a generation data set 8-26
  - a nontemporary data set 7-3
  - a private library 8-2
  - a temporary data set
    - See temporary data set
  - a temporary partitioned data set
    - See temporary data set
  - an indexed sequential data set 8-18
  - and retrieving indexed sequential data sets 8-18
  - and using private and temporary libraries 8-1
  - areas of a temporary indexed sequential data set
    - See temporary data set
  - or retrieving a nontemporary data set 7-3
- creating a model data set label 8-26
- Creating and Using a Subsystem Data Set 8-32
- CROPS subparameter of AMP parameter 12-12
- CYL subparameter of SPACE parameter
  - See also SPACE parameter
  - description 12-113
  - requesting space 7-39
- cylinder index
  - See NTM subparameter on DCB parameter
- cylinders
  - See CYL subparameter of SPACE parameter
- cylinders, number per device 18-10
- CYLOFL subparameter on DCB parameter 12-38
  - using with FCB parameter 12-78
  - with UCS parameter 12-129
- D operator command (JES3)
  - See DELAY operator command (JES3)
- D parameter (JES2)
  - See DEST parameter on /\*OUTPUT statement
- D subparameter on JES3 /\*MAIN statement
  - See DUMP subparameter on MAIN statement
- data
  - identifying 6-9-7-43
  - modifying 7-58
- data control block
  - See DCB parameter on DD statement
- data definition (DD) statement
  - See coding the DD statement
- data link, synchronous
  - See synchronous data link
- DATA parameter on DD statement
  - See also \* DD parameter
  - See also DLM parameter
  - creating a data set 18-2-18-3
  - data in input stream 8-13
  - description 12-28
  - examples 12-29
  - relationship to other parameters 12-28
  - restrictions 12-28
  - retrieving a data set 18-4
  - separating groups of data 12-29
  - syntax 12-28
  - use of 8-13
- data set
  - areas of a temporary indexed sequential
    - See temporary data set
  - backward references to VIO 8-11
  - cataloging 7-16
  - creating 18-2-18-3
  - defining a dummy 8-8
  - defining a temporary VIO 8-10
  - delaying writing of 7-55
  - deleting 7-15
  - disposition processing 7-13-7-19
  - disposition, specifying 7-14
  - dummy, requests to read or write 8-9
  - exclusive control of 7-20
  - extending 18-5
  - for mass storage volumes 7-39
  - identifying to system 6-9
  - indexed sequential, creating and retrieving 8-18
  - indexed sequential, requesting index space 7-42
  - information, specifying 7-1
  - integrity processing, how the system performs 7-20
  - integrity, insuring 7-19
  - keeping 7-16
  - members of a temporary partitioned
    - See temporary data set
  - multivolume 7-28
  - name, copying 7-6
  - nontemporary, creating 7-3
  - nontemporary, retrieving 7-3
  - output, assigning to output classes 7-51
  - output, delaying the writing of 7-55
  - output, processing for the job 7-44
  - output, suppressing the writing of 7-55
  - partitioned, requesting directory space 7-42
  - passed unreceived

# Index

- See passed unreceived data sets
- passing 7-17
- processing options, specifying 7-43
- requesting space for non-VSAM 7-39
- retrieving 18-4
- shared control of 7-20
- special 8-1
- specifying for MSS 7-37
- status 12-57
- status, specifying 7-14
- suppressing the writing of 7-55
- temporary
  - See temporary data set
  - temporary, creating 7-4
  - temporary, retrieving 7-4
  - temporary, using VIO for 8-10
  - temporary, using VIO to pass among job steps 8-12
  - uncataloging 7-17
  - VSAM 8-14
  - writing output 7-44
- data set label
  - completing the data control block 12-33
  - copying attributes from 12-34
  - model 8-26
- data set name
  - See also DSNNAME parameter on DD statement
  - copying 7-6
  - in apostrophes 7-7
  - special characters, rules of 7-7
- data set organization
  - See DSORG subparameter on DCB parameter
- data set status
  - See also DISP parameter on DD statement
  - use of 7-14
- data-set-sequence-number subparameter of LABEL parameter
  - description 12-89
  - use of 7-8
- DCB macro instruction
  - coding the DCB parameter 12-31
  - requesting exclusive control of part of a data set 7-20
- DCB macro instruction program written in assembler language 12-33
- DCB parameter on DD statement
  - BDAM subparameters 12-36-12-48
  - BISAM subparameters 12-36-12-48
  - BPAM subparameters 12-36-12-48
  - BSAM subparameters 12-36-12-48
  - BTAM subparameters 12-36-12-48
  - completing the data control block 12-33
  - copying DCB information 12-34
    - from a data set label 12-34
    - from an earlier DD statement 12-34
  - description 12-31
  - examples 12-35
  - EXCP subparameters 12-36-12-48
  - for dummy data set 8-8
  - for private libraries 8-3
  - GAM subparameters 12-36-12-48
  - modifying in cataloged or in-stream procedures 9-4
  - QISAM subparameters 12-36-12-48
  - QSAM subparameters 12-36-12-48
  - relationship to other parameters 12-33
  - subparameter definition 12-32
  - supplying DCB subparameters 12-33
  - syntax 12-31
  - TCAM subparameters 12-36-12-48
  - use of with indexed sequential data sets 8-20
  - use with generation data sets 8-27, 8-30
- DDNAME parameter 7-1
  - on DD statement 12-49
    - coding backward references 12-51
    - examples 12-51
    - relationship to other parameters 12-49
    - subparameter definition 12-49
    - syntax 12-49
  - on FORMAT PR statement (JES3) 17-10
  - on JES3 **//\*DATASET** statement 17-5
  - on JES3 **//\*FORMAT PU** statement 17-19
  - specifying 7-1
  - use of 7-1
- ddname subparameter
  - on JES3 **//\*FORMAT PR** statement 17-10
  - on JES3 **//\*FORMAT PU** statement 17-19
  - on UNIT parameter 12-132
- DEADLINE parameter on JES3 **//\*MAIN** statement 17-25
- deadline scheduling for JES3 3-27
  - See also DEADLINE parameter on JES3 **//\*MAIN** statement
- deallocating data sets dynamically 4-12, 6-3
  - See also FREE parameter on DD statement
- deallocation, dynamic
  - See dynamically allocating and deallocating data sets
  - See FREE parameter on DD statement
- dedicated devices
  - See also DEVPOOL parameter on JES3 **//\*NET** statement
  - statementdevrl
  - dependent job control 3-29
- default disposition processing 7-19
- default OUTPUT JCL statement, defined 14-3
- DEFAULT Parameter on OUTPUT JCL statement 14-20
- DEFER subparameter of UNIT parameter
  - description 12-132
  - use of 7-30
- deferred mounting of volumes, requesting 7-30, 12-132
  - See also DEFER subparameter of UNIT parameter
- deferred restart 5-27
- defining a dummy data set 8-8
- defining a VIO temporary data set 8-10
- defining symbolic parameters when writing a procedure 2-15
- DELAY operator command (JES3) 17-3
- delaying initiation of other jobs (JES3) 3-24
  - See also UPDATE parameter on JES3 **//\*MAIN** statement
- delaying initiation of your job in JES2 3-23

- See also `/*SETUP` statement
- See also TYPRUN Parameter, on JOB statement
- delaying job initiation (JES3) 3-24
- delaying the writing of an output data set 7-55
- DELETE subparameter of DISP parameter
  - description 12-59
  - use of 7-15
- deleting a data set
  - See DELETE subparameter of DISP parameter
- deleting records
  - exclusive control 7-20
- deleting unused space
  - See RLSE subparameter of SPACE parameter
- delimiter other than `/*`
  - See DLM parameter
- Delimiter Statement
  - See also `* DD` parameter
  - See also DATA parameter on DD statement
  - See also DLM parameter
  - description 15-7
  - example 15-7
  - input stream data 8-13
  - introduction 1-1
  - relationship to DLM parameter 15-7
- demounting a tape volume
  - See VOLUME parameter
- DEN subparameter on DCB parameter 12-38
- dependent job control for JES3: The Job Net
  - See also JES3 `/*NET` statement
  - defining a dependent job net 3-28
  - establishing dependencies between nets 3-29
  - examples 3-30-3-33
  - how to code 3-29
  - specifying dedicated devices 3-29
  - specifying early setup of resources 3-29
  - use of 3-27
- DEPT parameter on JES3 `/*NETACCT` statement 17-40
- DEST parameter
  - See also `/*ROUTE` statement
  - See also `/*XMIT` statement
  - See also controlling job execution node using JES3 networking
  - See also controlling output destination in a JES2 network
  - See also JES3 `/*ROUTE XEQ` statement
  - on `/*OUTPUT` statement 16-16
  - on DD statement 12-53
    - defaults 12-55
    - description 12-53
    - example 12-56
    - relationship to other parameters 12-55
    - subparameter definition for JES2 12-53
    - subparameter definition for JES3 12-54
    - syntax 12-53
  - on JES3 `/*FORMAT PR` statement 17-13
  - on JES3 `/*FORMAT PU` statement 17-20
  - on OUTPUT JCL statement 14-23
- DEST Parameter on OUTPUT JCL statement destination (JES2)
  - See DEST parameter destination (JES3)
  - See DEST parameter destination (TSO)
  - See DEST parameter destination (VM)
  - See DEST parameter device setup in JES3
  - See types of JES3 setup
- device-address subparameter of DEST parameter
  - on DD statement 12-55
- device-address subparameter of DEST parameter on JES3 `/*FORMAT PR` statement 17-13
- device-address subparameter of DEST parameter on JES3 `/*FORMAT PU` statement 17-21
- device-address subparameter of UNIT parameter
  - description 12-130
  - use of 7-28
- device-name subparameter of DEST parameter
  - on DD statement 12-54
  - on OUTPUT JCL statement 14-24
- device-name subparameter of DEST parameter on JES3 `/*FORMAT PR` statement 17-13
- device-name subparameter of DEST parameter on JES3 `/*FORMAT PU` statement 17-20
- device-type subparameter of UNIT parameter
  - description 12-130
  - use of 7-28
- devices
  - capacities 18-10
  - direct access 7-24
  - for dependent job control 3-27
  - JES3 setup 6-5
  - magnetic tape 7-24
  - specifying 7-52
  - valid for VSAM 8-15
- DEVPOOL parameter on JES3 `/*NET` statement
  - description 17-36
  - use of 3-29
- DEVRELSE parameter on JES3 `/*NET` statement
  - description 17-37
  - use of 3-29
- DHWS subparameter of SETUP parameter on JES3 `/*MAIN` statement 17-31
- DIAGNS subparameter on DCB parameter
  - description 12-38
  - using with DDNAME parameter 12-49
- direct access data sets
  - disposition processing 7-14
  - insuring integrity of 7-19
- direct access device 7-24
- direct access devices, capacities of 18-10
- direct access volumes
  - for partitioned data sets 7-42
  - for passed data sets 7-17
- directory space
  - See also SPACE parameter
  - requesting for a partitioned data set 7-42
- directory subparameter of SPACE parameter 12-114

# Index

- See also SPACE parameter
- DISABLE operator command (JES3) 17-3
- diskette reader writer 7-6, 12-66
  - See also associated data sets
  - See also DSID parameter on DD statement
- DISP parameter on DD statement
  - See also disposition processing
  - description 12-57
  - DISP = MOD for multivolume data set 12-61
  - examples 12-62
  - extending a multivolume data set 12-62
  - for private library 8-2
  - on STEPLIB DD statement 13-8
  - on SYSABEND DD statement 13-11
  - on SYSCHK DD statement 13-15
  - on SYSCKEOV DD statement 13-18
  - on SYSMDUMP DD statement 13-11
  - on SYSUDUMP DD statement 13-11
  - relationship to other parameters 12-60
  - subparameter definition 12-57
  - syntax 12-57
  - use of 7-13
  - use of with indexed sequential data sets 8-20
  - use with generation data sets 8-27, 8-29
- dispatching priority, assigning
  - See assigning a dispatching priority
- DISPLAY operator command (JCL) 15-2
- disposition processing
  - See also DISP parameter on DD statement
  - bypassing 7-19
  - conditional 7-13
  - default 7-19
  - examples of 7-23
  - of non-VSAM data sets 7-13
  - of passed unreceived data sets 7-18
- DJC
  - See dependent job control for JES3: The Job Net
- DLM parameter
  - See also \* DD parameter
  - See also DATA parameter on DD statement
  - See also Delimiter Statement
  - input stream data 8-13
  - on DD statement 12-64
    - default 12-65
    - description 12-64
    - example 12-65
    - relationship to other parameters 12-65
    - subparameter definition 12-64
    - syntax 12-64
  - on XMIT statement (JES2) 16-34
  - use of 8-13
- DOUBLE subparameter on JES3 **//\*FORMAT PR** statement 17-12
- DOUBLE subparameter on OUTPUT JCL statement 14-16
- DPRTY parameter on EXEC statement
  - calculating dispatching priority 5-21
  - default 11-14
  - examples 11-14
  - subparameter definition 11-13
  - syntax 11-13
  - use of 5-21
- DS subparameter of CHNSIZE parameter
  - on JES3 **//\*FORMAT PR** statement 17-11
  - on JES3 **//\*FORMAT PU** statement 17-20
- DSID parameter on DD statement
  - See also associated data sets
  - description 12-66
  - example 12-67
  - relationship to other parameters 12-67
  - subparameter definition 12-66
  - syntax 12-66
- DSN parameter on DD statement
  - See DSNNAME parameter on DD statement
- DSNAME parameter 7-2, 7-7
- DSNAME parameter on DD statement
  - copying 7-6
  - description 12-68
  - examples 12-71
  - for private library 8-2
  - for temporary data set 7-4
  - on STEPLIB DD statement 13-8
  - on SYSABEND DD statement 13-11
  - on SYSCHK DD statement 13-15
  - on SYSCKEOV DD statement 13-19
  - on SYSMDUMP DD statement 13-11
  - on SYSUDUMP DD statement 13-11
  - relationship to other parameters 12-71
  - specifying 7-2
  - specifying in apostrophes 7-7
  - subparameter definition 12-69
  - syntax 12-68
  - use for members of a partitioned data set 7-4
  - use for nontemporary data sets 7-3
  - use of 7-2
  - use of apostrophes 7-7
  - use of with indexed sequential data sets 8-19, 8-22, 8-23
  - use when creating private libraries 8-2
  - use when requesting dumps 8-6
  - use with generation data sets 7-4, 8-26, 8-28
  - use with indexed sequential data sets 8-22
- dsname subparameter of VOLUME parameter 12-138
- DSNAME = NULLFILE 8-9
- DSORG subparameter on DCB parameter
  - description 12-38
  - for indexed sequential data set 7-42
- DSP (JES3)
  - allowable on PROCESS statement 17-45
  - use 17-44
- dsp parameter on JES3 **//\*PAUSE** statement
  - examples 17-46
- dummy data set
  - bypassing disposition processing 7-19
  - bypassing I/O operations 7-55
  - defining 8-8
  - DSNAME = NULLFILE 8-9
  - requests to read 8-9
  - requests to write 8-9
  - suppressing output 7-55

- DUMMY parameter on DD statement
  - backward references 12-74
  - concatenated data sets 12-74
  - creating a dummy data set 8-8
  - description 12-73
  - examples 8-8, 12-74
  - parameters on DD DUMMY Statements 12-73
  - relationship to access methods 12-74
  - relationship to other parameters 12-74
  - suppressing output 7-55
  - syntax 12-73
  - use of 8-8
- dump requests
  - See abnormal termination dump
- DUMP subparameter on MAIN statement 17-25, 17-29
- DUMP value on CHARS parameter 14-8
- dump, abnormal termination
  - high-density 12-20, 12-79, 14-9, 14-27
  - high-density on OUTPUT JCL statement 14-8
  - printing 13-12
  - requesting 8-6
  - storing 13-11
  - to write more than one dump 13-12
  - when using IPCS 8-7
- DYNAM parameter on DD statement
  - See also dynamically allocating and deallocating data sets
  - description 12-76
  - dynamic allocation 4-12, 6-3
  - example 12-76
  - relationship to other parameters 12-76
  - syntax 12-76
- dynamic support program
  - See DSP (JES3)
- dynamically allocating and deallocating data sets 4-12, 6-3
- dynamically allocating data sets 4-12, 6-3
  - See also DYNAM parameter on DD statement
  - See also DYNAMNBR parameter
- dynamically deallocating data sets 4-12, 6-3
  - See also FREE parameter on DD statement
- DYNAMNBR parameter
  - See also dynamically allocating and deallocating data sets
  - default 11-15
  - description 11-15
  - example 11-16
  - subparameter definition 11-15
  - syntax 11-15
  - use of (dynamic allocation) 4-12, 6-3
- E operator command (JES3)
  - See ERASE operator command (JES3)
- E parameter (JES2)
  - See CKPTLNS parameter on /\*OUTPUT statement
  - See RESTART parameter on JOBPARM statement
- E subparameter of MODE parameter 17-5
- ellipsis in control statements 2-3
- ENABLE operator command (JES3) 17-3
- end program control
  - for JCL 15-8
- END subparameter of FREE parameter 12-84
  - See also dynamically allocating and deallocating data sets
- ENDCNTL Statement
  - description 15-8
  - example 15-8
  - syntax 15-8
- enqueueing on a data set 7-19
- entering data through the input stream 8-13
- ERASE operator command (JES3) 17-3
- EROPT subparameter on DCB parameter 12-39
- error option
  - See EROPT subparameter on DCB parameter
- esoteric name
  - See group-name subparameter of UNIT parameter
- establishing dependencies between different nets 3-29
- establishing job processing balance in JES3 5-19
- EVEN subparameter of COND parameter on EXEC statement
  - use of 5-8
- example of
  - an in-stream procedure containing symbolic parameters 2-21
  - coding the TIME parameter on JOB and EXEC statements 5-18
  - creating and retrieving an indexed sequential data set 8-24
  - dependent job control 3-30
  - disposition processing of non-VSAM data sets 7-23
  - dynamically deallocating data sets 4-13, 6-4
  - identifying data sets to the system 7-13
  - obtaining output (JES2) 3-9
  - obtaining output (JES3) 3-13
  - requesting space 7-43
  - requesting storage 5-26
  - requesting units and volumes 7-31
  - routing a job through a JES3 network 3-11
  - spool partitioning (JES3) 3-34
  - unit and volume affinities 7-35
  - unit and volume affinity 7-33
  - using the COND parameter in a job 5-9
  - using the internal reader 7-53
- Example of using the internal reader 7-53
- exclude particular processors (JES3) 5-3
- exclusive control of a data set 7-20
- exclusive OR 2-2
- EXCP (execute channel program) subparameters of DCB parameter 12-36-12-48
- execut channel program
  - See EXCP (execute channel program) subparameters of DCB parameter
- execute statement
  - See coding the EXEC statement
- Executing your job 1-7
- execution of job steps, conditional 5-5
  - See also conditionally executing job steps
- existing data sets

# Index

- default disposition processing 7-19
- volume request 7-24
- EXPDT subparameter of LABEL parameter
  - description 12-93
  - use of 7-12
  - when DELETE is coded 7-15
  - when KEEP is coded 7-16
- EXPDTCHK parameter on JES3 **//\*MAIN** statement 17-27
- expiration date
  - See EXPDT subparameter of LABEL parameter
- expiration date checking (JES3)
  - See EXPDTCHK parameter on JES3 **//\*MAIN** statement
- explicit setup
  - See types of JES3 setup
- explicit unit affinity 7-33
- extending a data set
  - additional space 7-41
  - more than one unit 7-29
  - parameters for 18-5
- external page storage
  - See virtual storage
- external writer 7-54
  - See also EXTWTR parameter
- EXTWTR parameter
  - on JES3 **//\*FORMAT PR** statement 17-14
  - on JES3 **//\*FORMAT PU** statement 17-21
- F operator command (JES3)
  - See MODIFY operator command
- F parameter (JES2)
  - See FORMS parameter on **/\*JOBPARM** statement
  - See FORMS parameter on **/\*OUTPUT** statement (JES2)
- FAIL operator command (JES3) 17-3
- FAILURE parameter on JES3 **//\*MAIN** statement
  - description 17-27
  - introduction 5-29
- FCB parameter
  - forms control, requesting 7-60
  - on **/\*OUTPUT** statement 16-17
  - on DD statement 12-77
    - defaults 12-78
    - description 12-77
    - relationship to other parameters 12-78
    - subparameter definition 12-77
    - syntax 12-77
  - on JES3 **//\*FORMAT PR** statement 17-14
  - on OUTPUT JCL statement 14-26
- FCB Parameter on OUTPUT JCL statement
- fcv-name subparameter of FCB parameter 12-77, 14-26
- FETCH parameter on JES3 **//\*MAIN** statement
  - description 17-27
  - use of 3-14
- fetching messages
  - See FETCH parameter on JES3 **//\*MAIN** statement
- fields in control statements 2-3
- fixed length record
  - See RECFM subparameter, on DCB parameter
- FLASH parameter
  - forms overlay, requesting 7-62
  - on **/\*OUTPUT** statement 16-18
  - on DD statement 12-81
    - defaults 12-81
    - description 12-81
    - example 12-83
    - relationship to other parameters 12-82
    - subparameter definition 12-81
    - syntax 12-81
  - on JES3 **//\*FORMAT PR** statement 17-14
- FLASH Parameter on OUTPUT JCL statement
- FLASHC parameter
  - on **/\*OUTPUT** statement 16-18
- FOLD subparameter of UCS parameter 12-127
- forcing step execution with the COND parameter 5-15
- form and character control, requesting 7-58
- form-name subparameter
  - in SYSOUT parameter 12-121
  - of FORMS parameter
    - on JES3 **//\*FORMAT PR** statement 17-15
    - on JES3 **//\*FORMAT PU** statement 17-21
  - output form, requesting 7-59
- FORMDEF Parameter on OUTPUT JCL statement
- forms
  - See form-name subparameter
  - See FORMS parameter
- forms control
  - See FCB parameter
  - See FORMS parameter
- forms control buffer feature
  - See FCB parameter
- forms control, requesting 7-60
- forms overflow
  - See OVFL parameter on JES3 **//\*FORMAT PR** statement
- forms overflow processing, specifying in JES3 7-56
- forms overlay
  - See FLASH parameter
- forms overlay, requesting 7-62
- FORMS parameter
  - on **/\*JOBPARM** statement 16-5
  - on **/\*OUTPUT** statement 16-19
  - on JES3 **//\*FORMAT PR** statement 17-15
  - on JES3 **//\*FORMAT PU** statement 17-21
  - on OUTPUT JCL statement 14-33
  - printer form and character control, requesting 7-58
- FORMS Parameter on OUTPUT JCL statement
- FREE operator command (JES3) 17-3
- FREE parameter on DD statement
  - description 12-84
  - subparameter definition 12-84
  - syntax 12-84
  - use of (dynamic deallocation) 4-12, 6-3
  - using 4-12, 6-3
- FRID subparameter on DCB parameter
  - description 12-39
  - using with FCB parameter 12-78

- FUNC** subparameter on **DCB** parameter  
 description 12-39  
 for interpretation of punched output 7-57  
 with **LABEL** parameter 12-93  
 3525 7-57
- G** parameter (JES2)  
 See **COPYG** parameter on **/\*OUTPUT** statement  
 See **PAGES** parameter on **/\*JOBPARM** statement
- GAM** subparameters of **DCB** parameter 12-36-12-48
- GDG** (generation data groups)  
 building a base entry 8-25  
 creating 8-26  
 creating a model data set label 8-26  
 definition 8-25  
 deleting 8-30  
 example 8-31  
 generations of 7-4  
 parameters for creating 8-26-8-27  
   **DCB** parameter 8-27  
   **DISP** parameter 8-27  
   **DSNAME** parameter 8-26  
   **LABEL** parameter 8-27  
   **SPACE** parameter 8-27  
   **UNIT** parameter 8-27  
   **VOLUME** parameter 8-27  
 parameters for retrieving 8-28-8-30  
   **DCB** parameter 8-30  
   **DISP** parameter 8-29  
   **DSNAME** parameter 8-28  
   **LABEL** parameter 8-30  
   **UNIT** parameter 8-29  
   **VOLUME** parameter 8-29  
 restarting 8-30  
 retrieving 8-28  
 retrieving a single generation data set 8-28  
 retrieving all generations of a **GDG** 8-28  
 uncataloging 8-30
- generation data group  
 See **GDG** (generation data groups)
- generation data groups residing on **DASD** volumes,  
 retrieving 7-39
- generation data set  
 See **GDG** (generation data groups)
- generations of a generation data group  
 See **GDG** (generation data groups)
- generic name  
 See device-type subparameter of **UNIT** parameter
- GETMAIN**  
 See **ADDRSPC** Parameter  
 See **REGION** Parameter
- global processor  
 See processor selection  
 See **SYSTEM** parameter on **JES3 /\*MAIN**  
 statement
- GNCP** subparameter on **DCB** parameter 12-39
- graphic device 7-24
- graphics access method  
 See **GAM** subparameters of **DCB** parameter
- group name  
 See group-name subparameter of **UNIT** parameter
- GROUP** Parameter, on **JOB** statement  
 See also **PASSWORD** and **USER** parameters  
 example 10-13  
 subparameter definition 10-12  
 syntax 10-12  
 use of 6-2
- group-name subparameter  
 of **DEST** parameter 17-13  
   on **DD** statement 12-55  
   on **JES3 /\*FORMAT PR** statement 17-13  
   on **JES3 /\*FORMAT PU** statement 17-21  
   on **OUTPUT JCL** statement 14-24  
 of **GROUP** parameter on **JOB** statement 10-12  
 of **ORG** parameter on **JES3 /\*MAIN**  
 statement 17-30  
 See also **JES3 /\*MAIN** statement
- group-name subparameter of **UNIT** parameter  
 description 12-131  
 use of 7-28
- group-value subparameter of **COPIES** parameter  
 on **/\*OUTPUT** statement 16-15  
 on **DD** statement 12-25  
 on **JES3 /\*FORMAT PR** statement 17-13  
 on **OUTPUT JCL** statement 14-17  
 use of 7-58
- GROUPID** Parameter on **OUTPUT JCL** statement  
 guide to cataloged procedure  
 See cataloged and in-stream procedures  
 guide to data allocation control 6-1  
 guide to job and step control 5-1  
 guide to job control 3-1  
 guide to special data sets  
   associated data sets 7-6  
   dummy 8-8  
   generation data groups (**GDG**) 8-25-8-31  
   in the input stream 8-13  
   introduction 8-1  
   **ISAM** 8-18-8-24  
   private and temporary libraries 8-1-8-5  
   subsystem data set 8-32  
   virtual input/output (**VIO**) 8-10  
   **VSAM** 8-14-8-17
- Guide to Specifying Data Set Information 7-1  
 guide to step control 4-1  
 guide to using data set control statements 6-9
- G11** character set (3211) 12-128, 14-54
- H** operator command (JES3)  
 See **DISABLE** operator command (JES3)
- high watermark setup  
 See types of **JES3** setup
- high-density dumps  
 See also **CHARS** parameter  
 See also **FCB** parameter  
 requesting 8-6
- HN** character set  
 for 1403 12-128, 14-54

# Index

- for 3203 Model 5 12-128, 14-54
- HOLD operator command (JCL) 15-2
- hold output
  - See HOLD parameter
- HOLD parameter
  - delaying the writing of an output data set 7-55
  - on DD statement 12-87
    - description 12-87
    - example 12-88
    - relationship to other parameters 12-88
    - subparameter definition 12-87
    - syntax 12-87
  - on JES3 **//\*MAIN** statement 17-28
  - relationship to NOTIFY
    - JES2 control statement 12-88
    - parameter on JOB statement 12-88
- HOLD subparameter of TYPRUN parameter 10-39
- HOLD subparameter on JES3 **//\*MAIN** statement 17-27
  - See also FAILURE parameter on JES3 **//\*MAIN** statement
- how the system handles the DDNAME parameter 7-1
- how the system performs data set integrity processing 7-20
- how the system satisfies specific volume requests 7-25
- how the system satisfies your primary and secondary request 7-40
- how to call cataloged and in-stream procedures 9-3
- how to code NET statements 3-29
- HWS subparameter
  - of SETUP parameter on JES3 **//\*MAIN** statement 17-31
- H11 character set (3211) 12-128, 14-54
  
- I operator command (JES3)
  - See INQUIRY operator command (JES3)
- I parameter (JES2)
  - See INDEX parameter on **/\*OUTPUT** statement
- I subparameter of AMP parameter 12-13
- IBM standard labels 7-8, 12-90
  - See also LABEL parameter on DD statement
  - use of 7-9, 7-10
- identifier field in control statements 2-3
- identifying an in-stream procedure 9-2
- identifying comments on an output listing 2-7
- identifying data sets to the system, example of 7-13
- identifying procedure statements on an output listing 9-13
- identifying the program to be executed 4-4
- IEBIMAGE program
  - See also MODIFY parameter
  - for character arrangement tables 7-60
  - for copy modification modules 7-58
- IEFBR14 program
  - description 4-7
  - example of use 5-15
- IL subparameter of AMP parameter 12-13
- image for printing a data set, requesting 7-60
- image identifier
  - See FCB parameter
- image-name subparameter of FCB parameter
  - on JES3 **//\*FORMAT PR** statement 17-14
- implied unit affinity 7-33
- IN subparameter of LABEL parameter
  - description 12-92
  - use of 7-11
- in-stream procedure
  - See also PEND Statement
  - See also PROC Statement
  - calling 9-1, 9-3
  - changes, allowing for 9-3
  - DD statement parameters, modifying 9-6
  - DD statements, adding 9-8
  - definition 1-7
  - description 2-21, 9-1
  - EXEC statement parameters, modifying 9-4
  - identifying 9-2
  - identifying on an output listing 9-13
  - introduction 1-7
  - maximum per job 1-7
  - modifying 9-4
  - OUTPUT JCL statement parameters, modifying 9-9
  - OUTPUT JCL statements, adding 9-10
  - restrictions on contents of 9-1
  - symbolic parameters with 9-2
  - symbolic parameters, example 2-15, 2-21
  - symbolic parameters, use 2-15-2-21
  - using 9-3
  - writing 9-1
- incremental quantity
  - See secondary request for space
- IND subparameter (JES2)
  - See SYSAFF parameter on **/\*JOBPARM** statement
- independent mode affinity (JES2) 5-3
  - See also SYSAFF parameter on **/\*JOBPARM** statement
- index area of indexed sequential data set 8-18
- INDEX parameter
  - on **/\*OUTPUT** statement 16-19
  - on OUTPUT JCL statement 14-36
- INDEX Parameter on OUTPUT JCL statement
- index print position
  - See INDEX parameter
- index space, requesting for an indexed sequential data set 7-42
- index subparameter of SPACE parameter 12-114
  - See also SPACE parameter
- indexed sequential data set
  - absolute track technique 8-21
  - area arrangement of 8-22
  - areas of 7-4, 8-18
  - creating 8-18
  - creating and retrieving 8-18
  - examples of creating and retrieving 8-24
  - insufficient storage for allocation 7-40, 8-20
  - nonspecific allocation technique 8-21
  - requesting index space 7-42
  - retrieving 8-22



specific tracks, assigning 7-43  
 temporary 7-6  
 use of with indexed sequential data sets 8-23  
 initiation of jobs, delaying (JES2) 3-23  
 initiation of jobs, delaying (JES3) 3-24  
 INOUT specification for BSAM, overriding 7-11  
   See also OPEN macro options, overriding  
 input stream  
   See also \* DD parameter  
   See also DATA parameter on DD statement  
   data in input stream 9-2  
   definition 1-7  
   entering commands 15-2  
   entering data 8-13  
 input/output operations, bypassing  
   See DUMMY parameter on DD statement  
 INQUIRY operator command (JES3) 17-3  
 installation management information 3-2  
 installation management information: the ACCT  
   parameter 4-10  
 installation-written writer routine 7-54  
 insuring data set integrity 7-19  
 INT parameter on JES3 **FORMAT** PU statement  
   See also JES3  
   description 17-21  
   use of 7-57  
 integrity, data set  
   exclusive control 7-20  
   how system performs 7-20  
   insuring 7-19  
   shared control 7-20  
   table summarizing 7-22  
 Interactive Problem Control System  
   printing a dump 13-12  
   requesting an abnormal termination dump 8-7  
 Interpretation of Punched Output 7-57  
 INTRDR 12-121, 14-56  
   on DD SYSOUT parameter 12-121  
   on OUTPUT JCL WRITER parameter 14-56  
   use of 7-52  
 INTVL subparameter on DCB parameter  
   description 12-39  
   using with FCB parameter 12-78  
 IORATE parameter on JES3 **MAIN** statement  
   description 17-28  
   use of 5-19  
 IPCS  
   See Interactive Problem Control System  
 ISAM data set  
   See indexed sequential data set  
 ISO/ANSI printer control characters in RECFM  
   subparameter 12-45  
 ISO/ANSI Version 1 labels 12-90  
 ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3  
   labels and user labels 12-91  
 ISO/ANSI/FIPS tape labels 12-110  
 ISO/ANSI/FIPS Version 3 labels 12-90  
  
 J parameter (JES2)

  See NOLOG parameter on **JOBPARM** statement  
   J parameter on JES3 **DATASET** statement 17-5  
 JCL Statements  
   CNTL Statement 15-5  
     description 15-5  
   coding conventions 2-1  
   COMMAND Statement 15-2  
     description 15-2  
   COMMENT Statement 15-4  
     description 15-4  
   continuing 2-6  
   DD Statement 12-1  
     description 12-1  
   DELIMITER Statement 15-7  
     description 15-7  
   ENDCNTL Statement 15-8  
     description 15-8  
   EXEC Statement 11-1  
     description 11-1  
   fields of 2-3-2-6  
   introduction 1-1  
   JOB Statement 10-1  
     description 10-1  
     introduction 1-2  
   NULL Statement 15-9  
     description 15-9  
   OUTPUT statement 14-1  
     description 14-1  
   PEND Statement 15-10  
     description 15-10  
   PROC Statement 15-11  
     description 15-11  
     requesting listings of 3-14  
 JCLHOLD subparameter of TYPRUN  
   parameter 10-39  
 JCLTEST (JES3) 3-25  
 JESDS parameter 7-49  
 JESDS Parameter on OUTPUT JCL statement  
 JESJCL subparameter on JES3 **FORMAT** PR  
   statement 17-10  
 JESMSG subparameter on JES3 **FORMAT** PR  
   statement 17-10  
 JES2  
   accounting information, introduction 3-4  
   assigning a job priority 5-20  
   hard-copy log 3-16  
   job initiation, delaying 3-23  
   job scheduling 5-1  
   NETACCT control statement, use of 3-4  
   NOTIFY control statement, use of 3-19  
   operator commands 16-2  
   output class processing 7-54  
   output limiting 7-56  
   remote job processing 3-20  
   SETUP statement, use of 3-26  
   SIGNOFF control statement, use of 3-21  
   SIGNON control statement, use of 3-21  
   support of the 3211 indexing feature 7-57  
 JES2 control statements  
 JES3

# Index

- assigning a job priority 5-20
- forms overflow processing and printer spacing, specifying 7-56
- handling of unit and volume references 7-31
- job initiation, delaying 3-24
- job processing balance, establishing 5-19
- job scheduling 5-2
- main device scheduler messages 3-17
- NET statement, use of 3-28
- NETACCT control statement, use of 3-5
- operator commands 17-3
- output class processing 7-54
- output limiting 7-56
- priority aging 5-21
- punch output interpretation on a 3525 7-57
- remote job processing 3-12, 3-22
- resource allocation 6-4
  - See also allocation
- setup 6-5
  - See also types of JES3 setup
- SETUP parameter, use of 3-26
- SIGNOFF control statement, use of 3-23
- SIGNON statement, use of 3-22
- spool partitioning 3-33, 6-1
- system messages 3-18
- JES3 **//\*FORMAT PR** statement
  - description 17-9
  - examples 17-17
  - introduction 1-5
  - mutually exclusive parameters 17-11, 17-14
  - parameter definition 17-10
  - printer form and character control, requesting 7-58
  - syntax 17-9
- JES3 **//\*FORMAT PU** statement
  - description 17-18
  - example 17-22
  - introduction 1-5
  - parameter definition 17-19
  - punched output, requesting interpretation of 7-57
- JES3 **//\*MAIN** statement
  - deadline scheduling 3-27
  - description 17-23
  - example 17-34
  - introduction 1-5
  - job processing balance 5-23
  - job setup 6-4
  - output 3-12
  - parameter definition 17-24
  - syntax 17-23
  - updating a procedure library 9-3
- JES3 **//\*NET** statement
  - dependent job control 3-27-3-33
  - description 17-35
  - establishing dependencies between nets 3-29
  - examples 3-30, 17-39
  - how to code 3-29
  - introduction 1-5
  - parameter definition 17-35
  - placing a job on hold 3-28
  - specifying dedicated devices 3-29
  - specifying early setup 3-29
  - syntax 17-35
  - termination of a job 3-28
  - use of 3-28
- JES3 **//\*NETACCT** statement
  - defaults 17-41
  - description 17-40
  - example 17-41
  - introduction 1-5
  - parameter definition 17-40
  - syntax 17-40
  - use of 3-5
- JES3 **//\*OPERATOR** statement
  - description 17-42
  - introduction 1-5
- JES3 **//\*PAUSE** statement
  - description 17-43
  - introduction 1-5
- JES3 **//\*PROCESS** statement
  - description 17-44
  - examples 17-46
  - introduction 1-5
  - nonstandard job processing 1-6
  - parameter definition 17-44
- JES3 **//\*ROUTE XEQ** statement
  - description 17-47
  - example 17-48
  - introduction 1-6
  - parameter definition 17-47
  - routing in a network 3-10
  - rules for coding 17-47
- JES3 control statements
  - job
    - assigning a priority (JES2) 5-20
    - assigning a priority (JES3) 5-20
    - class
      - See CLASS Parameter
    - failure, JES recovery 5-26
      - See also restarting a job
    - in input stream 1-8
    - initiation, delaying (JES2) 3-23
    - initiation, delaying (JES3) 3-24
    - introduction 1-1
  - journal
    - See JOURNAL parameter on JES3 **//\*MAIN** statement
  - library
    - See JOBLIB DD statement
    - See private library
  - log (JES2) 3-16
    - See also NOLOG parameter on **/\*JOBPARM** statement
  - management information on JOB statement 3-2
  - name, use of 3-1
  - performance 5-22
    - See also PERFORM Parameter
  - priority 5-1
    - See also **/\*PRIORITY** statement
    - See also PRTY Parameter
  - processing balance, establishing in JES3 5-19

- processing, remote in JES3 3-12
- related output 3-14
  - See also job log
  - See also MSGCLASS Parameter on JOB statement
  - See also MSGLEVEL parameter on JOB statement
- scheduling
  - deadline (JES3) 3-27
  - for JES2 5-1
  - for JES3 5-2
  - improving 5-1
- selection 5-1
- setup
  - See SETUP parameter on JES3 **//\*MAIN** statement
- Job Accounting Information Parameter
- job control language statements
  - See JCL Statements
- job entry subsystem (2)
  - See JES2
- job entry subsystem (3)
  - See JES3
- job log 3-14
- job step
  - dispatching priority, assigning 5-21
  - in input stream 1-9
  - introduction 1-2
  - maximum number permitted 1-7
  - naming 4-3
  - performance 5-22
- JOB** subparameter
  - of **SETUP** parameter on JES3 **//\*MAIN** statement 17-31
- JOBCAT** DD statement
  - catalogs, system search order 8-5
  - description 13-2
  - examples 13-3
  - location in the JCL 13-2
  - master catalog 8-5
  - private catalog 8-5
  - relationship to **STEPCAT** 13-2
  - syntax 13-2
  - VSAM data sets 8-14
- jobclass subparameter of **CLASS** parameter 10-9
- JOBLIB** DD statement
  - adding members to a private library 8-2
  - concatenating private libraries 8-4
  - creating a private library 8-2
  - description 13-4
  - examples 8-3, 8-4, 13-6
  - location in the JCL 13-5
  - relationship to **COND** 13-5
  - relationship to **STEPLIB** 13-5
  - retrieving a private library 8-3
  - rules for coding parameters 13-4
    - when the library is cataloged 13-4
    - when the library is not cataloged 13-4
  - syntax 13-4
- jobname
  - introduction 3-1
- JOURNAL** parameter on JES3 **//\*MAIN** statement 17-28
- JSTTEST** (JES3) 3-25
- K** parameter (JES2)
  - See **LINECT** parameter on **/\*JOBPARM** statement
  - See **LINECT** parameter on **/\*OUTPUT** statement
- KEEP** subparameter of **DISP** parameter
  - description 12-59
  - use of 7-16
- keeping a data set 7-16
  - See also **DISP** parameter on **DD** statement
- KEYLEN** subparameter on **DCB** parameter
  - description 12-40
  - with **SPACE** parameter 12-115
- keylength
  - See **KEYLEN** subparameter on **DCB** parameter
- keyword parameters
  - definition 2-5
  - list of 10-2
    - on **DD** statement 12-3
    - on **EXEC** statement 11-2
    - on **JOB** statement 10-2
- L** parameter (JES2)
  - See **LINDEX** parameter on **/\*OUTPUT** statement
  - See **LINES** parameter on **/\*JOBPARM** statement
- L** subparameter of **AMP** parameter 12-13
- label
  - See **LABEL** parameter on **DD** statement
- LABEL** parameter on **DD** statement
  - defaults 12-93
  - description 12-89
  - examples 12-94
  - expiration date, use of 7-12
  - relationship to other parameters 12-93
  - retention period, use of 7-12
  - subparameter definition 12-89
  - use of 7-7
  - use of with indexed sequential data sets 8-20
  - use of with nonspecific volume requests 7-10
  - use with generation data sets 8-27, 8-30
- label type on **LABEL** parameter
  - description 7-8, 12-90
  - use of 7-8
- length restriction for symbolic parameters 2-17
- lengthening a data set
  - See also **MOD** subparameter of **DISP** parameter
  - additional space 7-40-7-42
  - exclusive control 7-19
  - multivolume data set 7-28
- libraries, temporary 8-5
- library
  - See also **JOBLIB** DD statement
  - See also **STEPLIB** DD statement
  - placing a cataloged procedure in 9-2
  - private 8-1-8-4

# Index

- temporary 8-1-8-5
- LIMCT subparameter on DCB parameter 12-40
- limiting execution time
  - for a job 5-16
    - See also TIME Parameter, on JOB statement
  - for a job step 5-16
    - See also TIME Parameter, on EXEC statement
- limiting job and job step execution time 5-16
- limiting output records 7-55
- LINDEX parameter
  - on /\*OUTPUT statement 16-19
  - on OUTPUT JCL statement 14-39
- LINDEX Parameter on OUTPUT JCL statement
- LINECT parameter
  - on /\*JOBPARM statement 16-5
  - on /\*OUTPUT statement 16-19
  - on OUTPUT JCL statement 14-40
- LINECT Parameter on OUTPUT JCL statement
- LINES parameter
  - on /\*JOBPARM statement 16-6
  - on JES3 /\*MAIN statement 17-28
- listings of JCL statements and system messages 3-14
  - See also MSGCLASS Parameter on JOB statement
  - See also MSGLEVEL parameter on JOB statement
- LOCAL subparameter
  - of DEST parameter
    - on /\*OUTPUT statement 16-16
    - on DD statement 12-53
    - on JES3 /\*FORMAT PR statement 17-13
    - on JES3 /\*FORMAT PU statement 17-21
    - on OUTPUT JCL statement 14-23
  - of PRINT, PUNCH, and XEQ parameters
    - on ROUTE statement (JES2) 16-25
- LOG operator command (JCL) 15-2
- log, hard-copy (JES2) 3-16
- log, job 3-14
- logical record length
  - See LRECL subparameter on DCB parameter
- LRECL subparameter on DCB parameter
  - coded with \* parameter 12-6
  - description 12-40
  - with QNAME parameter 12-111
- LREGION parameter on JES3 /\*MAIN statement
  - description 17-29
  - use of 5-25
- LTM subparameter of LABEL parameter 12-91
- M parameter (JES2)
  - See BYTES parameter on /\*JOBPARM statement
  - See MODTRC parameter on /\*OUTPUT statement
- MACRF DCB macro operand
  - when coding DUMMY 12-74
- main device scheduler messages (JES3) 3-17
- MAS
  - See multi-access spool (JES2)
- mass storage group
  - See MSVGP parameter on DD statement
- mass storage system
  - See MSVGP parameter on DD statement
- mass storage volume groups 7-37
  - ensuring proper multivolume extensions 7-38
  - specifying SPACE parameter 7-39
  - specifying VOLUME parameter 7-38
  - volume requests 7-38
- mass storage volumes, requesting space for non-VSAM data sets on 7-39
- MDS messages (JES3) 3-17
- members of a partitioned data set
  - See partitioned data set, members
- members of a temporary partitioned data set
  - See temporary data set
- memory
  - See ADDRSPC Parameter
- message class parameter
  - See MSGCLASS Parameter on JOB statement
- message level parameter
  - See MSGLEVEL parameter on JOB statement
- MESSAGE operator command (JES3) 17-3
- message queue records
  - See THRESHOLD parameter
- messages, MDS (JES3) 3-17
- messages, system (JES3) 3-18
- minimum region size 5-24
  - See also REGION Parameter
- MOD subparameter of DISP parameter 12-58
  - use of 7-14
- mode for card reader/punch
  - See MODE subparameter on DCB parameter
- MODE parameter
  - on JES3 /\*DATASET statement 17-5
- MODE subparameter on DCB parameter
  - description 12-41
  - for binary input 3-26
- MODIFY operator command
  - for JCL 15-2
  - for JES3 17-3
- MODIFY parameter
  - copy modification, requesting 7-58
  - on /\*OUTPUT statement 16-20
  - on DD statement 12-95
    - defaults 12-96
    - description 12-95
    - example 12-97
    - relationship to other parameters 12-96
    - subparameter definition 12-95
    - syntax 12-95
  - on JES3 /\*FORMAT PR statement 17-15
  - on OUTPUT JCL statement 14-41
- MODIFY Parameter on OUTPUT JCL statement
- modifying cataloged and in-stream procedures 9-4
- modifying parameters on a DD statement 9-6
- modifying parameters on an EXEC statement 9-4
- modifying parameters on an OUTPUT JCL statement 9-9
- modifying parameters on DD statements that define concatenated data sets 9-8
- MODTRC parameter
  - on /\*OUTPUT statement 16-20
- module-name subparameter of MODIFY parameter

- on /\*OUTPUT statement 16-20
- on DD statement 12-95
- on JES3 /\*FORMAT PR statement 17-15
- on OUTPUT JCL statement 14-41
- MONITOR operator command (JCL) 15-2
- MOUNT operator command (JCL) 15-2
- MSGCLASS Parameter on JOB statement
  - examples 10-15
  - relationship to SYSOUT 10-14
  - subparameter definition 10-14
  - syntax 10-14
  - use of 3-16
- MSGCLASS subparameter of CLASS parameter (JES3) 17-6
  - See also /\*DATASET statement
- MSGLEVEL parameter on JOB statement
  - examples 10-17
  - subparameter definition 10-16
  - syntax 10-16
  - use of 3-14
  - using with COMMENT statement 15-4
- MSS
  - See MSVGP parameter on DD statement
- MSS parameter on JES3 /\*MAIN statement 17-29
- MSVGP parameter on DD statement
  - defining mass storage volumes 7-37
  - description 12-98
  - examples 12-100
  - relationship to other parameters 12-99
  - subparameter definition 12-98
  - syntax 12-98
- multi-access spool (JES2) 5-2, 5-3
- multiple copies of an output data set, requesting using JES2 7-57
- multiple copies of an output data set, requesting using JES3 7-58
- multiple units 7-29
- mutivolume data sets 7-28
- mutually exclusive parameters
  - table of 18-8
  - used to override a parameter in a procedure 9-6
- MXIG subparameter of SPACE parameter 12-114
  - See also SPACE parameter
  
- N operator command (JES3)
  - See ENABLE operator command (JES3)
- N parameter (JES2)
  - See also COPIES parameter on /\*JOBPARM statement
  - See also COPIES parameter on /\*OUTPUT statement (JES2)
  - on ROUTE statement (JES2) 16-25
  - on XEQ statement (JES2) 16-32
- N subparameter of BURST parameter
  - on /\*JOBPARM statement 16-4
  - on /\*OUTPUT statement 16-14
  - on DD statement 12-16
  - on OUTPUT JCL statement 14-6
- N subparameter of RESTART parameter
  - on /\*JOBPARM statement 16-6
- name field in control statements 2-3
- name of a data set, specifying 7-2
  - See also DSNNAME parameter on DD statement
- name subparameter
  - of DEST parameter
    - on /\*OUTPUT statement 16-16
    - on DD statement 12-53
    - on OUTPUT JCL statement 14-23
  - of PRINT, PUNCH, and XEQ parameters
    - on ROUTE statement (JES2) 16-25
- naming a job step 4-3
- naming the job 3-1
- NC subparameter of RD parameter
  - on JOB statement 10-29
- NCK subparameter of AMP parameter 12-13
- NCP subparameter on DCB parameter 12-41
- NETID parameter on JES3 /\*NET statement
  - dependent job control 3-27-3-33
  - description 17-35
- NETREL parameter on JES3 /\*NET statement 17-37
  - dependent job control 3-27-3-33
- network accounting information 3-4
  - See also NETACCT statement for JES2 or JES3
- network-account-number
  - on NETACCT Statement (JES2) 16-10
- networking
  - considerations 3-6
  - controlling job execution node (JES3) 3-11
  - controlling output (JES2) 3-7
  - introduction 3-6
  - routing (JES3) 3-10
  - routing in (JES2) 3-7
  - transmitting data (JES2) 3-7
- networking (JES3) 3-10
- new data sets
  - disposition processing 7-14
  - requesting space on direct access devices 7-39
  - specifying status of 7-14
  - unit and volume requests 7-24
- NEW subparameter of DISP parameter 12-57
  - exclusive control 7-19
  - use of 7-19-7-23
- new-password subparameter of PASSWORD parameter
  - on JOB statement 10-21
- NHOLD parameter on JES3 /\*NET statement
  - description 17-37
  - use of 3-28
- NL subparameter of LABEL parameter 12-91
- Nnnnn parameter
  - on XEQ statement (JES2) 16-32
  - on XMIT statement (JES2) 16-33
- Nnnnn subparameter of DEST parameter
  - of PRINT, PUNCH, and XEQ parameters
    - on ROUTE statement (JES2) 16-25
  - on /\*OUTPUT statement 16-16
  - on DD statement 12-54
  - on OUTPUT JCL statement 14-23
- NnnRmmm subparameter
  - of DEST parameter

# Index

- on DD statement 12-54
- NnnRmmmm subparameter
  - of DEST parameter
    - on /\*OUTPUT statement 16-16
    - on OUTPUT JCL statement 14-24
  - of PRINT, PUNCH, and XEQ parameters
    - on ROUTE statement (JES2) 16-25
- no password to read
  - See NOPWREAD subparameter of LABEL parameter
- NO subparameter 12-87
  - of CLASS parameter on JES3 /\*DATASET statement 17-6
  - of EXPDTCHK parameter on JES3 MAIN statement 17-27
  - of HOLD parameter on DD statement 12-87
  - of HOLD parameter on JES3 MAIN statement 17-28
  - of INT parameter on JES3 /\*FORMAT PU statement 17-22
  - of J parameter on JES3 /\*DATASET statement 17-5
- node, transmitting data to (JES2) 3-7
- nodename parameter
  - on XEQ statement (JES2) 16-32
  - on XMIT statement (JES2) 16-33
- nodename subparameter
  - of DEST parameter
    - on DD statement 12-55
    - on JES3 /\*FORMAT PR statement 17-13
    - on JES3 /\*FORMAT PU statement 17-21
    - on OUTPUT JCL statement 14-25
  - of ORG parameter
    - on JES3 /\*MAIN statement 17-30
- nodename.remote subparameter
  - of DEST parameter
    - on OUTPUT JCL statement 14-25
- nodename.userid parameter
  - on /\*NOTIFY Statement 16-11
- nodename.userid subparameter
  - of DEST parameter
    - on /\*OUTPUT statement 16-16
    - on OUTPUT JCL statement 14-24
    - on ROUTE statement (JES2) 16-25
  - of PRINT, PUNCH, and XEQ parameters
    - on ROUTE statement (JES2) 16-25
- NOLOG parameter on /\*JOBPARM statement
  - description 16-6
  - use of 3-14
- non-VSAM data sets on mass storage volumes, requesting space for 7-39
- NONE subparameter on JES3 /\*MAIN statement 17-27
  - See also FETCH parameter on JES3 /\*MAIN statement
- nonpageable dynamic area 5-24
  - See also ADDRSPC Parameter
  - See also REGION Parameter
- nonspecific allocation technique for indexed sequential data sets 8-21
- nonspecific volume requests 7-26
  - for mass storage volumes 7-38
  - space requests for 7-26
  - types of 7-26
- nonstandard job (JES3), definition 1-6
- nonstandard job processing-(JES3) 1-6
  - See also JES3 /\*PROCESS statement
- nonstandard labels 7-8
  - See also LABEL parameter on DD statement
- nontemporary data set
  - areas of an indexed sequential data set 7-4
  - creating 7-3
  - generations of a generation data group 7-4
  - members of partitioned data set 7-4
  - retrieving 7-3
- NOPWREAD subparameter of LABEL parameter
  - description 12-92
  - use of 7-10
- normal disposition of data sets 7-13
- NORMAL parameter on JES3 /\*NET statement 17-36
- Notation used to show syntax 2-1
- NOTIFY parameter on JOB statement
  - example 10-19
  - receiving notification 10-18
    - for JES2 10-18
    - for JES3 10-18
  - relationship to system affinity (JES2) 5-4
  - subparameter definition 10-18
  - syntax 10-18
  - use of 3-18
- NR subparameter of RD parameter
  - on Job statement 10-30
- NRC subparameter of AMP parameter 12-13
- NRCMP parameter on JES3 /\*NET statement 17-37
- NRE subparameter of AMP parameter 12-13
- NSL subparameter of LABEL parameter 12-91
- NTM subparameter on DCB parameter 12-41
- NULL Statement
  - description 15-9
  - example 15-9
  - introduction 1-1
- NULLFILE, assign to DSNNAME parameter 8-9
  - See also DUMMY parameter on DD statement, examples
- nullifying parameters in a procedure
  - DCB parameter 9-7
  - DUMMY parameter 9-7
    - on DD statements 9-6
    - on EXEC statements 9-4
    - on OUTPUT JCL statements 9-9
  - symbolic parameters 2-18
- O parameter (JES2)
  - See FLASH parameter on /\*OUTPUT statement
- obtaining output
  - example for JES2 3-9
  - example for JES3 3-13
- OFF subparameter of OVFL parameter (JES3) 17-15

- old data sets
  - See OLD subparameter of DISP parameter
- OLD subparameter of DISP parameter
  - description 12-57
  - use of 7-14
- on JOBLIB DD statement 13-4
- ON subparameter of OVFL parameter (JES3) 17-15
- ONLY subparameter of COND parameter on EXEC statement
  - use of 5-8
- OPEN macro options, overriding 7-11
  - See also IN subparameter of LABEL parameter
  - See also OUT subparameter of LABEL parameter
- OPEN/CLOSE/EOV trace option
  - See DIAGNS subparameter on DCB parameter
- operation field in control statements 2-3
- operator commands
  - for JCL 15-2
  - for JES2 16-2
  - for JES3 17-3
- operator subparameter of COND parameter
  - on EXEC statement 11-9
  - on JOB statement 10-10
- OPHOLD parameter on JES3 `/*NET` statement
  - description 17-38
  - use of 3-28
- OPTCD subparameter
  - for ASCII tape data sets 7-9
  - of AMP parameter 12-13
  - on DCB parameter 12-42
  - with QNAME parameter 12-111
- optional services
  - See OPTCD subparameter
- ORG parameter on JES3 `/*MAIN` statement
  - description 17-29
  - nodename subparameter 17-30
- OUT subparameter of LABEL parameter
  - description 12-92
  - use of 7-11
- OUTIN specification for BSAM, overriding 7-11
  - See also OPEN macro options, overriding
- OUTLIM parameter on DD statement
  - default 12-101
  - description 12-101
  - example 12-102
  - limiting output records 7-55
  - relationship to other parameters 12-101
  - subparameter definition 12-101
  - syntax 12-101
- output class
  - assigning data sets to 7-51
  - assigning messages to 3-14
  - processing (JES2) 7-54
  - processing (JES3) 7-54
- output data sets
  - allocating space for
    - See SPACE parameter
  - assigning to output classes 7-51
  - conditional disposition
    - See COND Parameter
  - See DISP parameter on DD statement
  - delaying the writing of 7-55
  - disposition
    - See DISP parameter on DD statement
  - parameters for creating, table of 18-2-18-3
  - parameters for extending, table of 18-5
  - processing for the job 7-44
  - routing of
    - See SYSOUT parameter
  - stackers for
    - See BURST parameter
    - See STACKER parameter on JES3 `/*FORMAT PR` statement
  - status
    - See DISP parameter on DD statement
  - suppressing the writing of 7-55
  - unit information
    - See UNIT parameter on DD statement
  - volume information
    - See VOLUME parameter
  - with UCS feature
    - See UCS parameter
- output destination, controlling using JES3 3-12
- output form, special, requesting 7-59
- OUTPUT JCL statement
  - See coding the OUTPUT JCL statement
- output limiting
  - See also OUTLIM parameter on DD statement
  - JES2 7-56
  - JES3 7-56
- output listing
  - dumps 8-6
  - identifying cataloged and in-stream procedures 9-13
  - JCL statements 3-14
  - suppressing of 7-55
  - system messages 3-14
- OUTPUT parameter 12-103
  - defaults 12-104
  - description 12-103
  - examples 12-105
  - overrides 12-104
  - subparameter definition 12-104
  - syntax 12-103
- output records, limiting 7-55
- overflow area of indexed sequential data set 8-18
- overflow processing, page, specifying in JES2 7-56
- overlay-name subparameter of FLASH parameter
  - on `/*OUTPUT` statement 16-18
  - on DD statement 12-81
  - on JES3 `/*FORMAT PR` statement 17-14
  - on OUTPUT JCL statement 14-28
- override
  - of cataloged procedures 1-7, 9-3-9-8
  - of original secondary quantity request for space 7-41
  - of symbolic parameters 2-15-2-19
  - parameters in a procedure 9-6
  - on a DD statement 9-6-9-8
  - on an EXEC statement 9-4-9-6

# Index

- on an OUTPUT JCL statement 9-9
- overriding OPEN macro options
  - See OPEN macro options, overriding
- OVFL parameter on JES3 **/\*FORMAT PR** statement
  - description 17-15
  - use of 7-56
- P parameter (JES2)
  - See CKPTPGS parameter on **/\*OUTPUT** statement
  - See PROCLIB parameter on **/\*JOBPARM** statement
- P subparameter of UNIT parameter
  - See also parallel mounting of devices
  - description 12-132
  - use of 7-29
- page
  - See REGION Parameter
- page overflow processing, specifying in JES2 7-56
- PAGEADD operator command (JCL) 15-2
- PAGEDEF Parameter on OUTPUT JCL statement
- PAGES parameter
  - on **/\*JOBPARM** statement 16-6
- PAGES parameter on JES3 **/\*MAIN** statement
  - description 17-30
- parallel mounting of devices 12-132
  - See also UNIT parameter on DD statement
- parameter definition 14-3
  - for BURST 14-6
  - for CHARS 14-7
  - for CKPTLINE 14-10
  - for CKPTPAGE 14-11
  - for CKPTSEC 14-12
  - for CLASS 14-13
  - for COMPACT 14-15
  - for CONTROL 14-16
  - for COPIES 14-17
  - for DEFAULT 14-20
  - for DEST 14-23
  - for FCB 14-26
  - for FLASH 14-28
  - for FORMDEF 14-31
  - for FORMS 14-33
  - for GROUPLD 14-34
  - for INDEX 14-36
  - for JESDS 14-37
  - for LINDEX 14-39
  - for LINECT 14-40
  - for MODIFY 14-41
  - for PAGEDEF 14-43
  - for PIMSG 14-45
  - for PRMODE 14-46
  - for PRTY 14-48
  - for THRESHLD 14-49
  - for TRC 14-51
  - for UCS 14-53
  - for WRITER 14-56
- Parameter field 2-5
- parameter field in control statements 2-3
- parameter field, coding 2-5
- parameters
  - adding, nullifying, overriding 2-21, 9-4
  - in cataloged or in-stream procedure 9-4-9-8
  - modifying on a DD statement 9-6-9-8
  - modifying on an EXEC statement 9-4-9-6
  - modifying on an OUTPUT JCL statement 9-9
  - notation for defining 2-1
  - symbolic 2-15-2-21
- the parameter field
- parenthesis
  - inclusion in variables 2-17-2-21
  - notation for coding 2-1
- PARM parameter ON EXEC statement
  - coding special characters 4-9
  - description 11-17-11-18
  - examples 11-18
  - modifying in a procedure 9-4-9-6
  - passing information 4-8
  - special characters 11-17
  - subparameter definition 11-17
  - syntax 11-17
  - using 4-8
- partition-name subparameter (JES3)
  - See SPART parameter on JES3 **/\*MAIN** statement
- partitioned data set
- partitioned data set, members 7-4
  - absolute track allocation 7-42
  - adding member 7-5
  - concatenated 9-8
  - creating 7-3-7-4, 18-2
  - directory space, requesting 7-42
  - extending 18-5
  - member.name 7-5
  - nontemporary name 7-3-7-4
  - retrieving a member of 7-3-7-4, 18-4
  - temporary name 7-5
- PASS subparameter of DISP parameter
  - description 12-59
  - use of 7-14, 7-17-7-19
- passed data sets, considerations 7-17
- passed unreceived data sets 7-18
- passing a data set 7-17
- passing a private library 8-1, 8-2
  - See also JOBLIB DD statement
  - See also STEPLIB DD statement
- passing information to the program in execution 4-8
- passing temporary data sets among job steps using VIO 8-12
- PASSWORD Parameter on JOB statement
  - See also GROUP and USER parameters
  - examples 10-21
  - subparameter definition 10-21
  - syntax 10-20
  - use of 6-2
- password protection
  - See NOPWREAD subparameter of LABEL parameter
  - See PASSWORD subparameter of LABEL parameter
- password subparameter



- of /\*SIGNON statement (JES2) 16-31
- of PASSWORD parameter on JOB statement 10-21
- of SIGNON statement (JES3) 17-50
- PASSWORD subparameter of LABEL parameter
  - description 12-92
  - use of 7-10
- PCAN character set 12-128, 14-54
  - for 1403 12-128, 14-54
  - for 3203 Model 5 12-128, 14-54
- PCHN character set 12-128, 14-54
  - for 1403 12-128, 14-54
  - for 3203 Model 5 12-128, 14-54
- PCI subparameter on DCB parameter 12-44
- PEND Statement
  - description 15-10
  - examples 15-10
  - introduction 1-1
  - use of 9-2
- PERFORM Parameter
  - example 10-23
  - on EXEC statement 11-19
    - default 11-19
    - examples 11-20
    - subparameter definition 11-19
    - syntax 11-19
    - use of, in JES2 5-22
    - use of, in JES3 5-23
  - on JOB statement 10-22
    - subparameter definition 10-22
    - syntax 10-22
    - use of, in JES2 5-22
    - use of, in JES3 5-23
  - using 5-2
- performance of jobs and job steps 5-22
  - See also PERFORM Parameter
  - in JES2 5-22
  - in JES3 5-23
- PGM parameter on EXEC statement
  - description 11-21-11-23
  - examples 11-22
  - JCLTEST 3-25
  - JSTTEST 3-25
  - restriction, cataloged procedures 9-4
  - subparameter definition 11-21
  - syntax 11-21
  - using 4-4
  - using with a private library 4-6
  - using with a temporary library 4-5
- physical records per track 18-11, 18-12
- PIMSG Parameter on OUTPUT JCL statement
- placing a cataloged procedure in a procedure library 9-2
- placing a job in a net on hold 3-28
- PN character set
  - for 1403 12-128, 14-54
  - for 3203 Model 5 12-128, 14-54
- PNAME parameter on JES3 /\*NETACCT statement 17-40
- point-to-point communications link 3-20
- positional parameters
  - definition 2-5
  - in parameters field 2-5
  - on DD statement 12-3
  - on EXEC statement 11-2
  - on JOB statement 10-2
- positioning the unit affinity request 7-34
- PR parameter on JES3 /\*FORMAT statement 17-10
- predecessor job
  - See dependent job control for JES3: The Job Net
- primary and secondary space request, how the system satisfies 7-40
- primary quantity subparameter of SPACE parameter 12-113
  - See also SPACE parameter
- prime area of indexed sequential data set 8-18
- print output (JES3)
  - See JES3 /\*FORMAT PR statement
- PRINT parameter on /\*ROUTE statement 16-24
- Print Services Facility 14-46
- Print Services Facility (PSF) 14-8
  - font list override by CHARS 14-8
  - font list override by UCS 12-128, 14-54
- PRINT subparameter on JES3 /\*MAIN statement 17-27
  - See also FAILURE parameter on JES3 /\*MAIN statement
- printer
  - See 1403, 3203 Model 5, 3211, or 3800 printer
- printer form, requesting 7-58
- printer spacing, specifying in JES3 7-56
- printer train 7-59
- priority
  - aging (JES3) 5-21
  - assigning to a job 5-20
  - automatic priority group (APG) 5-21
  - dispatching 5-21
    - See also DPRTY parameter on EXEC statement
  - output
    - See PRTY parameter on /\*FORMAT PR statement (JES3)
  - selection of jobs
    - See /\*PRIORITY statement
    - See PRTY parameter on JOB statement (JES2) or (JES3)
- private catalogs, using 8-5
  - See also JOBCAT DD statement
  - See also STEPCAT DD statement
- private library 4-6
  - See also JOBLIB DD statement
  - See also STEPLIB DD statement
  - See also UPDATE parameter on JES3 /\*MAIN statement
  - adding members 8-2
  - concatenating 8-4
  - creating 8-2
  - creating a private library 8-2
  - creating and using 8-1
  - defining JES3 catalog procedure 4-8
  - retrieving 8-3

# Index

- with PGM parameter 4-4-4-6
- PRIVATE subparameter of VOLUME parameter
  - description 12-136
  - use of 7-27
- private volumes, using 7-27
  - See also PRIVATE subparameter of VOLUME parameter
- PRMODE 14-46
- PRMODE Parameter on OUTPUT JCL statement
- PROC parameter
  - on EXEC statement 11-24
    - examples 11-25
    - subparameter definition 11-24
    - syntax 11-24
    - using 4-7
- PROC parameter on JES3 **/\*MAIN** statement
  - description 17-30
  - using 4-8
- PROC Statement
  - description 15-11
  - examples 15-12
  - introduction 1-1
  - use of 9-1-9-3
- procedure end
  - See PEND Statement
- procedure library (SYS1.PROCLIB)
  - cataloged procedure 1-7, 2-21, 9-1
  - definition 1-7
- Procedure name parameter
  - See PROC parameter
- procedure statement
  - See PROC Statement
- procedure step 9-1
- process-mode subparameter of PRMODE parameter 14-46
- processing options, specifying data set 7-43
- processing output data sets for the job 7-44
- processing program information 4-4
- Processing your job 1-7
- processor selection 5-3
  - for JES2 5-3
  - for JES3 5-4
- processor time limit
  - See TIME Parameter
- processor-id subparameter of MAIN parameter 17-24
- PROCLIB parameter on **/\*JOBPARM** statement
  - description 16-6
  - using 4-8
- procname subparameter of QNAME parameter 12-111
- program control
  - for JCL 15-5
- PROGRAM subparameter on JES3 **/\*FORMAT PR** statement 17-12
- PROGRAM subparameter on OUTPUT JCL statement 14-16
- program-name subparameter of PGM parameter 11-21
- program, calling 4-4
- programmer information on JOB statement 3-5
- programmer information: the programmer-name parameter 3-5
  - programmer's name parameter on JOB statement
    - examples 10-25
    - syntax 10-24
    - use of 3-5
- PROTECT parameter on DD statement
  - description 12-109
  - examples 12-110
  - relationship to other parameters 12-109
  - subparameter definition 12-109
  - syntax 12-109
- PRTSP subparameter on DCB parameter 12-45
  - with SYSOUT parameter 12-122
- PRTY Parameter
  - on JES3 **/\*FORMAT PR** statement 17-15
  - on JOB statement
    - example 10-27
    - subparameter definition 10-26
    - syntax 10-26
  - on OUTPUT JCL statement 14-48
  - use of with JES2 5-20
  - use of with JES3 5-20
- PRTY Parameter on OUTPUT JCL statement
- PSF 14-46
- PU parameter on JES3 **/\*FORMAT** statement 17-19
- public volume, requesting 7-26
  - See also VOLUME parameter
- punch output interpretation on a 3525, JES3 7-57
- PUNCH parameter on **/\*ROUTE** statement 16-24
- punched output, interpretation of 7-57
- punching a data set
  - See **/\*OUTPUT** Statement
  - See JES3 **/\*FORMAT PU** statement
- P11 character set (3211) 12-128, 14-54
  
- Q parameter (JES2)
  - See FLASHC parameter on **/\*OUTPUT** statement
- QISAM subparameters of DCB parameter 12-36, 12-48
  - See also indexed sequential data set
- QN character set
  - for 1403 12-128, 14-54
  - for 3203 Model 5 12-128, 14-54
- QNAME parameter on DD statement
  - description 12-111
  - example 12-111
  - relationship to other parameters 12-111
  - subparameter definition 12-111
  - syntax 12-111
- QNC character set
  - for 1403 12-128, 14-54
  - for 3203 Model 5 12-128, 14-54
- QSAM subparameters of DCB parameter 12-36-12-48
- qualified data set name 7-5
- queued indexed sequential access method
  - See QISAM subparameters of DCB parameter
- queued sequential access method
  - See QSAM subparameters of DCB parameter
- queuing options 5-18

- R operator command (JES3)
  - See RESTART operator command (JES3)
- R parameter (JES2)
  - See ROOM parameter on /\*JOBPARM statement
- R subparameter of RD parameter
  - on JOB statement 10-29
- RACF
  - See GROUP Parameter, on JOB statement
  - See PASSWORD Parameter on JOB statement
  - See PROTECT parameter on DD statement
  - See USER parameter on JOB statement
- racf-protected data sets 6-2
- RCK subparameter of AMP parameter 12-12
- RD Parameter 5-28
  - See also RESTART Parameter on JOB statement
  - on EXEC statement 11-26
    - examples 11-29
    - overrides 11-28
    - subparameter definition 11-27
    - syntax 11-26
    - use of 5-28
  - on JOB statement 10-28
    - default 10-30
    - examples 10-30
    - overrides 10-30
    - subparameter definition 10-29-10-30
    - syntax 10-28
    - use of 5-27
- RDJFCB macro 2-9
- READ macro instruction 7-20
- READ/WRITE macros before a CHECK macro
  - See NCP subparameter on DCB parameter
- reading a data set
  - dummy 8-9
  - multivolume 7-28
  - shared control 7-20
- reading column binary input 3-26
- real storage 5-23
  - See also requesting
- real storage requirements 5-24
- REAL subparameter of ADDRSPC parameter
  - on EXEC statement 11-6
  - on JOB statement 10-7
  - requesting storage 5-23
- RECFM subparameter
  - coded with DISP=MOD 12-61
  - of AMP parameter 12-13
  - on DCB parameter 12-45
  - with QNAME parameter 12-111
- record format
  - See RECFM subparameter, on DCB parameter
- record key position
  - See RKP subparameter on DCB parameter
- record length
  - See LRECL subparameter on DCB parameter
- REF subparameter of VOLUME parameter
  - description 12-138
  - specific volume request 7-24
  - volume affinity 7-27
- referback
  - See backward references
- references, backward
  - See backward references
- region
  - See REGION Parameter
- REGION Parameter
  - See also ADDRSPC Parameter
  - on EXEC statement 11-30
    - examples 11-31
    - override 11-31
    - subparameter definition 11-30
    - syntax 11-30
  - on JOB statement 10-31
    - default 10-31
    - examples 10-32
    - override 10-32
    - subparameter definition 10-31
    - syntax 10-31
    - use of 5-25
- region request, example 5-26
- region size, default 10-31, 11-31
- relational operators on the COND parameter
  - on EXEC statement 11-9
  - on JOB statement 10-10
  - use of 5-5-5-16
- relationship of the UNIT parameter to a volume reference 7-30
- relative generation numbers
  - assigning 7-4
  - obtaining from the catalog (JES2) 8-25
  - obtaining from the catalog (JES3) 8-25
  - relative generation numbers 8-25
    - obtaining from the catalog (JES2) 8-25
    - obtaining from the catalog (JES3) 8-25
- relative track number
  - See address subparameter of SPACE parameter
- RELEASE operator command (JCL) 15-2
- RELEASE parameter on JES3 /\*NET statement
  - description 17-38
  - use of 3-28
- releasing space
  - deleting a data set
    - See DELETE subparameter of DISP parameter
  - unused space
    - See RLSE subparameter of SPACE parameter
- RELEX macro instruction 7-20
- RELSCHCT parameter on JES3 /\*NET statement
  - description 17-38
  - use of 3-29
- remote job entry (JES2) 3-20
- remote job entry stations 3-21
- remote job processing 3-20
  - in JES2 3-20
  - in JES3 3-12, 3-22
- remote subparameter
  - of DEST parameter
    - on /\*OUTPUT statement 16-17
    - on DD statement 12-54
    - on JES3 /\*FORMAT PR statement 17-13
    - on JES3 /\*FORMAT PU statement 17-21

# Index

- on OUTPUT JCL statement 14-24
- of ORG parameter
  - See JES3 //\*MAIN statement
- remote terminal, use of 3-20
- remote work station, use of 3-20
- remote work stations (JES3) 3-22
- REMOTEnnn parameter
  - on /\*SIGNON statement (JES2) 16-30
- removable volume
  - See VOLUME parameter
- replacing variable data
  - See MODIFY parameter
- REPLY operator command (JCL) 15-2
- requesting
  - character arrangements with a 3800 printer 7-60
  - copy modification 7-58
  - directory space for a partitioned data set 7-42
  - forms control 7-60
  - forms overlay 7-62
  - index space for an indexed sequential data set 7-42
  - more than one unit 7-29
  - multiple copies of an output data set 7-57
    - using JES2 7-57
    - using JES3 7-58
  - multiple units 7-29
  - printer form and character control 7-58
  - space for non-VSAM data sets 7-39
  - space for non-VSAM data sets on mass storage volumes 7-39
  - special character set using the UCS feature 7-59
  - special output form 7-59
  - storage 5-23
    - See also ADDRSPC Parameter
    - when to request 5-23
  - storage for execution 5-23
  - unit affinity 12-132
    - See also AFF subparameter of UNIT parameter
  - units and volumes 7-24
- requests to read or write a dummy data set 8-9
- RESERVE subparameter on DCB parameter 12-47
- reserved volumes
  - when requesting multiple units 7-29
- RESET operator command (JCL) 15-2
- resources
  - dynamic allocation of 4-12, 6-3
  - requesting 7-24-7-27
- restart facility
  - See checkpoint/restart facility
- RESTART operator command (JES3) 17-3
- using the RESTART parameter 5-28, 5-29
- RESTART parameter on /\*JOBPARM statement 16-6
- RESTART Parameter on JOB statement
  - See also RD Parameter
  - See also SYSCHK DD statement
  - cautions when using 10-34
    - general 10-34
  - examples 10-35
  - subparameter definition 10-34
  - syntax 10-33
  - use of 5-28
    - use of with generation data sets 5-28
- RESTART subparameter on JES3 //\*MAIN statement 17-27
  - See also FAILURE parameter on JES3 //\*MAIN statement
- restarting a job
  - See also RD Parameter
  - See also RESTART Parameter on JOB statement
  - See also SYSCHK DD statement
  - See also SYSCKEOV DD statement
  - automatic restart 5-26
  - checkpoint restart 5-26
  - deferred restart 5-27
  - step restart 5-26
- RETAIN subparameter of VOLUME parameter
  - description 12-136
  - use of 7-27
- retaining tape volumes
  - See RETAIN subparameter of VOLUME parameter
- retention period
  - See RETPD subparameter of LABEL parameter
- RETPD subparameter of LABEL parameter
  - description 12-92
  - use of 7-12
- retrieving a nontemporary data set 7-3
- retrieving a temporary data set
  - See temporary data set
- retrieving an existing private library 8-3
- retrieving an indexed sequential data set 8-22
- retrieving generation data groups residing on DASD volumes 7-39
- return code test, specifying on EXEC statement
  - See Specifying Return Code Tests
- return code test, specifying on JOB statement
  - See Specifying Return Code Tests
- rewinding tapes
  - for DELETE 7-15
  - for KEEP 7-16
- RINGCHK parameter on JES3 //\*MAIN statement 17-31
- RKP subparameter on DCB parameter
  - description 12-47
  - using with FCB parameter 12-78
  - using with UCS parameter 12-129
- RLSE subparameter of SPACE parameter 12-114
  - See also SPACE parameter
- RMnnnn parameter
  - on /\*SIGNON statement (JES2) 16-31
- RMnnnn subparameter
  - of DEST parameter
    - on /\*OUTPUT statement 16-17
    - on DD statement 12-54
    - on OUTPUT JCL statement 14-24
  - of PRINT, PUNCH, and XEQ parameters
    - on /\*ROUTE statement 16-25
- RMTnnnn parameter
  - on /\*SIGNON statement (JES2) 16-31
- RMTnnnn subparameter
  - of DEST parameter
    - on /\*OUTPUT statement 16-17

- on DD statement 12-54
  - on OUTPUT JCL statement 14-24
- of PRINT, PUNCH, and XEQ parameters
  - on /\*ROUTE statement 16-25
- RN character set
  - for 1403 12-128, 14-54
  - for 3203 Model 5 12-128, 14-54
- RNC subparameter of RD parameter
  - on Job statement 10-29
- Rnnnn
  - of DEST parameter
    - on DD statement 12-54
    - on OUTPUT JCL statement 14-24
- Rnnnn subparameter
  - of DEST parameter
    - on /\*OUTPUT statement 16-17
  - of PRINT, PUNCH, and XEQ parameters (JES2) 16-25
  - on /\*ROUTE statement 16-25
- ROOM parameter on /\*JOBPARM statement 16-6
- ROOM parameter on JES3 /\*NETACCT statement 17-40
- ROUND subparameter of SPACE parameter
  - See also SPACE parameter
  - description 12-115
  - requesting space 7-40
- routing a job in a network (JES2) 3-7
- routing a job in a network (JES3 networking) 3-10
- routing a job in a network, example of (JES3) 3-11
- routing output data sets (JES2) 3-7
- routing output data sets (JES3) 3-12
- routing system messages 3-14
  - with JES2 3-14
  - with JES3 3-14
  
- S operator command (JES3)
  - See START operator command
- S parameter (JES2)
  - See SYSAFF parameter on /\*JOBPARM statement
- S subparameter of STACKER parameter (JES3) 17-16
- SCAN subparameter of TYPRUN parameter 10-40
- scanning JCL for syntax errors 10-39
- scanning JCL without job execution 3-25
- scheduling a job 5-1
- scheduling, deadline for JES3 3-27
- SDG subparameter on JES3 /\*NET statement 17-36
- SDLC
  - See synchronous data link
- secondary quantity subparameter of SPACE parameter
  - See also SPACE parameter
  - description 12-113
  - requesting space 7-41
- secondary request for space 7-41
  - See also secondary quantity subparameter of SPACE parameter
- selecting a cataloged procedure library 4-7
  - See also PROC parameter, on EXEC statement
  - in a JES2 system 4-8
  - in a JES3 system 4-8
- selecting a processing program (See also PGM Parameter) 4-4
- selecting a processor 5-3
  - See also processor selection
- selecting jobs
  - for JES2 5-1
  - for JES3 5-2
- SEND operator command
  - for JCL 15-2
  - for JES3 17-3
- separating groups of data 12-6, 12-28
- SEQUENCE macro
  - See RESERVE subparameter on DCB parameter
- sequence number subparameter of LABEL parameter
  - See data-set-sequence-number subparameter of LABEL parameter
- sequence of DD statements
  - See concatenating data sets
- SER subparameter of VOLUME parameter
  - description 12-137
  - specific volume request 7-24
- SET operator command (JCL) 15-2
- SETDMN operator command (JCL) 15-2
- SETUP parameter on JES3 /\*MAIN statement 17-31
  - See also FETCH parameter on JES3 /\*MAIN statement
  - description 17-27
  - use of 3-17
- seven track tape
  - See TRTCH subparameter on DCB parameter
- shared control of a data set 7-20
- sharing a library 8-3
- sharing a unit between data sets on different volumes 7-32
  - See also unit and volume affinity
- sharing volumes between data sets 7-27
  - See also unit and volume affinity
- SHR subparameter of DISP parameter
  - data set status 7-13
  - description 12-58
  - maintaining integrity 7-19
- SINGLE subparameter on JES3 /\*FORMAT PR statement 17-12
- SINGLE subparameter on OUTPUT JCL statement 14-16
- SL subparameter of LABEL parameter
  - description 12-90
  - for direct access data sets 12-90
  - for tape volume data sets 12-90
  - on a SYSCKEOV DD statement 13-18
  - system action 12-93
  - system action with OPTCD 12-43
  - use of 7-9
  - with EXPDTCHK parameter (JES3) 17-27
  - with the PROTECT parameter 12-110
  - with VSAM 8-17
- SLIP operator command (JCL) 15-2
- SN character set
  - for 1403 12-128, 14-54
  - for 3203 Model 5 12-128, 14-54

# Index

- SNA remote work stations (JES3) 3-22
- SNA RJE for JES2 3-20
- space
  - example of requesting 7-43
  - for directory 7-42
  - for index 7-42
  - for non-VSAM data sets 7-43
  - for non-VSAM data sets on mass storage volumes 7-39
  - for partitioned data set 7-42
  - index, requesting for an indexed sequential data set 7-42
  - primary requests 7-40
  - request for non-VSAM data sets on mass storage volumes 7-39
  - requesting 7-39
  - requesting for a partitioned data set directory 7-42
  - requesting space 7-43
  - secondary request for 7-41
  - secondary requests 7-41
  - space requests 7-43
  - specific tracks 7-42
  - value converted to space request 7-39
- space on a printer
  - See PRTSP subparameter on DCB parameter
- SPACE parameter
  - creating a private library 8-2
  - description 12-112
  - examples 12-116
  - for mass storage volumes 7-37
  - for non-VSAM data sets 7-39
  - index subparameters 7-42
  - relationship to other parameters 12-115
  - requesting space 7-39
  - storing a dump 13-11
  - subparameter definition 12-113
  - syntax 12-112
  - use of with indexed sequential data sets 8-21
  - use with generation data sets 8-27
- SPART parameter on JES3 **//\*MAIN** statement
  - See also spool partitioning (JES3)
  - description 17-32
  - use of 3-33, 6-1
- special character set, requesting using the UCS feature 7-59
- special data sets
- special job processing 3-23
- special output form, requesting 7-59
- specific image
  - See FCB parameter
- specific tracks
  - assigning 7-42
  - cautions with ISAM data sets 7-43
  - for partitioned data set 7-42
  - how to request 7-42
- specific volume request
  - for mass storage volumes 7-38
  - how to make 7-24
  - restriction using multi-device type VSAM data set 8-15
  - system action 7-24
  - with space request 7-40
- Specific Volume Requests 7-24
- specifying
  - device for an output data set (JES2) 7-54
  - device for an output data set (JES3) 7-54
  - label type in the LABEL parameter 7-8
- Specifying a Disposition for the Data Set 7-14
- Specifying Data Set Processing Options 7-43
- Specifying Data Set Status 7-14
- Specifying Data Sets for Mass Storage Systems (MSS) 7-37
- Specifying Devices Dedicated to a Net 3-29
- Specifying Early Setup of Resources for a Job in a Net 3-29
- Specifying JES2 page overflow processing 7-56
- Specifying JES3 Forms Overflow Processing and Printer Spacing 7-56
- Specifying Return Code Tests 5-5
  - on JOB statement 5-5, 5-7
  - See also COND Parameter, on EXEC statement on the EXEC statement 5-7
  - relationship to the EXEC statement 5-6
- specifying storage requirements 5-24
  - See also REGION Parameter
- Specifying Storage Requirements with the REGION Parameter 5-24
- Specifying System Action for Termination of a Job in a Net 3-28
- Specifying the DDNAME Parameter 7-1
- Specifying the Device 7-52
- Specifying the DSNAME Parameter 7-2
- Specifying the DSNAME Parameter in Apostrophes 7-7
- Specifying the Internal Reader 7-52
  - allocating 7-52
  - control statements 7-52
    - /\*DEL** 7-53
    - /\*EOF** 7-52
    - /\*PURGE** 7-53
    - /\*SCAN** 7-53
  - example 7-53
  - use of 7-52
- Specifying the LABEL Parameter 7-7
- Specifying the Label Type 7-8
- Specifying the TIME parameter on the JES2 JOBPARM Statement 5-18
- Specifying Unit Information 7-28
- Specifying Volume Affinity When Using Multivolume Data Sets 7-35
- Specifying Volume Information 7-24
  - See also volume information
- spool partitioning (JES3)
  - See SPART parameter on JES3 **//\*MAIN** statement
- STACK subparameter on DCB parameter 12-47
- stacker bin
  - See STACK subparameter on DCB parameter
- STACKER parameter on JES3 **//\*FORMAT PR** statement
  - description 17-16

- stackers for printed output
  - See BURST parameter
  - See STACKER parameter on JES3 **//\***FORMAT PR statement
- standard job (JES3), definition 1-6
- standard labels
  - See SL subparameter of LABEL parameter
- STANDARD subparameter
  - of CHARS parameter 17-11
  - of FLASH parameter (JES3) 17-14
  - of FORMS parameter (JES3) 17-15, 17-21
  - of STACKER parameter (JES3) 17-16
  - of TRAIN parameter (JES3) 17-16
- START operator command
  - for JCL 15-2
  - for JES3 17-3
- status, data set 7-14
- STD
  - on **/\*JOBPARM** statement 16-5
  - on **/\*OUTPUT** statement 16-19
- STDWTR on OUTPUT JCL WRITER parameter 14-56
- STDWTR on sysout DD statement 12-121
- STD1, standard FCB image 7-61
- STD2, standard FCB image 7-61
- STD3, standard FCB image 7-61
- step restart
  - See also RD Parameter
  - See also RESTART Parameter on JOB statement for generation data groups 8-30
  - use of 5-26
- step setup in JES3
  - See types of JES3 setup, high watermark setup
- STEPCAT DD statement
  - description 13-7
  - example 13-7
  - master catalog 8-5
  - overrides 13-7
  - private catalog 8-5
  - syntax 13-7
  - user catalog 8-5
  - VSAM 8-14-8-17
- STEPLIB DD statement 8-4
  - concatenating private libraries 8-4
  - creating and retrieving a private library 8-1
  - description 13-8
  - effect with JOBLIB DD statement 8-1
  - examples 13-10
  - location in the JCL 13-9
  - relationship to JOBLIB 13-9
  - syntax 13-8
- stepname subparameter
  - in RESTART parameter 10-34
- stepname.procstepname subparameter
  - in RESTART parameter 10-34
- stepname, assigning 11-1
- STOP operator command (JCL) 15-2
- STOPMN operator command (JCL) 15-2
- storage requirements, real 5-24
- storage requirements, virtual 5-24
- storage, example of requesting 5-26
- storage, requesting
  - See requesting
- storage, specifying requirements
  - See specifying storage requirements
- STRNO subparameter of AMP parameter 12-14
- Submitting your job 1-7
- SUBSYS parameter on DD statement
  - description 12-117
  - effect on DD statement parameters 8-32
  - examples 12-118
  - relationship to other parameters 12-118
  - subparameter definition 12-117
  - syntax 12-117
- subsystem data set, creating and using 8-32
- subsystem data sets, creating and using 8-32
- subsystem-name subparameter
  - on SUBSYS parameter 12-117
- successor job
  - dependent job control 3-27-3-33
  - early setup of 3-27
- SUL subparameter of LABEL parameter 12-90
- suppressing the writing of an output data set 7-55
- SWITCH operator command (JES3) 17-3
- symbolic parameters
  - assigning default values 2-16
  - assigning values 2-15, 2-16, 2-17
  - defining when writing a procedure 2-15
  - example 2-15, 2-21
  - length restriction 2-17
  - nullifying 2-16, 2-18
  - use of commas 2-19
  - using 2-15
- SYNAD subparameter of AMP parameter 12-14
- synchronous data link 3-20
- syntax checking 3-25
  - for JES3 3-25
- SYSABEND DD statement
  - description 13-11
  - examples 13-13
  - location in the JCL 13-11
    - storing a dump 13-11
    - to print a dump 13-12
  - overriding dump requests 13-13
  - requesting abnormal termination dumps 8-6
  - syntax 13-11
  - system action 13-13
  - use of 8-6
- SYSAFF parameter on **/\*JOBPARM** statement
  - description 16-6
  - relationship to NOTIFY parameter 5-4
  - using 5-3
- SYSALLDA devices 12-132
- SYSCHK DD statement
  - See also RESTART Parameter on JOB statement
  - description 13-15
  - examples 13-17
  - location in the JCL 13-17
    - when the checkpoint data set is cataloged 13-15

# Index

- when the checkpoint data set is not cataloged 13-15
- syntax 13-15
- use of 5-27
- SYSCKEOV DD statement**
  - description 13-18
  - example 13-19
  - parameters 13-18
  - syntax 13-18
- SYSMDUMP DD statement**
  - description 13-11
  - examples 13-13
  - location in the JCL 13-11
    - storing a dump 13-11
    - to print a dump 13-12
  - overriding dump requests 13-13
  - requesting abnormal termination dumps 8-6
  - syntax 13-11
  - system action 13-13
  - use of 8-6
- SYSMSG subparameter on JES3 // \*FORMAT PR statement** 17-10
- SYSOUT parameter**
  - description 12-120
  - examples 12-123
  - print a dump 8-6
  - relationship to other parameters 12-122
  - relationships 12-123, 14-13
    - to a JES2 held class 12-123, 14-13
    - to MSGCLASS parameter 12-123, 14-14
  - requesting multiple copies (JES2) 7-57
  - requesting multiple copies (JES3) 7-58
  - subparameter definition 12-121
  - syntax 12-120
- system affinity, use of (JES2) 5-3
  - See also SYSAFF parameter on /\*JOBPARM statement
- system messages 3-14
  - See also MSGLEVEL parameter on JOB statement
- system messages (JES3) 3-18
- SYSTEM parameter on JES3 // \*MAIN statement** 17-32
  - using 5-4
- SYSUDUMP DD statement**
  - description 13-11
  - examples 13-13
  - location in the JCL 13-11
    - storing a dump 13-11
    - to print a dump 13-12
  - overriding dump requests 13-13
  - requesting abnormal termination dumps 8-6
  - syntax 13-11
  - system action 13-13
  - use of 8-6
- T operator command (JES3)**
  - See SEND operator command
- T parameter (JES2)**
  - See TIME parameter on /\*JOBPARM statement
- See UCS parameter on /\*OUTPUT statement
- table-name subparameter of CHARS parameter
  - on DD statement 12-18
  - on JES3 // \*FORMAT PR statement 17-11
- tape density
  - See DEN subparameter on DCB parameter
- tape device, eligible for allocation 7-26
- tapemark 7-8
- TCAM**
  - NOTIFY parameter on JOB statement 10-18
  - QNAME parameter on DD statement 12-111
  - TCAM subparameters of DCB parameter 12-36-12-48
  - TERM parameter on DD statement 12-125
- tcamname subparameter of QNAME parameter 12-111
- telecommunications access method
  - See TCAM
- teleprocessing device 7-24
- temporary data set
  - areas of an indexed sequential data set 7-6
  - creating 7-4
  - defining a VIO 8-10
  - definition 7-5
  - deletion, conditions of 7-5
  - DSNAME, copying 7-6
  - DSNAME, specifying in apostrophes 7-7
  - members of a partitioned data set 7-5
  - retrieving 7-4
  - specifying disposition
    - See DISP parameter on DD statement
  - using VIO for 8-10
  - using VIO to pass among job steps 8-12
- temporary library
  - creating and using 8-1
  - definition 4-5
  - use of 8-5
- TERM parameter**
  - description 12-125
  - examples 12-126
  - relationship to other parameters 12-125
  - syntax 12-125
- testing JCL without execution 3-25
  - See also TYPRUN Parameter, on JOB statement
- the basic space request: unit of measurement and primary quantity 7-40
- the RD parameter on the JOB statement 5-27
- the RESTART parameter on the JOB statement 5-28
- THRESH subparameter on DCB parameter 12-47
  - with OUTLIM parameter 12-101
- THRESHLD Parameter on OUTPUT JCL statement
- THRESHOLD parameter
  - on JES3 // \*FORMAT PR statement 17-16
- THWS subparameter
  - of SETUP parameter on JES3 // \*MAIN statement 17-31
- time limit for processor
  - on EXEC statement 11-32
  - on JOB statement 10-36
- TIME Parameter



- on /\*JOBPARM statement 16-7
- on EXEC statement 11-32
  - defaults 11-33
  - examples 5-18, 11-33
  - subparameter definition 11-32
  - syntax 11-32
  - system conversion 11-33
  - use of with cataloged procedures 5-17
- on JOB statement 10-36
  - examples 5-18
  - examples on JOB and EXEC statements 10-38
  - examples on JOB statement 10-37
  - subparameter definition 10-36
  - syntax 10-36
  - system conversion 10-37
- time-dependent program, requesting real storage 5-23
- TN character set 12-128, 14-54
  - for 1403 12-128, 14-54
  - for 3203 Model 5 12-128, 14-54
- TPROCESS macro 12-111
- TRACE subparameter of AMP parameter 12-14
- tracks
  - assigning specific 7-42
  - capacities 18-11, 18-12
  - cylinder index
    - See NTM subparameter on DCB parameter
  - description 12-113
  - number per cylinder 18-10
  - overflow
    - See CYLOFL subparameter on DCB parameter
  - physical records 18-11, 18-12
  - searching
    - See LIMCT subparameter on DCB parameter
  - space requests 7-39
- TRAIN parameter on JES3 /\*FORMAT PR statement 17-16
- train-name subparameter of TRAIN parameter (JES3) 17-16
- translation of ASCII tape data sets, requesting 7-9
- transmitting data in a network (JES2) 3-7
- TRC Parameter on OUTPUT JCL statement
- trc subparameter of MODIFY parameter
  - on /\*OUTPUT statement 16-20
  - on DD statement 12-95
  - on JES3 /\*FORMAT PR statement 17-15
  - on OUTPUT JCL statement 14-42
- trc subparameter of MODTRC parameter 16-20
- TRIPLE subparameter on JES3 /\*FORMAT PR statement 17-12
- TRIPLE subparameter on OUTPUT JCL statement 14-16
- TRK subparameter of SPACE parameter
  - See also SPACE parameter
  - description 12-113
  - space requests 7-39
- TRTCH subparameter on DCB parameter 12-48
- TSO, introduction 3-18
- TYPE parameter on JES3 /\*MAIN statement
  - description 17-33
  - selecting a processor 5-3
- type subparameter
  - of DEST parameter
    - on OUTPUT JCL statement 14-25
- type subparameter of DEST parameter
  - on JES3 /\*FORMAT PR statement 17-13
  - on JES3 /\*FORMAT PU statement 17-21
- types of JES3 setup
  - See also SETUP parameter on JES3 /\*MAIN statement
  - high watermark setup 6-6
  - job setup 6-5
- TYPRUN Parameter, on JOB statement
  - bypassing job initiation 3-25
  - copying JCL without execution 3-25
  - delaying job initiation (JES2) 3-23
  - examples 10-40
  - subparameter definition 10-39
  - syntax 10-39
  - syntax checking 3-25
  - use of 3-24
- T11 character set (3211) 12-128, 14-54
- UCS feature, using to request a special character set 7-59
- UCS parameter
  - on /\*OUTPUT statement 16-20
  - on DD statement 12-127
    - default 12-128
    - description 12-127
    - examples 12-129
    - relationship to other parameters 12-129
    - subparameter definition 12-127
    - syntax 12-127
  - on OUTPUT JCL statement 14-53
  - use of 7-59
- UCS Parameter on OUTPUT JCL statement
- uncataloging a data set 7-17
- UNCATLG subparameter of DISP parameter
  - description 12-59
  - use of 7-17
- underline, use of in control statements 2-3
- unit and volume affinity
  - examples 7-33, 7-35
  - positioning the request 7-34
  - requests 7-32
- unit and volume references, JES3 handling 7-31
- unit information
  - JES3 handling 7-31
  - relationship to a volume reference 7-30
  - relationship to volume reference 7-30
  - requesting deferred mounting of volumes 7-30
  - requesting more than one 7-29
  - sharing a unit between data sets 7-32
  - specifying 7-28
- unit of measurement, cylinders, blocks, and tracks 7-40
- UNIT parameter on DD statement
  - creating a private library 8-1
  - description 12-130
  - examples 12-134

# Index

- generation data groups 8-27, 8-29
- indexed sequential data sets 8-19, 8-22
- JES3 handling 7-31
- multiple units 7-28
- overrides 12-133
- relationship to other parameters 12-133
- requesting 7-24
- storing a dump 8-6
- subparameter definition 12-130
- syntax 12-130
- unit affinity 7-32
- use of 7-28
- use of with indexed sequential data sets 8-19
- use with generation data sets 8-27, 8-29
- using mass storage volumes 7-37
- write output data set 7-52
- unit record device 7-24
- unit-count subparameter of UNIT parameter 12-131
- unit, definition 7-24
- unit, specifying output device 7-44
- unit, types you can specify 7-28
  - See also UNIT parameter on DD statement
- units and volumes, example of requesting 7-31
- units and volumes, requesting
  - See requesting, units and volumes
- units, requesting multiple 7-29
- universal character set
  - See UCS parameter
- UNLOAD operator command (JCL) 15-2
- Unnn subparameter
  - of DEST parameter
    - on /\*OUTPUT statement 16-17
    - on DD statement 12-54
    - on OUTPUT JCL statement 14-24
  - of PRINT, PUNCH, and XEQ parameters
    - on ROUTE statement (JES2) 16-26
- UPDATE parameter on JES3 /\*MAIN statement
  - description 17-33
  - using 4-8
- The USER parameter on the JES3 MAIN control statement 3-19
- USER parameter on JES3 /\*MAIN statement
  - description 17-34
  - use of 3-18
- USER parameter on JOB statement
  - See also PASSWORD and GROUP parameters
  - example 10-42
  - subparameter definition 10-41
  - syntax 10-41
  - use of 6-2
- userid parameter
  - on /\*NOTIFY Statement 16-11
- USERID parameter on JES3 /\*NETACCT statement 17-40
- userid subparameter
  - of NOTIFY parameter on JOB statement 10-18
  - of USER parameter on JOB statement 10-41
- using
  - private catalogs 8-5
  - private volumes 7-27

- the COND parameter to force step execution 5-15
- the COND parameter with cataloged procedures 5-13
- the COND subparameters EVEN and ONLY 5-8
- the JES3 LREGION parameter to define logical storage 5-25
- the RESTART parameter with generation data sets 5-28
- the TIME parameter for cataloged procedures 5-17
- VIO to pass temporary data sets among job steps 8-12
- virtual input/output (VIO) for temporary data sets 8-10
- using cataloged and in-stream procedures 9-3

## V format

- See RECFM subparameter
- VARY operator command
  - for JCL 15-2
  - for JES3 17-3
- VERIFY subparameter of FCB parameter 12-78
- VERIFY subparameter of UCS parameter 12-127
- vertical bar 2-2
- VIO data sets, backward references to 8-11
- VIO, defining a temporary data set 8-10
- VIO, using for temporary data sets 8-10
- VIRT subparameter of ADDRSPC parameter
  - on EXEC statement 11-6
  - on JOB statement 10-7
  - requesting storage 5-23
- virtual input/output (VIO) for temporary data sets, using 8-10
- virtual storage 5-23
  - See also requesting
- virtual storage access method (VSAM) 8-14-8-17
- virtual storage requirements 5-24
- virtual volumes, mass storage system 7-37, 7-39
  - See also MSVGP parameter on DD statement
- volume
  - See also VOLUME parameter
  - deferred mounting 7-30
  - maximum volume request 7-28
  - multivolume requests 7-28
- volume affinity
  - examples 7-33
  - when using multivolume data sets 7-35
- volume and unit references, JES3 handling 7-31
- volume attributes 7-36
- volume information
  - multivolume data sets 7-28
  - nonspecific requests 7-26
  - private requests 7-27
  - sharing volumes 7-27
  - specific requests 7-24
  - specifying 7-24
- VOLUME parameter
  - creating a private library 8-1
  - description 12-135
  - examples 12-140

- for multivolume data sets 7-28
  - for volume affinity 7-27
  - nonspecific requests 7-26
  - overrides 12-139
  - relationship to other parameters 12-139
  - specific requests 7-24
  - subparameter definition 12-136
  - syntax 12-136
  - use of 7-24
  - use of with indexed sequential data sets 8-19
  - use with generation data sets 8-27, 8-29
  - volume requests, for mass storage 7-37
  - volume requests, nonspecific 7-26
  - volume requests, nonspecific for mass storage volumes 7-38
  - volume requests, specific 7-24
  - volume requests, specific for mass storage volumes 7-38
  - volume serial number
    - See also SER subparameter of VOLUME parameter
    - parameter on /\*SETUP statement (JES2) 16-28
    - subparameter of VOLUME parameter 12-137
  - volume-count subparameter of VOLUME parameter
    - description 12-137
    - maximum number of volumes you can request 12-137
  - volume-sequence-number subparameter of VOLUME parameter
    - description 12-136
    - use of 7-25
  - volumes and units, requesting
    - See requesting, units and volumes
  - volumes, private, using 7-27
  - volumes, sharing between data sets 7-27
  - VSAM data sets
    - for private catalogs 8-5
    - introduction 8-14
    - processing VSAM data sets 8-14-8-17
- W** subparameter on JES3 /\*MAIN statement  
 See WARNING subparameter on JES3 /\*MAIN statement
- wait-state time limit  
 See TIME Parameter, on EXEC statement  
 See TIME Parameter, on JOB statement
- WARNING subparameter on JES3 /\*MAIN statement 17-25, 17-28  
 See also CARDS parameter
- when to request real storage 5-23
- when you code the DDNAME parameter 7-1
- work station, controlling output to
  - for JES2 3-7
  - for JES3 3-12
- WRITE macro instruction 7-20
- WRITELOG operator command (JCL) 15-2
- WRITER Parameter on OUTPUT JCL statement
- writer-name subparameter
  - in SYSOUT parameter 12-121
- writing a dummy data set 8-9
- writing cataloged and in-stream procedures 9-1
- writing output data sets 7-44
- X** operator command (JES3)  
 See CALL operator command (JES3)
- X** parameter (JES2)  
 See CHARS parameter on /\*OUTPUT statement
- X/ (cataloged procedure) 9-13
- XEQ parameter on /\*ROUTE statement 16-24
- XN character set
  - for 1403 12-128, 14-54
  - for 3203 Model 5 12-128, 14-54
- XX (cataloged procedure) 9-13
- XX\* (cataloged procedure) 9-13
- Y** parameter (JES2)  
 See MODIFY parameter on /\*OUTPUT statement
- Y** subparameter of BURST parameter
  - on /\*JOBPARM statement 16-4
  - on /\*OUTPUT statement 16-14
  - on DD statement 12-16
  - on OUTPUT JCL statement 14-6
- Y** subparameter of RESTART parameter
  - on /\*JOBPARM statement 16-6
- YES** subparameter 12-87
  - of EXPDTCHK parameter (JES3) 17-27
  - of HOLD parameter 12-87
  - of HOLD parameter (JES3) 17-28
  - of INT parameter (JES3) 17-21
  - of J parameter (JES3) 17-5
- YN** character set
  - for 1403 12-128, 14-54
  - for 3203 Model 5 12-128, 14-54
- Z** operator command (JES3)  
 See MESSAGE operator command (JES3)
- 1403
  - requesting a special character set 7-60
  - requesting specific forms control 7-60
- 1440 subparameter of TIME parameter
  - on EXEC statement 11-32
  - on JOB statement 10-36
- 3203 Model 5
  - how JES2 handles 7-59
  - requesting a special character set 7-60
  - requesting specific forms control 7-60
- 3211
  - indexing feature, JES2 support of 7-57
  - requesting a special character set 7-60
  - requesting specific forms control 7-60
- 3330 Model 11
  - coding 12-131
- 3330V virtual volume 7-36, 12-98  
 See also MSVGP parameter on DD statement
- 3340 fixed head feature 7-29

## Index

- 3348 model 70F data module 7-29
- 3525 punch interpretation 7-57
- 3540 Diskette 7-6
  - See also DSID parameter on DD statement
- 3800 printer
  - bursting output 7-62
    - See also BURST parameter
    - See also STACKER parameter on JES3  
//\*FORMAT PR statement
  - forms control 7-61
  - obtaining output for JES2 7-43
    - See also /\*OUTPUT Statement
  - obtaining output for JES3 7-43
    - See also JES3 /\*FORMAT PR statement
  - printing a dump with more data per page 8-6
  - requesting character arrangements 7-60
    - See also CHARS parameter
    - requesting copy modification 7-58
    - See also MODIFY parameter
    - requesting forms overlay 7-62
    - See also FLASH parameter
    - requesting multiple copies 7-57
    - See also COPIES parameter
    - using JES2 7-57
    - using JES3 7-58
    - stacking the paper output 7-43
    - See also BURST parameter
- 3850 Mass Storage System 7-24
  - See also MSVGP parameter on DD statement
- 6, FCB image 7-61
- 8, FCB image 7-61





This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

**Note:** *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity    Accuracy    Completeness    Organization    Coding    Retrieval    Legibility

If you wish a reply, give your name, company, mailing address, and date:

---

---

---

---

What is your occupation? \_\_\_\_\_

How do you use this publication? \_\_\_\_\_

Number of latest Newsletter associated with this publication: \_\_\_\_\_

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Reader's Comment Form

Fold and tape

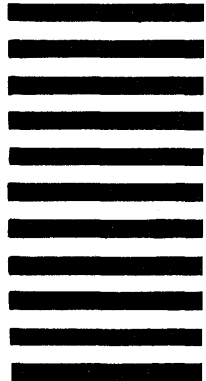
Please Do Not Staple

Fold and tape



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**  
FIRST CLASS PERMIT 40 ARMONK, NEW YORK



POSTAGE WILL BE PAID BY ADDRESSEE:

**International Business Machines Corporation**  
**Department D58, Building 920-2**  
**PO Box 390**  
**Poughkeepsie, New York 12602**

Fold and tape

Please Do Not Staple

Fold and tape

